Synthetic Data for Machine Learning

Abdulrahman Kerim, BSc (Hons), BSc (Hons), MSc

School of Computing and Communications

Lancaster University

A thesis submitted for the degree of

Doctor of Philosophy

August, 2024

Synthetic Data for Machine Learning

Abdulrahman Kerim, BSc (Hons), BSc (Hons), MSc.

School of Computing and Communications, Lancaster University

A thesis submitted for the degree of *Doctor of Philosophy.* August, 2024.

Abstract

Supervised machine learning methods require large-scale training datasets to converge. Collecting and annotating training data is expensive, time-consuming, error-prone, and not always practical. Usually, synthetic data is used as a feasible data source to increase the amount of training data. However, just directly using synthetic data may actually harm the model's performance or may not be as effective as it could be. This thesis addresses the challenges of generating large-scale synthetic data, improving domain adaptation in semantic segmentation, advancing video stabilization in adverse conditions, and conducting a rigorous assessment of synthetic data usability in classification tasks. By contributing novel solutions to these multifaceted problems, this work bolsters the field of computer vision, offering strong foundations for a broad range of applications for utilizing synthetic data for computer vision tasks.

In this thesis, we divide the study into three main problems: (i) Tackle the problem of generating diverse and photorealistic synthetic data; (ii) Explore synthetic-aware computer vision solutions for semantic segmentation and video stabilization; (iii) Assess the usability of synthetically generated data for different computer vision tasks.

We developed a new synthetic data generator called *Silver*. Photo-realism, diversity, scalability, and full 3D virtual world generation at run-time are the key aspects of this generator. The photo-realism was approached by utilizing the state-of-the-art High Definition Render Pipeline (HDRP) of the Unity game engine. In parallel, the Procedural Content Generation (PCG) concept was employed to create a full 3D virtual world at run-time, while the scalability (expansion and adaptability) of the system was attained by taking advantage of the modular approach followed as we built the system from scratch. *Silver* can be used to provide clean, unbiased, and large-scale training and testing data for various computer vision tasks.

Regarding synthetic-aware computer vision models, we developed a novel architecture specifically designed to use synthetic training data for semantic segmentation domain adaptation. We propose a simple yet powerful addition to DeepLabV3+ by using weather and time-of-the-day supervisors trained with multi-task learning, making it both weather and nighttime-aware, which improves its mIoU accuracy under adverse conditions while maintaining adequate performance under standard conditions.

Similarly, we also proposed a synthetic-aware adverse weather video stabilization algorithm that dispenses real data for training, relying solely on synthetic data. Our approach leverages specially generated synthetic data to avoid the feature extraction issues faced by current methods. To achieve this, we leveraged our novel data generator to produce the required training data with an automatic ground-truth extraction procedure.

We also propose a new dataset called VSAC105Real and compare our method to five recent video stabilization algorithms using two benchmarks. Our method generalizes well on real-world videos across all weather conditions and does not require large-scale synthetic training data.

Finally, we assess the usability of the generated synthetic data. We propose a novel usability metric that disentangles photorealism from diversity. This new metric is a simple yet effective way to rank synthetic images. The quantitative results show that we can achieve similar or better results by training on 50% less synthetic data. Additionally, we qualitatively assess the impact of photorealism and evaluate many architectures on different datasets for that aim.

Acknowledgements

I would like to thank my supervisor, Dr. Leandro Soriano Marcolino, for his generous support, guidance, and patience. His kindness has been a constant source of encouragement, making the challenges more manageable and the journey more exciting. I feel truly fortunate to have had such a kind and supportive supervisor by my side, and I extend my sincere appreciation for his invaluable role in my academic and personal growth. I am sincerely grateful to my second supervisor Dr. Richard Jiang for his valuable guidance. I am also profoundly grateful to my co-supervisor Dr. Erickson Nascimento (UFMG/Microsoft) for his invaluable support and exceptional mentorship throughout my PhD journey. His expertise and insight have significantly enriched the depth and quality of my research. I extend my sincere appreciation for the time, knowledge, and encouragement to all my supervisors, mentors, and colleagues.

I would like to thank the Faculty of Science and Technology of Lancaster University for generously funding my PhD and the High-End Computing facility of Lancaster University for providing the computing resources.

I would like to thank my esteemed research collaborators Dr. Washington L. S. Ramos (Google) and Felipe Cadar Chamone (UFMG) for dedication, expertise, and collaborative spirit which have been pivotal in achieving this milestone in my academic journey. I also thank the anonymous reviewers for their constructive feedback that helped to improve my papers.

I would like to extend my sincere gratitude to my family - father and mother. Additionally, the two extraordinary individuals who have been my pillars of strength and support throughout this journey - my loving wife Emeni and our precious daughter Taj.

Declaration

I declare that the work presented in this thesis is, to the best of my knowledge and belief, original and my own work. The material has not been submitted, either in whole or in part, for a degree at this, or any other university. This thesis does not exceed the maximum permitted word length of 80,000 words including appendices and footnotes, but excluding the bibliography. A rough estimate of the word count is: 30134

Abdulrahman Kerim

Publications

The following publications, shown below, has been created directly from the thesis, from which large portions of these published works are used within this thesis:

Abdulrahman Kerim, Washington LS Ramos, Leandro Soriano Marcolino, Erickson R Nascimento, and Richard Jiang (2024). "Leveraging Synthetic Data to Learn Video Stabilization Under Adverse Conditions". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*

Abdulrahman Kerim, Felipe Chamone, Washington Ramos, Leandro Soriano Marcolino, Erickson R Nascimento, and Richard Jiang (2022). "Semantic Segmentation under Adverse Conditions: A Weather and Nighttime-Aware Synthetic Data-Based Approach". In: *British Machine Vision Conference (BMVC)*

Abdulrahman Kerim, Leandro Soriano Marcolino, Erickson R Nascimento, and Richard Jiang (2024). "On the Usability Usability of Synthetic Data for Image Classification". In: *Under Review - Computer Vision and Pattern Recognition (CVPR)*

Abdulrahman Kerim, Leandro Soriano Marcolino, and Richard Jiang (2021). "Silver: Novel Rendering Engine for Data Hungry Computer Vision Models". In: *2nd International Workshop on Data Quality Assessment for Machine Learning*

The following book has been written while developing this thesis, and to an extent has guided the thesis into what it has become:

Abdulrahman Kerim (2023). *Synthetic Data for Machine Learning: Revolutionize your approach to machine learning with this comprehensive conceptual guide.* Packt Publishing. ISBN: 9781803232607. URL: `https://books.google.co.uk/`

books?id=JpXeEAAAQBAJ

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The recent remarkable achievements of Artificial Neural Networks (ANN) in addressing complex problems have motivated researchers to extend their application to machine perception or computer vision challenges. Simultaneously, advancements in chip design, microelectronics, and the advent of General-Purpose Graphics Processor Architectures, such as the General Purpose Graphics Processing Unit (GPGPU), have facilitated the training of deep neural networks with millions or yet billions of parameters.

These sophisticated ANNs, characterized by their heightened non-linearity, excel in accurately approximating the underlying functions or phenomena inherent in complex vision problems, including semantic segmentation, instance segmentation, and object recognition. Regrettably, the training of these deep learning models demands an extensive volume of data alongside their corresponding annotations or ground-truths. The process of locating, gathering, and annotating suitable data is burdensome, time-intensive, error-prone, costly, and raises privacy concerns.

The deficiency in diverse, high-quality, and precisely labeled data can be attributed to the aforementioned challenges. Unfortunately, these issues contribute significantly to major data quality concerns within the realm of computer vision, serving as a notable impediment to achieving optimal performance in practical computer vision models.

Thus, in response to these challenges, we introduce novel synthetic data-generation pipelines and a new usability metric that ranks synthetic images based on their usability. We propose new synthetic-aware architectures and training procedures to better leverage synthetic data. Furthermore, we also address the significance of selecting the right images for training ML models, emphasizing the balance between photorealistic representation and diverse content. In this thesis, we focus specifically on three fundamental downstream computer vision tasks: semantic segmentation, video stabilization, and classification.

1.1 Research Questions

This thesis tries to answer the following essential research questions:

- Can leading-edge game engines like Unity effectively address data deficiency in computer vision by enabling the development of a real-time, three-dimensional data generator for synthetic datasets?
- Can we develop methods that can leverage synthetic data in a better way for three major computer vision tasks: classification, semantic segmentation, and video stabilization?
- How effective our developed methods are and why?

1.2 Synthetic Data Generator

A promising solution to the data deficiency problem seems to be in the leading-edge game engines like Unity (Unity, 2024), Unreal Engine (Unreal Engine, 2024), and CryEngine (Cry Engine, 2024). Thus, leveraging the powerful tools of the Unity game engine, we developed a new data generator that creates three-dimensional, photo-realistic virtual worlds, procedurally at run-time (see Chapter 4). The generator supports various computer vision tasks such as semantic segmentation, instance segmentation, depth estimation, video stabilization, and others. It allows researchers, with no computer graphics background, to generate large-scale synthetic datasets for training or testing their own computer vision models. *Silver*, the data generator proposed in this thesis, is a pioneer work using HDRP of the Unity game engine to build a 3D full and detailed virtual world at run-time for synthetic data generation. Unlike other frameworks, *Silver* considers data generation as an online problem which opens the door for using synthetic data with online learning algorithms, too. *The work presented in Chapter 4 appeared as a workshop paper in Kerim*, *Soriano Marcolino*, and *R. Jiang (2021).* The source code for *Silver* is accessible on GitHub at the following link: `https://github.com/A-Kerim/Silver`.

1.3 Synthetic-Aware Semantic Segmentation

To investigate the usability of our generated synthetic data and the efficacy of our synthetic-aware computer vision approach, we developed a synthetic-aware semantic segmentation architecture (see Chapter 5). In more detail, we developed a novel synthetic-aware training procedure that can be used to train on both synthetic and real data simultaneously. In particular, we significantly improved *DeepLabV3+* (L.-C. Chen, Y. Zhu, et al., 2018) robustness on adverse conditions by making its encoder both weather and nighttime aware. We also extended the *Silver* simulator (see Chapter 4) to generate more photo-realistic and diverse adverse weather conditions and increase the supported semantic segmentation classes. Leveraging our extended version of *Silver*, we generated a new synthetic semantic segmentation dataset, the *AWSS*, composed of photo-realistic annotated images spanning foggy, rainy, and snowy weather conditions and nighttime attributes. *The work presented in Chapter 5 appeared as a conference paper in Kerim*, *Chamone, et al. (2022).* The source code for *Silver* is accessible on GitHub at the following link: `https://github.com/lsmcolab/Semantic-Segmentation-under-Adverse-Conditions`.

1.4 Synthetic-Aware Video Stabilization

While synthetic data has been used successfully as a complement or alternative to real data, developing computer vision architectures that are specifically targeted to leverage synthetic data is still overlooked in the field. Thus, in this thesis we propose a novel synthetic-aware video stabilization method, achieving state-of-the-art results on real videos while being trained only on synthetic videos (see Chapter 6). We also extend the *Silver* generator (see Chapter 4) to be capable of producing specially designed training videos for video stabilization task. Furthermore, we provide a new video stabilization dataset, *VSAC105Real*, composed of real videos spanning foggy, rainy, snowy weather, and nighttime attributes. This dataset helps researchers to accurately assess the performance of video stabilization algorithms under adverse conditions. *The work presented in Chapter 6 appeared as a conference paper in Kerim, Ramos, et al. (2024).* The source code for *Silver* is accessible on GitHub at the following link: `https://github.com/A-Kerim/SyntheticData4VideoStabilization_WACV_2024`.

1.5 Usability of Synthetic Data

While synthetic data has been showing great progress recently, there is still a great urge to assess the usability of these synthetically generated data. For that aim, we proposed a novel training procedure and usability metric that disentangles photorealism from diversity. We show that our metric is a simple yet effective way to rank synthetic images based on their usability (see Chapter 7). Furthermore, we propose a new pipeline for generating synthetic data by integrating Large Language Models, specifically Chat Generative Pre-trained Transformer (*ChatGPT-3.5*) (OpenAI, 2023), with Stable Diffusion (Rombach et al., 2022). The quantitative results show we can achieve similar or better results by training on 50% less synthetic data. Additionally, we quantitatively assess the impact of photorealism on synthetic data usability. We perform an extensive set of experiments by evaluating six different architectures on three different datasets to assess the effectiveness of our metric and approach. *The work presented in Chapter 7 has been submitted to CVPR as a conference paper (currently under review) as detailed in Kerim, Marcolino, et al. (2024).*

1.6 Navigating the Dissertation

This thesis is structured into eight chapters to provide a comprehensive understanding of the research presented in this thesis. In Chapter 2, we provide the necessary background to understand the research presented in this thesis. We then delve into the literature review in Chapter 3, where we examine relevant synthetic data generation methods, semantic segmentation, and video stabilization state-of-the-art methods, breakthroughs, emerging trends, and existing gaps in the field. Chapter 4 outlines our synthetic data generator. Moving forward, Chapter 5 presents our synthetic-aware semantic segmentation methodology. Similarly, Chapter 6 illustrates our synthetic-aware approach to video stabilization. Finally, Chapter 7 presents our second novel synthetic data generation and training pipeline, usability metric, and its applications. Finally, in the concluding chapter, Chapter 8, we provide our final thoughts, conclusions, possible extensions, and discussions for future work directions.

Chapter 2

Background

Any structure must have a strong foundation. The cornerstones anchor the foundation. For some reason the cornerstones that I chose to begin with I never changed.

John Wooden

In this chapter, we discuss the relevant background for this thesis. we will embark on a comprehensive introduction to machine learning (ML) in Section 2.1. Building upon this understanding, we will address the contemporary challenges that permeate the landscape of modern ML, setting the stage for a focused examination in subsequent sections.

Section 2.2 will critically analyze the challenges and complexities associated with the annotation of real data, shedding light on the details that accompany the process.

Moving forward, Section 2.3 will illustrate the methodologies of ground truth generation, a crucial element in ensuring the accuracy and reliability of ML models. Section 2.4 will pivot towards introducing synthetic data generation and exploring innovative approaches to generate artificial datasets. Following this, Section 2.5 defines and briefly introduces *Realism*, *Photorealism*, and *Diversity* in synthetic data and ML.

Sections 2.6 and 2.7 will extend the discussion into the realms of semantic segmentation and video stabilization, providing a solid understanding of these advanced applications, respectively. Finally, in Section 2.8, the chapter will draw to a close with a concise overview of synthetic data usability metrics, encapsulating the essential criteria for evaluating the efficacy of synthetic datasets in the realm of ML.

This comprehensive exploration aims to equip the reader with a multifaceted understanding of key concepts and challenges, laying the groundwork for subsequent chapters.

2.1 Artificial Intelligence, Machine Learning, and Deep Learning

Artificial intelligence (AI) is an area of computer science that study the creation of intelligent machines (Russell and Norvig, 2016). AI involves replicating human-like intelligence in computers or devices. It is focused on creating software that performs tasks that usually require intelligence when performed by people (Kurzweil et al., 1990). This involves complex tasks such as perception (Boff, Kaufman, and Thomas, 1986; Hosseini, 2024), reasoning and generalization (Evans, 2002; Johnson-Laird, Khemlani, and Goodwin, 2015), planning, and interaction with the environment. Despite our remarkable progress in understanding these processes, much remains unknown about the intricacies of human intelligence (Neisser et al., 1996), including functions like vision and reasoning. Consequently, the aim to develop *intelligent* machines is a relatively recent in human history.

Learning-based AI, particularly ML, has emerged as a fundamental direction in AI research. ML is a subset of AI that enables computer programs to learn from experience, without explicit rules prescribed by humans. This contrasts with traditional rule-based AI, which is often challenging and time-consuming to engineer, especially when dealing with complex tasks like computer vision and natural language processing.

ML encompasses various types, including supervised, unsupervised, and reinforcement learning, each tailored to different learning paradigms and problem domains. Deep learning (DL), a subset of ML, has become central to many remarkable AI applications. DL utilizes artificial neural networks (ANNs) inspired by the human brain, with multiple layers that progressively learn abstract representations of data. This hierarchical learning enables DL models to discern complex patterns and structures in data.

The depth of ANNs, reflected in the number of layers, directly influences their capacity to model complex relationships in data. For instance, well-known and state-of-the-art DL models like AlexNet, VGGNet, ResNet-50, Transformers (e.g., ViT (Dosovitskiy, Beyer, et al., 2020) and Swin Transformer (Z. Liu et al., 2021)) consist of increasingly deeper architectures, allowing them to capture more sophisticated patterns and correlations from the training data which help models to achieve more accurate predictions.

The main issue with state-of-the-art DL models is that they require a large-scale training dataset to converge because they usually have a tremendous number of parameters (i.e., weights and biases) to tweak to minimize the loss. In ML, loss is a way to penalize wrong predictions. At the same time, it is an indication of how well the model is learning the training data. Collecting and annotating such large datasets is extremely hard and expensive.

Nowadays, using synthetic data as an alternative or complementary to real data is a hot topic. It is a trending topic in research and industry. Many companies such as Google (Google's Waymo utilizes synthetic data to train autonomous cars) and Microsoft (they use synthetic data to handle privacy issues with sensitive data) started recently to invest in using synthetic data to train next-generation ML models (Merfeld, Wilhelms, and Henkel, 2019).

2.2 Issues with Real Data Annotation Process

As we have seen so far, annotations are critical to both training and testing. Thus, any mislabeling, biased annotations, or insufficient annotated data will drastically impact the learning and evaluation process of your ML model. As you can expect, the annotation process is time-consuming, expensive, and error-prone, and this is what we will see in this section.

2.2.1 The annotation process is expensive

To train state-of-the-art computer vision or natural language processing (NLP) models, you need large-scale training data. For example, BERT (Devlin et al., 2018) was trained on BooksCorpos (800 million words) and Wikipedia (2,500 million words). Similarly, ViT (Dosovitskiy, Beyer, et al., 2020) was trained on ImageNet (14 million images) and JFT (303 million images). Annotating such huge datasets is extremely difficult and challenging.

Furthermore, it is time-consuming and expensive. It should be noted that the time required to annotate a dataset depends on three main elements:

- The task or problem;
- Dataset size;
- Granularity level.

Next, we will be looking at each of these in more detail.

2.2.1.1 Task

For example, annotating a dataset for a binary classification problem is easier and requires less time compared to annotating a dataset for semantic segmentation. Thus, the nature of the task also imposes clear difficulty on the annotation process. Even for the same task, let us say semantic segmentation, annotating a single image under standard weather conditions and normal illumination takes approximately

90 minutes for the Cityscapes dataset (Cordts et al., 2016). However, doing similar annotation for images under adverse conditions such as snow, rain, and fog or at low illumination such as nighttime takes up to three hours for the ACDC dataset (Sakaridis, D. Dai, and Van Gool, 2021).

2.2.1.2 Dataset size

As expected, the larger the dataset, the harder to annotate. The complexity comes from managing such a huge dataset and ensuring the same annotation and data collection protocol is being followed by a large group of annotators. These annotators may have different languages, backgrounds, experiences, and skills. Indeed, guiding such a huge, diverse team, probably in different geographical locations, is not simple.

2.2.1.3 Granularity level

The more detail you want your ground truth to capture, the more work for the annotators to perform. Let us take visual object tracking as an example. Annotating images for single-object tracking is easier than multi-object tracking. We find the same thing for semantic segmentation, too. Annotating a semantic segmentation dataset with three classes is easier than ten classes. Furthermore, the type of class also creates difficulty for the annotator. In other words, small objects may be harder to differentiate from the background and thus harder to annotate. Next, we look at the main reasons behind noisy ground truth issues commonly seen in real datasets.

2.2.2 The annotation process is error-prone

In this section, we shed light on the key reasons behind issues in manually annotated real data. These reasons inlcude:

- Human factor;
- Recording tools;
- Scene attributes;

- Annotation tools.

2.2.2.1 Human factor

The most important element in the annotation process is humans. However, we are limited by our perceptions of the world. Humans struggle to perceive with the naked eye the visual content in scenarios such as low illumination, cluttered scenes, or when objects are far from the camera, transparent, and so on. At the same time, miscommunication and misunderstanding of annotation protocol is another major issue. For example, assume you asked a team of annotators to annotate images for a visual object-tracking training dataset. You aim only to consider the person object for this task. Some annotators will annotate humans without objects while other annotators may consider other objects carried by humans as part of the object of interest (see Figure 2.1). Furthermore, some annotators may consider only the unoccluded part of the human. This will cause a major inconsistency in the training data and the model will struggle to learn the task and will never converge.

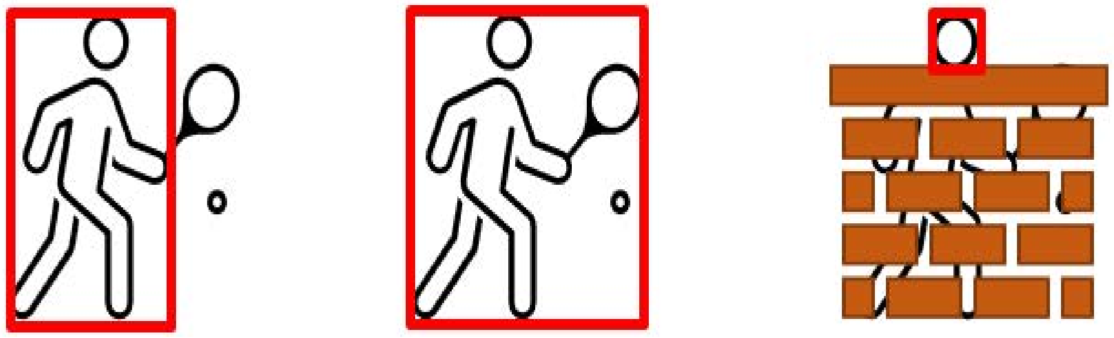


Figure 2.1: Samples of annotation errors due to unclear annotation protocol.

2.2.2.2 Recording tools

If the recoding camera is shaky, the captured images will be blurred and thus the annotators will fail to accurately identify the actual pixels of the object from the background. Furthermore, the intrinsic and extrinsic parameters of the camera drastically change how the 3D scene will be projected into a 2D image. The focal length of the lens, shutter speed, lens distortion, and others all introduce certain errors in the annotation process. In other words, the objects annotated by annotators may not exactly correspond to the same object in the raw image or even in the 3D world.

2.2.2.3 Scene attributes

Attributes such as weather conditions and time of the day all play an important role in the annotation process. As we have mentioned earlier, clear weather in the daytime may help the annotators to clearly identify objects as compared to adverse conditions at nighttime. In parallel to this, crowded and cluttered scenes are much more difficult to annotate and more subject to errors (see Figure 2.2).

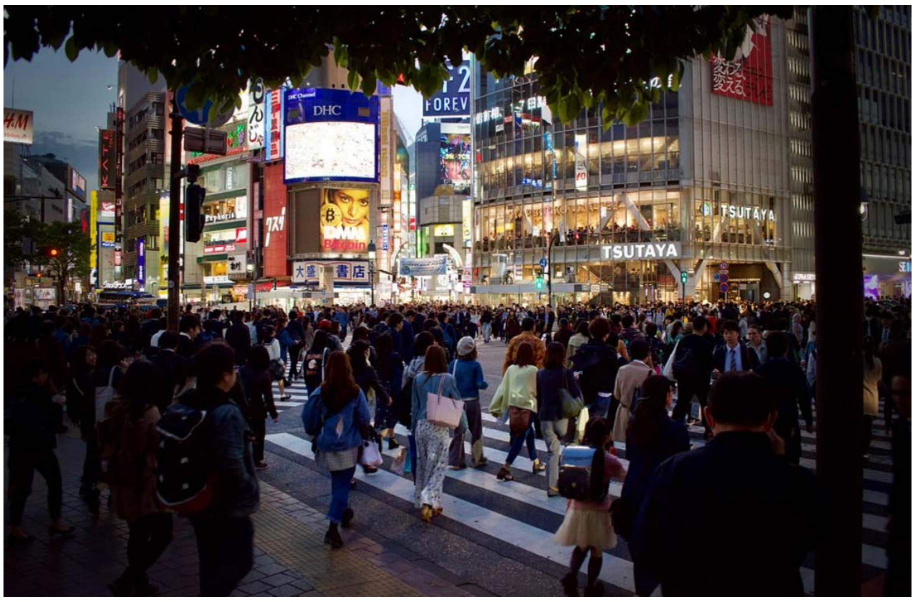


Figure 2.2: Scene attribute: crowded scenes are more subject to annotation errors.

2.2.2.4 Annotation tools

To enhance the annotation process, there are various annotation tools, such as Labelbox, Scale AI, Dataloop, HiveData, and LabelMe. Some of the annotation tools integrate AI components to optimize the annotation process by assisting the human annotator, such as Labelbox. While these AI-assisted methods are promising, they are not practical and reliable yet. In other words, the human annotator still needs to verify and correct the predictions. Additionally, some of these methods are slow and far from able to provide real-time assistance. In addition to this, if the problem is novel, the AI assistance will not work as expected because the AI model was not trained on similar scenarios. Given the task, dataset size, and team specifications, a suitable annotation tool should be selected. The annotation tool should be the same for all annotators to ensure consistency among the annotators and the created ground truth.

2.2.3 The snnotation process is biased

To understand the world and to reason about it efficiently, our brains build fast decisions and judgments based on our previous experience and systems of beliefs (Dietrich, 2010). ML models learn to reason and perceive the world using the training data. We try to collect and annotate the data objectively. However, unintentionally, we reflect our biases on the data we collect and annotate. Consequently, ML models also become biased and unfair. We will discuss the three common factors of annotation process bias next:

- Understanding the problem and task
- Background, ideology, and culture
- Subjectivity and emotions

2.2.3.1 Understanding the problem and task

The annotator may not know the problem, may not understand the data, or understand why the data is collected and annotated. Thus, they may make wrong assumptions or misinterpret data. Furthermore, given the differences between the annotators, they may understand the problem differently.

2.2.3.2 Background, ideology, and culture

This is a critical factor behind inconsistency in the annotation process. Let us imagine that you asked a group of 10 annotators to annotate a dataset for action recognition. You have only two actions: confirmation or negation. Your annotation team members are from the UK, Bulgaria, and India. The Bulgarian annotators will understand and annotate head shaking as "Yes" and nodding as "No." The other annotators will do the opposite. Thus, you will have wrong training data and your model will not learn this task. There are also other scenarios where the bias is not clear and cannot be easily identified, and this is the hardest issue under this scope.

2.2.3.3 Subjectivity and emotions

For some problems such as text sentiment analysis, which is a well-known NLP technique used to understand textual data, a human annotator may be biased toward certain political parties, football teams, genders, and ethnicities. Thus, the annotations will be biased to the annotator's point of view as well.

2.3 Ground Truth Generation Procedure for Computer Vision

In this section, we will look at different ML tasks and the followed procedures to generate their corresponding ground truth.

Computer vision aims at enabling computers to see using digital images. It is not surprising to know that vision is one of the most complex functionalities performed by our brain. Thus, imitating vision is not simple, and it is rather complex for state-of-the-art computer vision models.

Computer vision tasks include semantic segmentation, instance segmentation, optical flow estimation, depth estimation, normal map estimation, visual object tracking, and many more. Each task has its own unique way of generating the corresponding ground truth. Next, we will see samples of these tasks.

2.3.1 Image classification

The training images for this task usually contain one object, which is the object of interest. The annotation for this task is simply looking at each image and selecting one class or more describing the object in the image.

2.3.2 Semantic and instance segmentation

For semantic and instance segmentation, the annotator needs to assign a class label for each pixel in the image. In other words, the annotator is asked to partition the image into different segments where each segment demonstrates one class for semantic segmentation and one instance for instance segmentation.

2.3.3 Object detection and tracking

In object detection and tracking, the annotator draws a bounding box around each object in the image. Object tracking works using video to track an initially selected object throughout the video. On the other hand, object detection works on images to detect objects as required by the task.

2.3.4 Optical flow estimation

Optical flow is the relative apparent motion of objects from one frame to another. The motion could be because of objects or camera motion. Optical flow has many key applications in tasks, such as structure from motion, video compression, and video stabilization. Structure from motion is widely used in 3D construction, and navigation and manipulation tasks in robotics, augmented reality, and games. Video compression is essential for video streaming, storage, and transmission; video stabilization, on the other hand, is crucial for timelapse videos, and videos recorded by drones or head-mounted cameras. Thus, optical flow has enormous applications in practice.

Please note that it is extremely hard to generate ground truth for optical flow. Some approaches apply complex procedures to achieve this under many assumptions, such as an indoor environment and a limited number of objects and motions.

2.3.5 Depth estimation

Depth estimation is the task of measuring the distance of each pixel in the scene to the camera. It is essential for 3D vision and has many applications, such as 3D scene reconstruction, autonomous cars and navigation, medical imaging, and augmented reality. Usually, there are two major approaches for depth estimation: one uses monocular images and the other is based on stereo images utilizing epipolar geometry. Similar to optical flow, generating ground truth for depth estimation is extremely hard in the real world. Therefore, for optical flow and depth estimation, most of the standard datasets and benchmarks used are synthetic datasets. For optical flow, we can recognize synthetic datasets such as FlyingChairs (Dosovitskiy, Fischer, et al., 2015), FlyingThings3D (Mayer et al., 2016), and Kubric (Greff et al., 2022). For depth estimation, we can mention Virtual KITTI (Gaidon et al., 2016), DENSE (Javier Hidalgo-Carrio and Scaramuzza, 2020), and DrivingStereo (G. Yang et al., 2019).

2.4 Synthetic Data Generation

Computer vision, a field dedicated to endowing machines with the ability to interpret and understand visual information, relies heavily on the availability of high-quality training data. However, the traditional methods of collecting and annotating large-scale datasets for diverse computer vision tasks pose significant challenges. To overcome these challenges, researchers have turned to synthetic data generation as a compelling alternative, offering the means to create artificial datasets that mirror real-world scenarios. This section provides an in-depth exploration of various synthetic data generation approaches in computer vision, delving into the methodologies and advancements that underpin this transformative paradigm. In this part, we will introduce the main synthetic data generation approaches:

- Simulators and rendering engines;
- Generative adversarial networks;
- Video games;
- Diffusion models.

2.4.1 Leveraging simulators and rendering engines to generate synthetic data

2.4.1.1 Simulators

A simulator is a software or a program written to imitate or simulate certain processes or phenomena of the real world. Simulators usually create a virtual world where scientists, engineers, and other users can test their algorithms, products, and hypotheses. At the same time, you can use this virtual environment to help you learn about and practice complex tasks. These tasks are usually dangerous and very expensive to perform in the real world. For example, driving simulators teach learners how to drive and how to react to unexpected scenarios such as a child suddenly crossing the street, which is extremely dangerous to do in the real world.

Simulators are used in various fields, such as aviation, healthcare, engineering, driving, space, farming, and gaming. In Figure 2.3, you can find examples of these simulators.

2.4.1.2 Rendering and game engines

Renders and game engines are software used mainly to generate images or videos. They are composed of various subsystems responsible for simulating, for example,

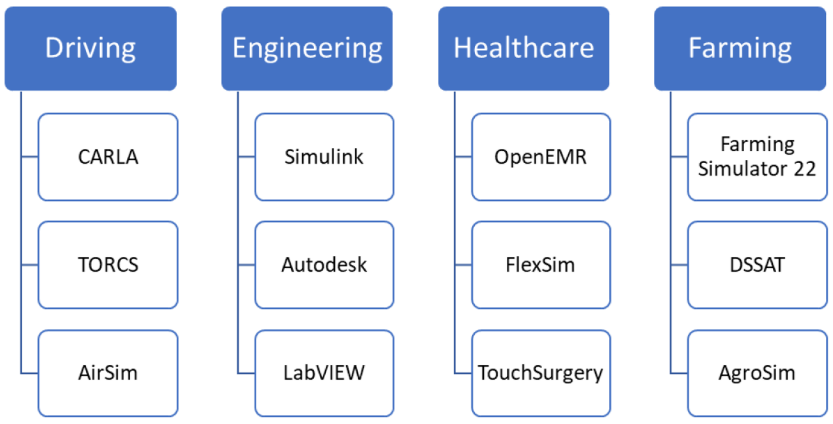


Figure 2.3: Examples of simulators utilized in driving, engineering, healthcare, and farming.

physics, lighting, and sound. They are usually used in fields such as gaming, animation, virtual reality, augmented reality, and the metaverse. Unlike simulators, game engines can be used to create virtual worlds that may or may not be designed to mimic the real world. Game engines are mainly used to develop games. However, they can be utilized for training and simulation, films and television, and visualization. In Figure 2.4, you can see some examples of modern rendering and game engines.

2.4.1.3 Generating synthetic data

To generate synthetic data with its corresponding ground truth, it is usually recommended to follow these steps:

- Identify the task and ground truth to generate;
- Create the 3D virtual world in the game engine;
- Set the virtual camera;
- Add noise and anomalies;
- Set the labeling pipeline;

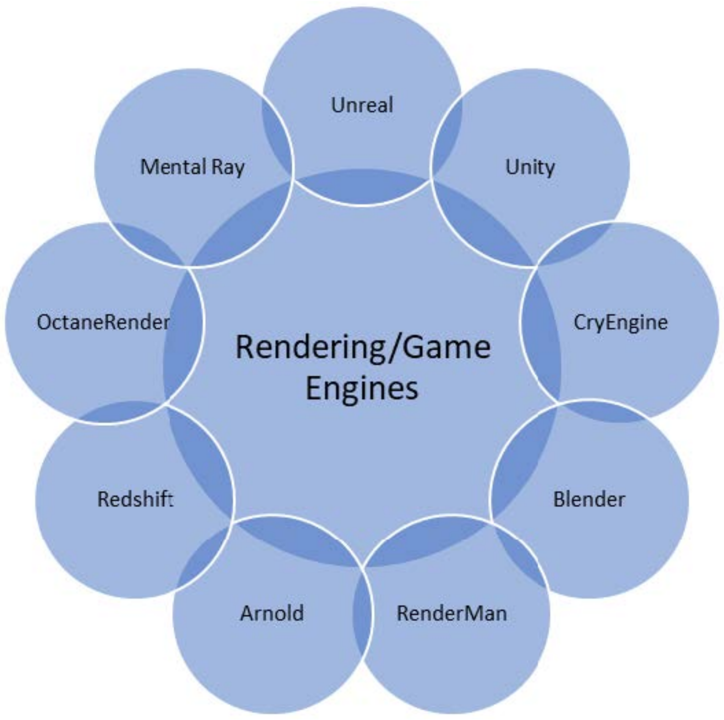


Figure 2.4: Examples of modern rendering and game engines.

- Generate the training data with the ground truth.

2.4.2 Exploring generative adversarial networks

2.4.2.1 Generative adversarial networks (GANs)

In this section, we will introduce GANs and briefly discuss the evolution and progression of this particular data generation method. Then, we will explain the standard architecture of a typical GAN and how they work. The concept of GANs was introduced in the 2014 paper Generative Adversarial Networks (Goodfellow et al., 2020), by Ian J. Goodfellow and his research team. In the same year, conditional GANs were introduced, allowing us to generate more customizable synthetic data. Then, Deep Convolutional GANs (DCGANs) were suggested in 2015, which facilitated the generation of high-resolution images.

After that, CycleGANs were proposed in 2017 for unsupervised image-to-image translation tasks. This opened the door for enormous applications such as domain

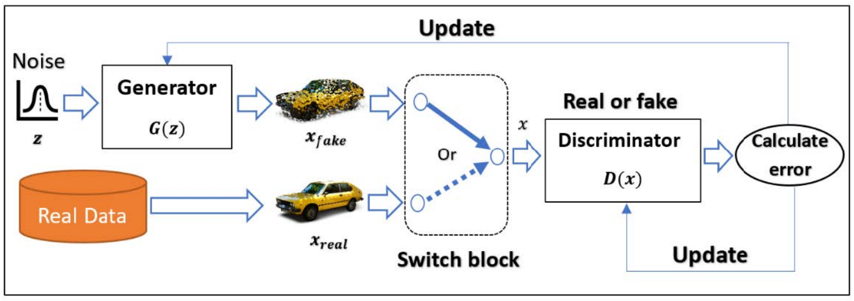


Figure 2.5: A typical architecture and training process of GANs.

adaptation. StyleGAN was introduced in 2019, bringing GANs to new fields such as art and fashion. GANs have also been showing impressive progress in the field of video synthesis. In fact, the recent work by NVIDIA is a testament to their tremendous potential (T.-C. Wang, Mallya, and M.-Y. Liu, 2021). This work shows that GANs can now recreate a talking-head video using only a single source image.

Next, we delve into the architecture of GANs. Most DL methods and architectures are designed to predict something. It could be weather conditions, stock prices, object classes, or something else. However, GANs were proposed to generate something. It could be images, videos, texts, music, or point clouds. At the heart of this capability lies the essential problem of learning how to generate training samples from a given domain or dataset. GANs are DL methods that can learn complex data distributions and can be leveraged to generate an unlimited number of samples that belong to a specific distribution. These generated synthetic samples have many applications for data augmentation, style transfer, and data privacy.

2.4.2.2 GAN training algorithm

The training algorithm is a crucial aspect of enabling GANs to generate useful synthetic data. The following is a step-by-step procedure that can be utilized to train GANs as shown in Figure 2.5:

- Create z by sampling a random noise following a suitable noise distribution such as uniform, Gaussian, Binomial, Poisson, Exponential, Gamma, and Weibull distributions.
- Feed z to the generator to produce a synthetic or fake sample, x fake.
- Pass both x fake and x real to a switch block, which randomly selects one of its inputs and passes it to the discriminator.
- The discriminator classifies the given sample as real or fake.
- Calculate the error.
- Backpropagate the error to both the generator and discriminator.
- Update the weights of the generator and discriminator

2.4.3 Video games as a source of synthetic data

Video games are interactive electronic games used primarily for entertainment. The player usually interacts with game elements to achieve an objective. The volume, quality, and quantity of games released each year have grown exponentially in recent years. Video games are now utilized in education, training, rehabilitation, personal development, and just recently in ML research. Specifically, they are presented as a rich and excellent synthetic data resource for training and testing ML models.

This synthetic data generation approach transfers the problem from creating virtual worlds to generate synthetic data to manipulating a video game to generate synthetic data instead. This method presents a convenient and efficient way to generate synthetic data. Examples of video games that have been leveraged for this purpose are listed as follows:

- **Grand Theft Auto V:** Playing for Data: Ground Truth from Computer Games (Richter, Vineet, et al., 2016).

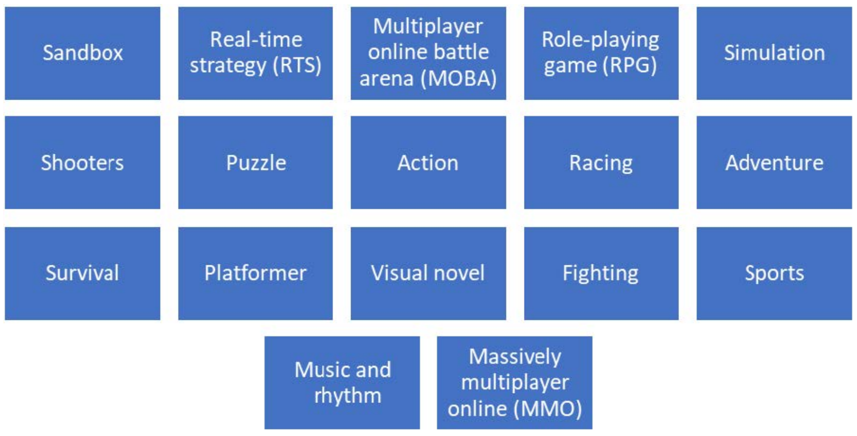


Figure 2.6: Examples of video game genres.

- **Minecraft:** Exploring the Impacts from Datasets to Monocular Depth Estimation (MDE) Models with MineNavi (X. Wang et al., 2020).
- **Half-Life 2:** OVVV: Using Virtual Worlds to Design and Evaluate Surveillance Systems (Taylor, Chosak, and Brewer, 2007).

The diversity in video game genres, as shown in Figure 2.6, makes generating various and rich synthetic data by utilizing video games a more attractive option for researchers. There are various ways to leverage video games in ML applications, such as the following:

- Utilizing games for general data collection;
- Utilizing games for social studies;
- Utilizing simulation games for data generation.

2.4.4 Exploring diffusion models for synthetic data

Diffusion Models (DMs) are generative models that were recently proposed as a clever solution to generate images, audio, videos, time series, and texts. DMs are excellent at modeling complex probability distributions, structures, temporal dependencies, and correlations in data. The initial mathematical model behind

DMs was first proposed and applied in the field of statistical mechanics to study the random motion of particles in gases and liquids. DMs are powerful generative models that can usually generate higher-quality and more privacy-preserving synthetic data compared to other approaches. Additionally, DMs rely on strong mathematical and theoretical foundations.

One of the first works to show that DMs can be utilized to generate photorealistic images was Denoising Diffusion Probabilistic Models (Ho, Jain, and Abbeel, 2020), which was proposed by researchers from UC Berkeley. This pioneering work was followed by another work by OpenAI titled Diffusion Models Beat GANs on Image Synthesis (Dhariwal and Nichol, 2021), showing that DMs are better at generating photorealistic synthetic images. Then, other researchers started to explore the potential of these DMs in different fields and compare them to VAEs and GANs. Variational Autoencoders (VAEs) are one of the earliest solutions for generating synthetic data. They are based on using an encoder to encode data from a high-dimensional space (such as RGB images) into a latent low-dimensional space. Then, the decoder is used to reconstruct these encoded samples from the latent space to the original high-dimensional space. In the training process, the VAE is forced to minimize the loss between the original training sample and the reconstructed one by the decoder.

Assuming the model was trained on a sufficient number of training samples, it can then be used to generate new synthetic data by sampling points from the latent space and using the decoder to decode them, from the latent low-dimensional space to the high-dimensional one, as shown in Figure 2.7.

Before we shift our focus to semantic segmentation, we define three essential concepts and terms used throughout this thesis: *Realism*, *Photorealism*, and *Diversity*.

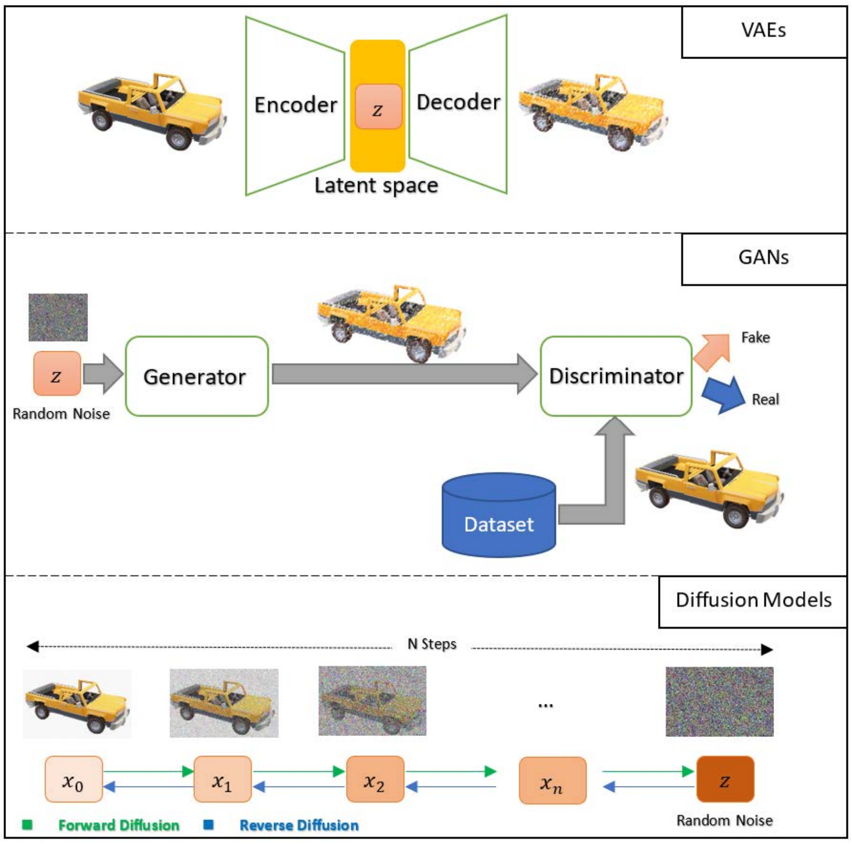


Figure 2.7: The training process and architectures of the main generative models – VAEs, GANs, and DMs.

2.5 Realism, Photorealism, and Diversity

In this section, *Realism*, *Photorealism,* and *Diversity* concepts are defined and explained.

2.5.1 Realism

According to the *Merriam-Webster* dictionary, *Realism* is defined as "concern for fact or reality and rejection of the impractical and visionary". Many researchers argue that making a realistic image or scene is an art, not a science (Ferwerda, 2003; Mackinlay et al., 1988). Therefore, it is extremely hard to come up with a closed-form mathematical expression or definition of realism. Realism, nevertheless,

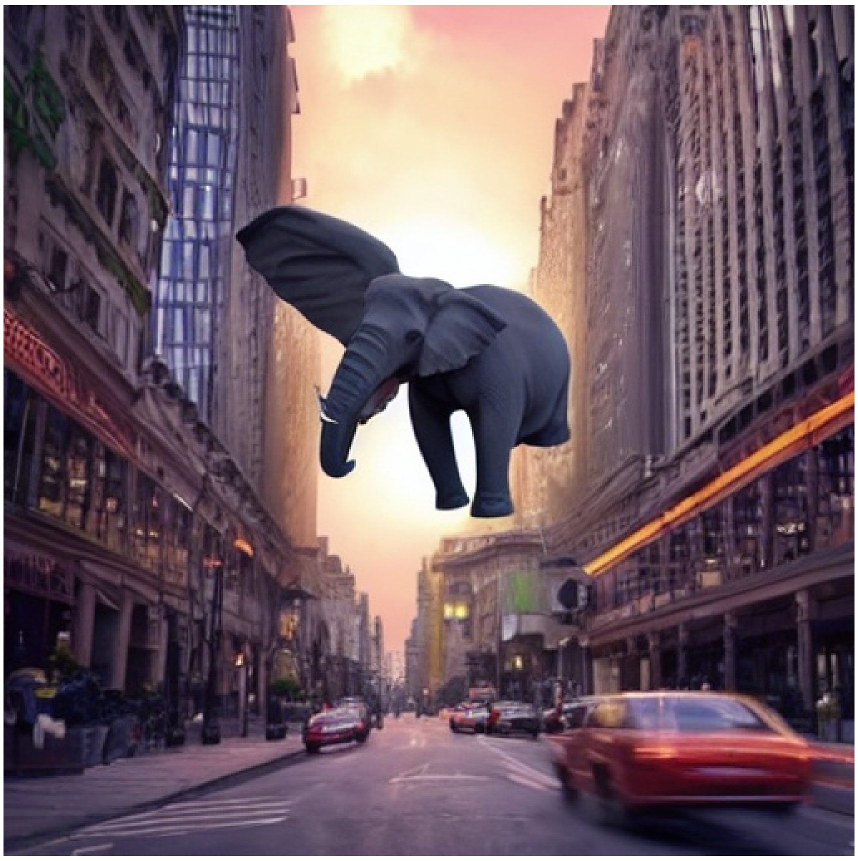


Figure 2.8: Photorealistic rendering of a nonrealistic scene, generated using DeepAI.

in the synthetic data field is essential to generate useful synthetic images and can be understood in the same way. When the virtual world is created, it is essential to focus on the level of realism in the scene. In other words, objects' sizes, proportions, locations, movement, colors, and other properties must imitate their counterpart in the real world. Otherwise, even if the synthetic images are photorealistic, they may not be necessarily useful. A descriptive image showing a nonrealistic scene rendered using a highly photorealistic approach is shown in Figure 2.9. .

2.5.2 Photorealism

Let us assume that we built a realistic virtual world. However, we use non-photorealistic rendering (NPR) or a highly stylized rendering pipeline to render the scene as shown in Figure 2.9. The resultant generated images will be non-photorealistic but capture a realistic scene. Therefore, they will not be useful, too. One of the early aims of computer graphics was to generate images that are indistinguishable from photographs (Rademacher et al., 2001). Photorealism involves not only capturing the visual appearance of objects and scenes but also simulating realistic light interactions with scene elements, medium, and environment (e.g., fog, rain, snow) between the light source and the camera, and surface textures to produce highly convincing visual results for the human eye.

Merriam-Webster dictionary defines photorealism as "the quality in art (such as animation or painting) of depicting or seeming to depict real people, objects, etc. with the exactness of a photograph". In computer graphics, photorealistic rendering techniques usually strive to replicate the optical characteristics of real-world cameras, lighting setups, and physical properties of objects materials. When generating photorealistic synthetic images for training or testing computer vision models, images that are perceptually similar to real photographs are considered more photorealistic. Training ML models on photorealistic images of realistic virtual scenes helps these models mitigate the synthetic-to-real or real-to-synthetic domain shift problems.

2.5.3 Diversity

The positive effect of *Diversity* on ML model generalization in the real world is well-known and widely studied in the field of ML (Gong, Zhong, and W. Hu, 2019; Hyontai, 2018; T. Chen et al., 2022). Data diversity refers to the degree of variability or dissimilarity present within a dataset. For example, it describes how much samples, features, or images, represent distinct characteristics or properties compared to other samples. Let us change our focus to the diversity of synthetic data for computer vision applications and tasks. Let us also link it with what we have discussed about *realism* and *photorealism*.

First, let us suppose we have a realistic virtual scene and we are using a state-of-the-art photorealistic rendering pipeline to render the scene. Is that sufficient to

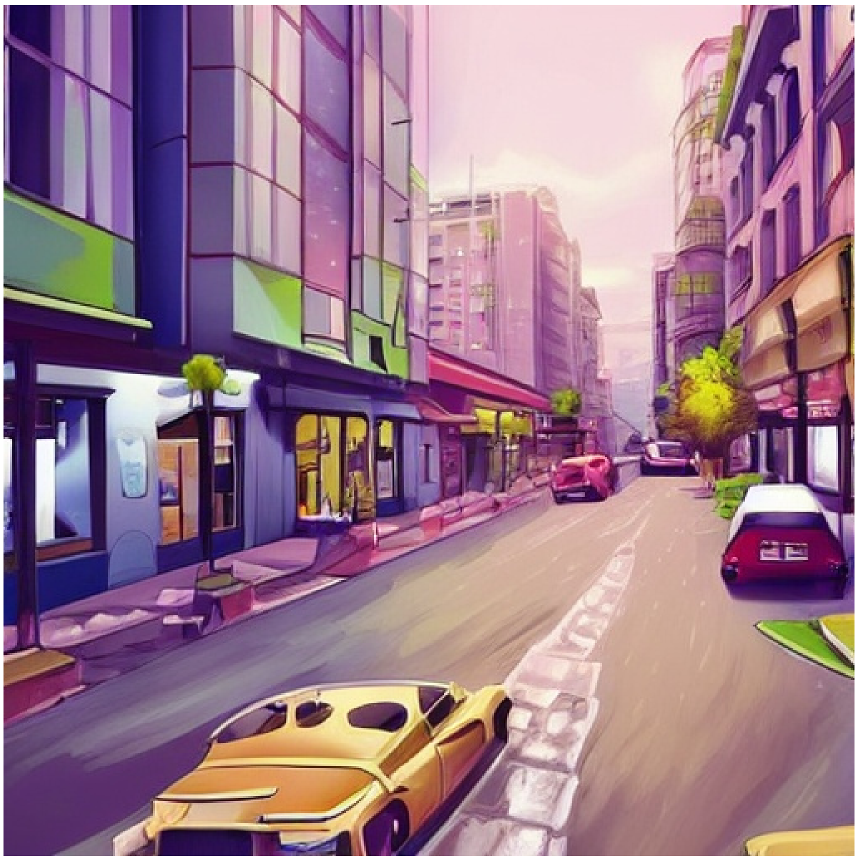


Figure 2.9: A non-photorealistic rendering of a realistic scene, generated using DeepAI.

generate useful synthetic images? The answer is no, not necessarily. We need also to ensure that our scene and generated data are diverse. For instance, using thirty 3D car models in your scene rather than only three 3D models is better in terms of diversity.

Diversity in this context refers to the variation present within a virtual scene, dataset or a set of generated images. Beyond intensity and surface texture variations, diversity encompasses a wide range of factors such as geometric variations (e.g., different shapes, sizes, orientations), object-level variations (e.g., various object categories, poses, configurations), and contextual variations (e.g., different backgrounds, lighting conditions, occlusions). In computer vision and ML, ensuring diversity in datasets is crucial for robust model training and generalization, as it exposes models to a wide spectrum of visual conditions and scenarios, helping them learn more robust and versatile representations.

In summary, for synthetic images to be useful for training and testing computer vision models, they need to be realistic, photorealistic, and diverse. Throughout this thesis and for simplicity, we mean by photorealistic images the images that are photorealistic and capture a realistic scene.

2.6 Semantic Segmentation

Semantic segmentation is a computer vision task that involves classifying and labeling each pixel in an image with a corresponding class label. Unlike image classification, where the goal is to assign a single label to an entire image, semantic segmentation aims to understand the content of an image at a pixel level.

In semantic segmentation, the objective is to partition an image into meaningful segments or regions and assign a specific semantic label to each segment. The semantic labels typically represent object categories, such as "person," "car," and "tree." The result is a detailed and pixel-level understanding of the scene, enabling a computer vision system to differentiate between various objects and their boundaries as shown in Figure 2.10.

Semantic segmentation has numerous applications, including autonomous driving, medical image analysis, robotics, scene understanding, and augmented reality. For example, in autonomous driving, semantic segmentation helps identify and classify objects on the road, such as pedestrians, vehicles, and traffic signs. Then, certain decisions can be taken by specific algorithms.

It should be noted that many state-of-the-art semantic segmentation methods leverage deep learning techniques, particularly convolutional neural networks (CNNs). These networks learn hierarchical features to capture spatial dependencies and patterns within images, making them well-suited for complex segmentation tasks. As we know, training and evaluating semantic segmentation models often

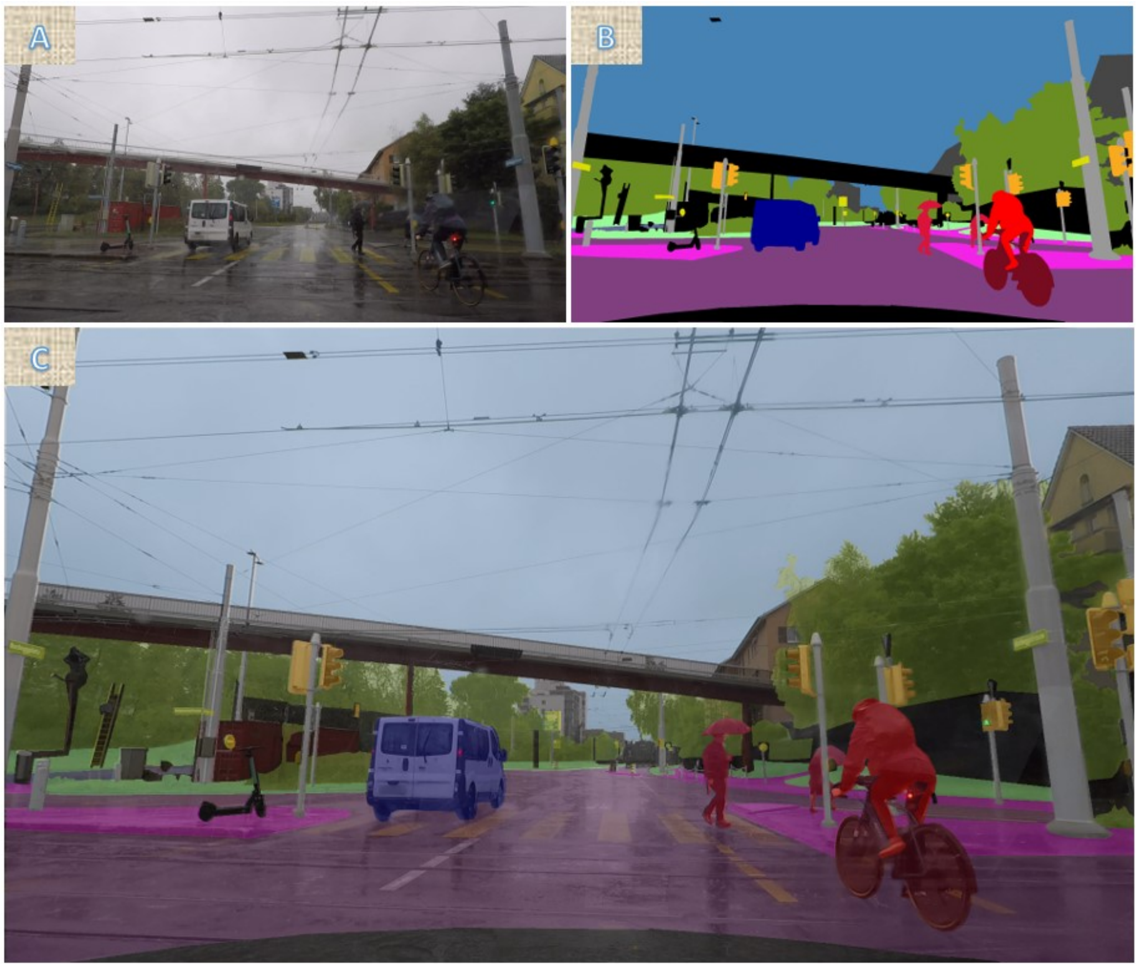


Figure 2.10: Visualizing of a training sample from the ACDC Dataset: A) RGB image, B) Semantic segmentation ground truth, and C) Overlay for clear understanding of pixel-level annotations.

require large annotated datasets with pixel-level labels.

Semantic segmentation plays a crucial role in advancing computer vision applications by providing a more detailed and context-aware understanding of visual data. It is an essential component in various real-world systems where precise scene understanding is required.

2.6.1 *DeepLabV3+* architecture

DeepLabV3+ is a state-of-the-art and widely known semantic segmentation architecture. It builds on *DeepLabV3* (L.-C. Chen, Papandreou, Schroff, et al., 2017) and it is especially powerful at achieving accurate semantic segmentation predictions along object boundaries.

DeepLabV3+ was developed by Liang-Chieh Chen et al. (Google Inc). This architecture builds upon its predecessor, *DeepLabV3*, by incorporating Atrous Spatial Pyramid Pooling (ASPP) module.

Atrous convolution. Atrous convolution is also called *dilated convolution.* It is commonly used in in convolutions for deep learning. For more details, please refer to L.-C. Chen, Papandreou, Kokkinos, et al. (2017).

Atrous convolution is a powerful technique that provides explicit control over the resolution of features computed by the convolution operation. It enables the adjustment of a filter's field-of-view, allowing the capture of multi-scale information. This operation extends the capabilities of the standard convolution operation as shown in this equation:

$$x_{\text{atrous}}(i) = \sum_{k=1}^{K} w(k) \cdot x(i + r \cdot k) \tag{2.1}$$

Here, $x_{\text{atrous}}(i)$ is the output at position i, $w(k)$ is the filter weight at position k, r is the rate (or dilation factor), and K is the kernel size. Changing r's value, permits adjusting the convolution filter's field-of-view.

Spatial pyramid pooling (SPP). One primary objective of SPP is to address the challenge of processing input images with varying sizes and aspect ratios and producing a fixed-length feature vector.

SPP partitions the input feature map into a predetermined number of spatial bins, effectively capturing information across various scales and resolutions. Each bin is associated with a specific receptive field size, enabling the model to incorporate both local and global context. Within each bin, pooling operations are independently applied to extract the required features. The outcome of these pooling operations in all bins is concatenated into a fixed-size vector. Please note that the size of the vector does not depend on the original image size. This methodology ensures that the network can effectively process input images of diverse dimensions while still capturing essential spatial information. One way to represent the process is as shown in this equation:

$$\text{SPP}(x) = \text{concat}(\text{pool}_1(x), \text{pool}_2(x), ..., \text{pool}_n(x)) \tag{2.2}$$

Here, $\text{pool}_i(x)$ represents the pooling operation at scale i, and concat denotes the concatenation of the pooled features. Therefore, by incorporating features from various scales, SPP enhances the model's robustness to objects of different sizes within the input images, contributing to improved performance in tasks that demand scale-invariant representations such as semantic segmentation. For more details, please refer to He et al. (2015).

ASPP enables the integration of multi-scale features, crucial for capturing contextual information and improving segmentation accuracy. As we can see from Figure 2.11, key innovation in *DeepLabV3+* lies in the use of a decoder module that refines the coarse segmentation results obtained from the ASPP and helps to recover objects boundaries information during the prediction process. This decoder, comprising upsampling and skip connections, facilitates the generation of high-resolution segmentation maps.

In the context of this thesis (Chapter 5), we use the *DeepLabV3+* architecture as a baseline to enhance the robustness of semantic segmentation models under adverse weather conditions using synthetic data. The architecture's ability to capture intricate contextual information and refine coarse predictions made it an ideal candidate for addressing challenges posed by diverse and complex environmental factors. By extending the *DeepLabV3+* encoder to be weather and nighttime aware, we aimed to improve the model's performance in adverse conditions, contributing

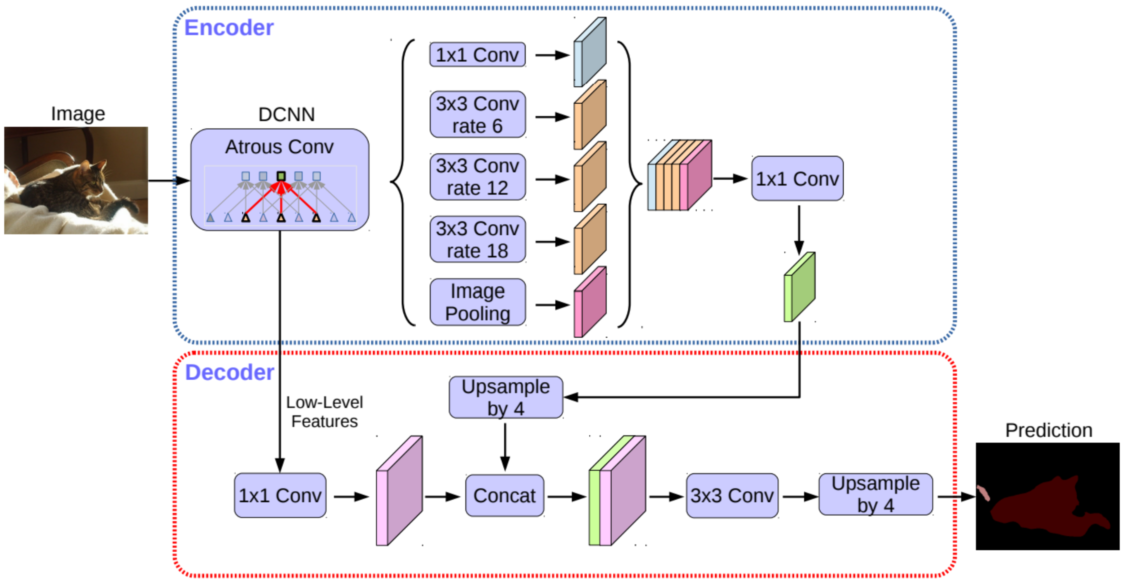


Figure 2.11: *DeepLabV3+* architecture (adapted from L.-C. Chen, Y. Zhu, et al. (2018)).

to the development of synthetic-aware computer vision approaches (see Chapter 5 for more details).

2.7 Video Stabilization

Video stabilization in computer vision refers to the process of reducing unwanted motion and jitter in a video sequence, resulting in a smoother and more visually appealing output. This technique is particularly useful when the original video footage is shaky or contains vibrations, which can be distracting for viewers or other computer vision algorithms. For example, surveillance camera footage may be affected by external factors such as wind or vibrations, leading to shaky video. In this case, video stabilization ensures that surveillance videos provide a stable and clear view, aiding in the accurate analysis and interpretation of the recorded events.

The main goal of video stabilization is to compensate for undesired camera movements, such as hand tremors or vibrations during recording, by applying computational methods to adjust the frames of the video. This can be achieved through various algorithms and techniques (see Chapter 3).

The overall process typically involves the following steps:

- Motion estimation;
- Transformation calculation;
- Frame alignment;
- Interpolation.

Next, we will discuss each of these steps in more detail.

2.7.1 Motion estimation

This step helps identify the undesired camera movements. It involves analyzing consecutive frames to determine the motion or displacement of objects in the video sequence. This information is then used to calculate the transformations needed to align the frames and reduce unwanted motion. These are some commonly used approaches for motion estimation:

- Block matching;
- Optical flow;
- Feature tracking;
- Phase correlation;
- Global motion models;
- Hybrid approaches.

Next, we will delve into each of them in more detail.

2.7.1.1 Block matching

One common approach for motion estimation is block matching. In this method, the current frame is divided into blocks, and the corresponding blocks in the next frame are searched to find the best match. The displacement between the blocks provides an estimate of the motion.

2.7.1.2 Optical flow

Optical flow is another widely used technique for motion estimation. It involves tracking the movement of pixels or features between consecutive frames. The apparent motion of pixels is represented as a vector field, indicating the direction and magnitude of motion at each point.

2.7.1.3 Feature tracking

Instead of considering all pixels, some algorithms focus on tracking specific features or keypoints in the video frames. These features could be corners, edges, or other distinctive points. Feature tracking algorithms, such as the Kanade-Lucas-Tomasi (KLT) tracker (Lucas and Kanade, 1981), follow the movement of these features over time.

2.7.1.4 Phase correlation

Phase correlation is a frequency-domain approach for motion estimation. It involves computing the cross-correlation of the Fourier transforms of two frames to find the phase shift, which corresponds to the motion between the frames.

2.7.1.5 Global motion models

Some video stabilization algorithms use global motion models to estimate the overall motion of the camera. These models may include translations, rotations, and scaling.

Kalman filters (Welch, Bishop, et al., 1995) or other predictive models can be employed to improve the accuracy of motion estimation.

2.7.1.6 Hybrid approaches

Many state-of-the-art video stabilization methods use hybrid approaches that combine multiple motion estimation techniques. For example, a system might use block matching for coarse motion estimation and then refine the results with optical flow or feature tracking for more accurate and detailed motion information.

Real-time video stabilization requires efficient motion estimation algorithms. Some methods use hierarchical approaches, where motion is estimated at different scales to balance accuracy and computational efficiency.

Motion estimation algorithms need to be robust to outliers and handle situations where some parts of the frame may be occluded or exhibit different motion. Robust techniques, such as RANSAC (Fischler and Bolles, 1981) and MAGSAC (Barath, Matas, and Noskova, 2019) or via learning approaches (Jirong Zhang et al., 2020; DeTone, Malisiewicz, and Rabinovich, 2016) which are often employed to deal with outliers and ensure accurate motion estimation. The output of the motion estimation step provides the necessary information to calculate the transformations needed to align frames in the video sequence. This, in turn, enables subsequent steps in the video stabilization pipeline, such as frame alignment and interpolation, to produce a stabilized video output.

2.7.2 Transformation calculation

In this step, we determine or calculate the transformation needed to align frames and minimize motion discrepancies. Let us particularly focus on affine and homography transformations.

2.7.2.1 Affine transformation

An affine transformation is a linear mapping method that preserves points, straight lines, and planes. It includes translation, rotation, scaling, and shearing operations.

The main elements of the affine transformation are the following:

- **Translation:** Moving an object in a straight line without rotation or deformation;
- **Rotation:** Changing the orientation of an object around a fixed point;
- **Scaling:** Changing the size of an object, either uniformly or along different axes;
- **Shearing:** Deforming an object by shifting points in one direction.

Affine transformations are commonly used in video stabilization to model the overall motion between consecutive frames. By estimating the coefficients of the affine transformation matrix, it becomes possible to represent the translation, rotation, and scaling needed to align frames.

Mathematical representation: The affine transformation A matrix for a 2D space is usually represented as:

$$\mathrm{A} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}.$$

Please note that a and e represent scaling factors, b and d represent shearing factors, and c and f represent translation values.

2.7.2.2 Homography transformation

Homography is a more general transformation that includes affine transformations but also allows for perspective distortions.

Unlike affine transformations, homography can represent the projective transformation that occurs when the camera viewpoint changes or when scenes involve depth.

Homography is suitable for cases where the camera undergoes significant changes in orientation, such as tilting or panning. It is often used to model the transformation between frames in scenarios where perspective changes are prominent.

Mathematical representation: In a 3×3 matrix, a homography transformation can be represented as:

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}.$$

The homography matrix H can be used to map points from one frame to another, accounting for both affine and perspective transformations.

The choice between affine and homography often depends on the characteristics of the video data and the specific requirements of the stabilization application. In video stabilization pipelines, the transformation calculation step involves estimating the coefficients of these transformation matrices, whether affine or homography, to align frames and minimize motion discrepancies for a smooth and stabilized output.

2.7.3 Frame alignment

Frame alignment is applying the calculated transformation to each frame to align them properly. This step ensures that the frames are adjusted to reduce motion artifacts.

2.7.4 Interpolation

Interpolation is filling in any gaps or missing information caused by the alignment process. Interpolation methods are often used to generate new frames that smooth out the video. Let us examine two key types of interpolation:

I) Linear Interpolation. Linear interpolation estimates values at intermediate points based on the straight-line segment connecting two neighboring known points. For a variable x between two known points (x_0, y_0) and (x_1, y_1), the linear interpolation formula is given by:

$$y = y_0 + \frac{(x - x_0) \cdot (y_1 - y_0)}{x_1 - x_0} \tag{2.3}$$

II) Bilinear Interpolation. Bilinear interpolation is an extension of linear interpolation for two dimensions. For a point (x, y) within a grid formed by four known points (x_0, y_0), (x_1, y_0), (x_0, y_1), and (x_1, y_1), the bilinear interpolation formula is given by:

$$f(x,y) = (1-\alpha)(1-\beta)f(x_0,y_0)+\alpha(1-\beta)f(x_1,y_0)+(1-\alpha)\beta f(x_0,y_1)+\alpha\beta f(x_1,y_1) \tag{2.4}$$

where $\alpha = \frac{x-x_0}{x_1-x_0}$ and $\beta = \frac{y-y_0}{y_1-y_0}$.

Video stabilization is widely used in various applications, including video editing, surveillance systems, and virtual and augmented reality, where a steady video stream is essential for providing a better viewing experience. Additionally, gyroscopes or accelerometers embedded in some recording devices can provide input to aid in stabilization.

2.8 Generative AI and Synthetic Data Usability

Generative AI is a category of AI systems that can generate new and original content. Generative AI is currently at the forefront of AI advancements and shows great promise for the future. These generative AI systems are designed to learn patterns and structures from existing data and then use that knowledge to create new, similar data. One of the most common approaches to generative AI is the use of generative models, such as GANs (Creswell et al., 2018; Goodfellow et al., 2020; Chakraborty et al., 2024) and VAEs (Wu, Cao, and Qi, 2024; Vahdat and Kautz, 2020).

2.8.1 Traditional data quality metrics

Structural Similarity Index (SSIM). The Structural Similarity Index (SSIM) is a metric used to assess perceptual similarity between two images. It is computed using the following formula:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2.5}$$

Here are the components of the SSIM formula:

- x and y are the input images being compared.
- μ_x and μ_y are the average luminance values of x and y, respectively. They can be calculated as shown in these two equations:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x(i) \tag{2.6}$$

$$\mu_y = \frac{1}{N}\sum_{i=1}^{N} y(i) \tag{2.7}$$

- σ_x^2 and σ_y^2 are the variances of x and y, respectively. They can calculated as shown in the following three equations:

$$\sigma_x^2 = \frac{1}{N}\sum_{i=1}^{N} (x(i) - \mu_x)^2 \tag{2.8}$$

$$\sigma_y^2 = \frac{1}{N}\sum_{i=1}^{N} (y(i) - \mu_y)^2 \tag{2.9}$$

$$\sigma_{xy} = \frac{1}{N}\sum_{i=1}^{N} (x(i) - \mu_x)(y(i) - \mu_y) \tag{2.10}$$

- σ_{xy} is the covariance of x and y. It can be calculate using this equation:

$$\sigma_{xy} = \frac{1}{N}\sum_{i=1}^{N} (x(i) - \mu_x)(y(i) - \mu_y) \tag{2.11}$$

- C_1 and C_2 are constants added to the formula to avoid instability near zero. They can be calculate as shown in the following two equations:

$$C_1 = (k_1 L)^2 \tag{2.12}$$

$$C_2 = (k_2 L)^2 \tag{2.13}$$

Where k_1 and k_2 are constants, and L is the dynamic range of pixel values (typically $L = 2^{\text{bit depth}} - 1$).

Peak Signal-to-Noise Ratio (PSNR). Peak Signal-to-Noise Ratio (PSNR) is a widely used metric in image processing and video compression to evaluate the quality of reconstructed images to their originals. It measures the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. PSNR is expressed in decibels (dB) and is calculated using the Mean Squared Error (MSE) between the original and the reconstructed signals.

The formula to compute PSNR is as follows:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right) \tag{2.14}$$

where:

- MAX is the maximum possible pixel value of the image (fluctuation). It is usually 255 for 8-bit images.

- MSE is the Mean Squared Error or cumulative squared error between the two images, defined as:

$$\text{MSE} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{2.15}$$

  where $I(i,j)$ and $K(i,j)$ are the intensity values of the original and reconstructed images respectively, and m and n are the dimensions of the images.

A higher PSNR value indicates a higher quality reconstruction, as it implies a lower level of noise or distortion in the reconstructed signal (image) compared to the original signal (image).

Inception Score (IS). The Inception Score (IS) is a widely-used metric for evaluating the quality and diversity of images generated by generative models, particularly in the context of generative adversarial networks (GANs). It is computed based on the softmax outputs of an Inception-v3 classifier, which is pre-trained on a large dataset such as ImageNet. Given a set of generated images $\{x_i\}_{i=1}^N$ produced by a generative model, the Inception Score is calculated as follows:

$$\mathrm{IS} = \exp\left(\mathbb{E}_x\left[D_{\mathrm{KL}}(P(y|x)||P(y))\right]\right) \tag{2.16}$$

where $P(y)$ is the marginal distribution of class labels over all images, and $P(y|x)$ is the conditional distribution of class labels for a given image x. The Kullback-Leibler divergence D_{KL} measures the dissimilarity between these distributions.

Fréchet Inception Distance (FID). The Fréchet Inception Distance (FID) is a widely used metric for evaluating the quality of generative models, particularly for GANs. It measures the similarity between the distribution of real images and generated images by computing the Fréchet distance between their feature representations obtained from a pre-trained deep neural network, typically an Inception network. The FID score is computed using the following:

$$\mathrm{FID} = ||\mu_r - \mu_g||_2^2 + \mathrm{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}) \tag{2.17}$$

where μ_r and μ_g are the mean feature representations of real and generated images, respectively, and Σ_r and Σ_g are their corresponding covariance matrices.

A lower FID score usually indicates a higher similarity between the distributions of real and generated images, suggesting a better performance of the generative model. Conversely, a higher FID score signifies greater dissimilarity, indicating potential issues in the generative model and synthetic data being generated.

2.8.2 Synthetic data usability

Synthetic data refers to artificially generated data rather than data collected from the real world. In computer vision, synthetic data is often created to supplement or replace real-world datasets. Assessing the quality of generated synthetic data is crucial to ensure that the synthetic dataset accurately represents the characteristics of the real-world data and is suitable for training robust ML models.

The primary goal of training a ML model is to enable it to generalize well to unseen, real-world data. Assessing the quality of generated images ensures that the synthetic data accurately captures the visual patterns and complexities present in the target domain. High-quality synthetic images contribute to improved model performance and robust generalization. Thus, the quality of generated images directly influences the learning representation acquired by the model during training. If synthetic data lacks fidelity to real-world scenarios, the model may learn irrelevant features or patterns that do not align with the actual data distribution. Assessing image quality helps in crafting synthetic datasets that facilitate the learning of meaningful representations, enhancing the model's understanding of the target domain.

Poorly generated synthetic images may introduce biases or unintended artifacts into the training process. If the generated data deviates significantly from real-world distributions, the model may be biased or produce unexpected results when deployed. Quality assessment, or synthetic data usability assessment, allows for the identification and rectification of issues, ensuring that synthetic data accurately represents the desired characteristics of the target domain.

Additionally, synthetic data usability assessment helps in optimizing the generation process and resource utilization. It allows researchers to identify areas where improvements can be made, ensuring that computational resources are allocated efficiently. By focusing on generating high-quality images, researchers can achieve better results with fewer resources, reducing the computational burden of training models.

In conclusion, assessing the quality of generated images in synthetic data is a fundamental step in the process of creating effective training datasets for ML models. It contributes to model performance, generalization, and the overall success of applications relying on synthetic data. Usability assessment ensures that synthetic data faithfully represents the target domain, yielding reliable and accurate results in real-world scenarios.

Chapter 3

Related Work

Science is a collaborative enterprise, spanning the generations. When it permits us to see the far side of some new horizon, we remember those who prepared the way – seeing for them also.

Carl Sagan

In this chapter, we include a necessary review of related work for this thesis. We start with a review of synthetic data generation methods, tools, and procedures in Section 3.1. Then, Section 3.2 will extend our review to semantic segmentation methods. Following this, we review well-known video stabilization approaches in Section 3.3. Finally, in Section 3.5, We provide a concise overview of synthetic data usability metrics, encapsulating the essential criteria for evaluating the efficacy of synthetic datasets in the realm of ML.

3.1 Synthetic Data Generation

With the substantial advancements in deep learning-based approaches, we have witnessed unprecedented progress in computer vision. This progress is attributed to the large-scale benchmark datasets that were collected in the past few years (Deng et al., 2009; Lin et al., 2014; Krizhevsky, G. Hinton, et al., 2009; Ge et al., 2019). Although the exponential increase in the amount of digital data today makes data collection easier than before, manual labelling of large volumes of examples with high quality and accurate labels still requires too much effort and comes with a tremendous cost.

The utilization of synthetic data in the computer vision field has just started recently. Its increase in popularity started to attract many researchers to propose novel algorithms to generate such datasets with their corresponding ground-truths. Although special care needs to be taken to weigh each method's advantages and disadvantages, they seem a promising solution to overcome the lack of suitable data for training supervised learning models. In this section, we summarize the mainstream of synthetic data generation.

3.1.1 Video Games for Synthetic Data Generation

Adapting a specific video game to generate synthetic data with its corresponding ground-truth for the task of semantic segmentation was shown by Richter, Vineet, et al. (2016), where the game Grand Theft Auto V was modified for that purpose. At the same time, another work by Shafaei, James J Little, and Schmidt (2016a) investigated utilizing photo-realistic video games to generate synthetic data and their corresponding ground truths for image segmentation and depth estimation. Using open source animation movies was another method discussed by Butler et al. (2012) where they were able to obtain an optical flow large-scale dataset, MPI-Sintel, following a systematic and easy process.

Unfortunately, the previous methods present a partial solution for the data generation issue because of the lack of control on the environment elements. At the same time, integrating new elements or behaviours to the scene like including a new 3D model, material or texture is extremely hard or simply unattainable. Another issue is the limitations of the proposed systems to specific computer vision tasks. Although these approaches are based on high quality and rich 3D virtual worlds, the failure of such methods to randomize the scene elements will lead to some clear repetitions (to scene elements) when a large-scale dataset is required to be generated by these methods. Procedural synthetic data generation offers an alternative to the previous solution.

3.1.2 Video Games and PCG

Procedural Content Generation (PCG) has been proposed as a solution for creating realistic-looking environments in relatively short amounts of time, making it easier and cheaper for users to generate virtual worlds from scratch. In its simplest form, a procedural generation framework follows systematic recipes to generate scenes, populations, and actions, based on the given set of instructions. *Silver*, our synthetic data generator illustrated in Chapter 4, is based on this concept.

Under this category, Roberto de Souza et al. (2017) investigated the possibility of adapting the PCG concept with Ragdoll physics, random perturbations and muscle weakening to generate a wide range of human actions systematically with their corresponding labels. Another work by Cheung et al. (2016) applied the concept of procedural generation to generate labelled crowd videos. Alternatively, Wrenninge and Unger (2018) presented a photorealistic and diverse synthetic dataset that can be generated entirely procedurally. The ability to parameterize the scene generation process and the fact that these parameters are not correlated are the main contributions of Wrenninge and Unger (2018). While procedural concept improves diversity, it still may limits the scalability of the system and it does not guarantee the photo-realism of the virtual world being generated by it. Therefore, an extra effort was taken with *Silver* to address these limitations.

An ideal solution to the problem is one that ensures high realism, diversity, scalability, controllability, and most importantly the generalizability of the approach to all computer vision tasks.

Table 3.1: **Comparison among synthetic semantic segmentation datasets.** Our dataset, named AWSS, is composed of photo-realistic pixel-wise annotated images under standard and adverse conditions.

	Weather Conditions				Times-of-Day		Photo-realism	Public Availability
	Normal	Rain	Fog	Snow	Daytime	Nighttime	/	/
GTA-V (Richter, Vineet, et al., 2016)	✔	✔	-	-	✔	-	✔	✔
Synscapes (Tsirikoglou et al., 2017; Wrenninge and Unger, 2018)	✔	-	-	-	✔	-	✔	✔
Virtual KITTI (Gaidon et al., 2016)	✔	✔	✔	-	✔	-	-	✔
Synthia (Ros et al., 2016)	✔	✔	-	✔	✔	✔	-	-
SHIFT (T. Sun et al., 2022)	✔	✔	✔	-	✔	✔	✔	✔
AWSS (Ours)	✔	✔	✔	✔	✔	✔	✔	✔

3.2 Semantic Segmentation

Our work utilizes synthetic data for domain adaptation to improve performance under adverse conditions (see Chapter 5). Thus, we review the usage of synthetic data in the semantic segmentation field. At the same time, we highlight the recent progress in domain adaptation methods similar to our work presented in Chapter 5.

3.2.1 Synthetic data for semantic segmentation

The high performance of recent semantic segmentation models is associated with the ability to train deep models on large-scale training data. The early real semantic segmentation datasets like CamVid (Brostow, Fauqueur, and Cipolla, 2009), Stanford Background (B. Liu, Gould, and Koller, 2010; Saxena, Chung, and A. Ng, 2005; Saxena, M. Sun, and A. Y. Ng, 2008), and KITTI-Layout (Alvarez et al., 2012) are limited in terms of the number of training samples, classes, resolution, and diversity. The problem is partially alleviated with the recent availability of datasets like Cityscapes (Cordts et al., 2016), ACDC (Sakaridis, D. Dai, and Van Gool, 2021), ADE20K (B. Zhou, Hang Zhao, Puig, Fidler, et al., 2017), and Mapillary Vistas (Neuhold et al., 2017). Nevertheless, annotating large-scale datasets of high-resolution images is still the bottleneck. At the same time, ensuring diverse training data under challenging attributes like adverse weather conditions is not only dangerous, time-consuming, and hard to collect but also cumbersome and subjective to human errors in the annotation process.

Synthetic data comes as a resort to handle all the above issues. Their success in computer vision is specifically seen in semantic segmentation. Goyal et al. (2017) demonstrate that augmenting synthetic data with weakly annotated data can improve the performance on the PASCAL VOC dataset (Everingham et al., 2015). Similarly, Richter, Vineet, et al. (2016) generate synthetic training data by utilizing the Grand Theft Auto V game. They show that training semantic segmentation models on one third of the training split of CamVid (Brostow, Fauqueur, and Cipolla, 2009) dataset along with their generated synthetic data achieves superior results compared to training on the full CamVid (Brostow, Fauqueur, and Cipolla, 2009). In parallel, Ivanovs et al. (2022) augment the Cityscapes (Cordts et al., 2016) dataset with synthetic images generated using the CARLA (Dosovitskiy, Ros, et al., 2017) simulator. They show that the performance improves when compared to training only on Cityscapes (Cordts et al., 2016). Similar to these works, we use synthetic data to boost the performance of semantic segmentation models. However, we tackle the domain shift problem using synthetic data and a synthetic-aware training procedure.

3.3 Video Stabilization

A major contribution of our work presented in this thesis, Chapter 6, is utilizing synthetic data for the task of video stabilization. Thus, video stabilization and synthetic data generation literature are briefly reviewed in this section.

Video stabilization methods are usually categorized into non-learning-based and learning-based.

3.3.1 Non-learning-based video stabilization

Non-learning-based video stabilization methods do not perform training. For instance, Grundmann, Kwatra, and Essa (2011) stabilize the shaky camera trajectory using L1-norm optimization under constraints, and Bradley et al. (2021) address the stabilization task as a constrained convex optimization problem. Although these methods do not require training data for tuning the model's parameters, they work only under predefined conditions, their parameters must be tuned manually, and their results tend to be less pleasant.

3.3.2 Learning-based video stabilization

The learning-based approaches are classified into unsupervised and supervised approaches. Non-supervised methods require training videos but do not demand shaky-stable video pairs. DIFRINT (Choi and Kweon, 2020), for example, is trained end-to-end and utilizes frame interpolation to synthesize middle frames for stabilization. On the other hand, supervised learning approaches require labeled data, which is the main limitation of applying them to video stabilization. StabNet (M. Wang et al., 2018) uses a mechanical stabilizer to generate ground-truth stable videos to train CNNs for video stabilization. The network learns a warping transformation of multi-grids given the shaky and previously stabilized frames. Y.-L. Liu et al. (2021) apply a learning-based hybrid-space fusion to compensate for optical flow inaccuracy. J. Yu and Ramamoorthi (2020) stabilize videos by computing the per-pixel warp field from the shaky video optical flow, allowing it to handle better moving objects and occlusion.

The previous methods present a partial solution to the video stabilization problem since they are assumed to work under normal weather conditions and sufficient illumination. However, finding resilient features in adverse conditions is rather challenging. For example, rain particles, foggy weather conditions, and low illumination pose clear challenges to finding robust features. Thus, it leads to inaccurate motion estimation and low-quality video stabilization. Our proposed video stabilization method (see Chapter 6) belongs to the supervised learning-based category. However, unlike other methods, we use only synthetic data for training. No pre-training or fine-tuning on real data is required by our method, and by using only a small-scale training dataset, it is more robust than state-of-the-art methods.

3.3.3 Affine and homography transformation

Estimating affine and homography transformations between two images is common to aligning one image with another. There are different ways to find these matrices, like applying a feature extractor (e.g., SIFT (Lowe, 2004) and OAN (Jiahui Zhang et al., 2019)) and an outlier rejection algorithm (e.g., RANSAC (Fischler and Bolles, 1981) and MAGSAC (Barath, Matas, and Noskova, 2019)) or via learning approaches (Jirong Zhang et al., 2020; DeTone, Malisiewicz, and Rabinovich, 2016).

Although the traditional feature extraction approach does well under standard conditions, it performs poorly under challenging scenarios. Supervised approaches cannot reflect scene parallax (Ye et al., 2021), and generating suitable training data is hard. Unsupervised approaches may solve these problems, but they fail under large baseline alignment, which makes them impractical for video stabilization under sharp camera movements.

Unlike previous methods, our model learns the affine transformation in a supervised manner using synthetic training data alone. Although it is possible to decompose the homography matrix to extract camera translation, rotation, and scale, it is inaccurate. Moreover, training a model to estimate the affine transformation is easier than estimating the homography. Although homography can better model the camera motion between frames for a limited number of frames, it starts to cause many artifacts like skew and perspective as the number of frames becomes larger (Grundmann, Kwatra, and Essa, 2011; S. Li et al., 2015). Additionally, it overfits even with some regularization. Thus, utilizing homography transformation is harder to train (more parameters and easier to overfit) and more subject to artifacts.

Synthetic data generation. Synthetic data is typically used to overcome training data scarcity for supervised learning models (Butler et al., 2012; Richter, Vineet, et al., 2016; Shafaei, James J Little, and Schmidt, 2016a; Yilin Liu, Xue, and H. Huang, 2021; Kerim, Celikcan, et al., 2021; Tsirikoglou, 2022). Richter, Vineet, et al. (2016) modified the GTA-V game to generate synthetic data and the corresponding ground truth for semantic segmentation. Shafaei, James J Little, and Schmidt (2016a) used photo-realistic games to generate data for image segmentation and depth estimation. Dosovitskiy, Ros, et al. (2017) presented CARLA, an autonomous driving simulator that provides ground-truth data for semantic segmentation and depth estimation tasks. Recently, the UrbanScene3D simulator was proposed by Yilin Liu, Xue, and H. Huang (2021) for autonomous driving and flying support.

While Sim2RealVS (Rao et al., 2023) shares some similarities with our work. It deploys the GTA-V game, which was not originally developed for the synthetic data generation task. In contrast, our simulator was developed for that reason. Sim2RealVS is limited to the GTA-V game assets (textures, objects, characters, and camera parameters). In comparison, our simulator provides more diversity and controllability. For each synthetic video a new city is procedurally created. Additionally, modifying or adding new scene elements can be easily done with our simulator, unlike Sim2RealVS approach using GTA-V.

These previous methods partially solve the data generation issue because of the lack of control over the generation process. They fail to randomize scene elements, leading to less diverse datasets. Our synthetic data generator utilizes procedural content generation to create 3D virtual worlds and generates special training data for video stabilization. In our experiments shown in chaper 6, we will show that generating appropriate training data and creating a synthetic data pipeline can achieve superior results. Our algorithm can still provide better results in real videos even though our synthetic data generator does not generate state-of-the-art photo-realistic videos. Additionally, our synthetic-aware algorithm and specially designed synthetic data teach the model to accurately estimate the affine transformation while not being overfitted to the synthetic data distribution. Thus, it mitigates the domain gap and achieves satisfactory results on real data.

3.4 Domain Adaptation (DA)

ML models trained on one domain or dataset may not necessarily generalise well to a new domain or dataset. This is usually called domain shift (Singhal et al., 2023; K. Zhou et al., 2022). Domain shift is a major limitation of synthetic data in practice. Models trained on synthetic data do not perform well on real-world data (Sankaranarayanan et al., 2018; Y. Xu et al., 2019; Dundar et al., 2020). This is because of many factors including for example distribution shift between the source and target data. It is possible, for example, that the synthetic images are only captured using perfect sensors. However, the real data is captured using faulty or imprecise sensors. Additionally, it is possible that synthetic data overfits a set of weather conditions, time-of-days, scenarios, and textures. Therefore, ML models, trained on such synthetic data, learn how to react appropriately only to such scenarios during the training process. However, when the model is tested on real data which is noisy, diverse, and sometimes unpredictable causes the ML model to fail drastically.

In the field of DA, several methodologies have been explored to address the challenges. These include employing batch normalization, as demonstrated in studies such as Y. Li et al. (2018) and Chang et al. (2019). Additionally, domain randomization has shown effectiveness, as evidenced by research conducted by Yue et al. (2019) and Volpi, Larlus, and Rogez (2021). Moreover, techniques involving learning disentangled representations and generative modeling have been applied, as seen in works by Ilse et al. (2020), D. Li et al. (2017), and G. Wang et al. (2020).

For instance, Sankaranarayanan et al. (2018) propose a Generative Adversarial Network (GAN) based approach that minimizes the distance between the encodings of both domains. They show that their approach can boost the performance of synthetic-to-real domain adaptation tasks. Our work presented in Chapter 5 is similar to theirs as we use synthetic data for domain adaptation and propose a synthetic-aware training procedure. However, our work tackles this problem under harder set-ups utilizing synthetic data to mitigate standard-to-adverse domain shifts.

In the same context, Alshammari, Akcay, and Breckon (2020) address standard to foggy weather domain shift by using an adversarial training strategy that guides the model to produce outputs close to the target domain. Similarly, X. Ma et al. (2022) tackle standard weather to foggy weather domain adaptation using both fog and style variations by adopting a Cumulative style-fog-dual disentanglement Domain Adaptation method (CuDA-Net). Alternatively, Q. Xu et al. (2021) address the daytime to nighttime domain shift. They utilize a novel Curriculum Domain Adaptation method (CDAda) that uses labeled synthetic nighttime images.

Our methods presented in Chapters 5 and 6 are closely related to these works. However, we tackle domain adaptation from a standard domain (*i.e.,* daytime and normal weather condition) to an adverse domain (*i.e.*, nighttime and adverse weather conditions such as rain, fog, and snow).

3.5 Synthetic Data Usability Metric

In this section, we summarize the key synthetic data generation methods and the main usability metrics.

3.5.1 Synthetic data generation methods

Synthetic data generation methods have been showing great progress in the computer vision field. It seems as a promising solution to overcome the lack of suitable labeled data for training deep-supervised learning models (Butler et al., 2012; Richter, Vineet, et al., 2016; Shafaei, James J. Little, and Schmidt, 2016b; Yilin Liu, Xue, and H. Huang, 2021; Tsirikoglou, 2022). Several synthetic data generation methods have been developed. Notable contributions in this area include the following methods:

- Game engines;
- Generative adversarial networks (GANs);
- Diffusion models (DMs).

Game engines. Using video games (Shafaei, James J. Little, and Schmidt, 2016b; Kiefer, Ott, and Zell, 2022) and game engines (Kerim, Aslan, et al., 2021; H. Lee et al., 2023; Q. Wang et al., 2021) were shown to be effective ways to generate automatically-labeled large-scale synthetic data for a wide range of computer vision tasks, such as semantic segmentation and depth estimation (Shafaei, James J. Little, and Schmidt, 2016b), video stabilization (Rao et al., 2023), and person re-identification (X. Sun and Zheng, 2019). These approaches are limited in diversity and photorealism to game engines used, game genre, environment, and artists' skills. Additionally, creating large-scale datasets can be computationally expensive and poses intellectual property issues since video games are not usually developed for this aim.

Generative adversarial networks (GANs). GANs have been widely utilized to generate synthetic training data for computer vision (H. Ali, Grönlund, and Z. Shah, 2023; Torfi, Fox, and Reddy, 2022). It was shown in Al Khalil et al. (2023) that utilizing synthetic images generated by their segmentation-informed conditional GAN improves the performance and robustness of heart cavity segmentation from short-axis cardiac MR images. Similarly, it was shown in L. Li et al. (2023) that utilizing synthetic images generated by their GAN and Neural Radiance Field (NeRF)-based framework improves the performance of 3D object detectors. One of the primary challenges with these methods is their stability issues during training, frequently leading to issues such as mode collapse (Kodali et al., 2017).

Diffusion models (DMs). Stable Diffusion, as demonstrated in Shipard et al. (2023), was used to boost the diversity of the generated images used for training a zero-shot classifier. Our work is similar to that work in using DMs to generate the training images. However, our approach tackles the diversity and photorealism of the generated images. Furthermore, we incorporate a Large Language Model (LLM) to create attributes utilized in DMs' prompts

Training on synthetic images generated by DMs for the task of skin disease classification was shown in Akrout et al. (2023) to achieve similar accuracy to training on real data. It was also shown that complementing ImageNet training dataset with synthetic images generated by DMs does improve ImageNet classification accuracy substantially.

Similar to that work in Akrout et al. (2023), we also explore the effectiveness of training on synthetic data. However, instead of limiting our study to a single architecture, we consider six diverse and well-known architectures.

Thus, the integration of LLMs with DMs models offers a holistic solution, addressing both semantic and visual aspects of synthetic image generation. This makes it a robust choice compared to other methods, particularly when aiming for diverse, realistic, and semantically aligned synthetic datasets for training computer vision models.

3.5.2 Synthetic data usability

There are many metrics in the literature deployed to assess the quality of generated synthetic images, such as the Fréchet Inception Distance (FID) (Heusel et al., 2017), Inception Score (IS) (Salimans et al., 2016), Peak-Signal-to-Noise Ratio (PSNR), and the Structural Similarity Index Measure (SSIM) (Z. Wang et al., 2004). However, there are very few studies to assess the usability of a generated synthetic image.

FID and IS. FID score is usually utilized to assess the quality of the generated images by GANs (Lucic et al., 2018; Borji, 2019). It calculates the Wasserstein distance between multivariate Gaussians embedded into a feature space of a specific layer of the Inception-V3 model (Szegedy et al., 2016) pretrained on ImageNet. On the other hand, IS calculates the Kullback-Leibler (KL) divergence between conditional and marginal class distributions utilizing the Inception-V3 model as well.

As shown in Barratt and Sharma (2018) and Ravuri and Vinyals (2019), FID and IS entangle diversity and photorealism (being close to the real distribution). Thus, they may not be ideal metrics to assess the usability of synthetic images. Additionally, they are not always interpretable in terms of photorealism and diversity.

SSIM and PSNR. SSIM can be utilized to measure the perceptual similarity between two images based on the degradation of structural information. On the other hand, PSNR is the ratio between the original (real) image peak of power and the estimated power of the noise from synthetically generated or reconstructed image.

Many studies have shown that SSIM and PSNR have many limitations (Pambrun and Noumeir, 2015; Huynh-Thu and Ghanbari, 2008). Both metrics focus on pixel-level differences and are sensitive to simple geometric transformations. Additionally, they do not consider perceptual quality and diversity.

Our proposed metric addresses these limitations. Our metric disentangles photorealism from diversity. At the same time, it represents a novel metric for assessing the visual quality and usability of the generated synthetic images.

Global consistency and complexity. In Scholz et al. (2023), two metrics are proposed to measure the global consistency of medical synthetic images (biological plausibility). They show that their approach is more robust compared to FID as it can explicitly measure global consistency on a per-image basis. Unlike that work, our approach is a general-purpose metric, not limited to certain fields. At the same time, it evaluates the diversity of the generated images. Similar to our work, Mahon and Lukasiewicz (2022) propose a metric for measuring image complexity using hierarchical clustering. However, the authors make many assumptions about the cluster probability distribution. Additionally, it requires many hyperparameter tuning.

Unlike these metrics, the metric proposed in this work does not require hyperparameter tuning and presents a more generic and practical metric. Additionally, our approach focuses on the usability of synthetic images which is not directly or necessarily dependent on the complexity or visual plausibility.

Chapter 4

A Synthetic Data Generator for Computer Vision Models

Imagination is the beginning of creation. You imagine what you desire; you will what you imagine; and at last, you create what you will.

George Bernard Shaw

The work presented in this chapter appeared as a workshop paper in Kerim, Soriano Marcolino, and R. Jiang (2021).

Large-scale synthetic data is needed to support the deep learning big-bang that started in the recent decade and influenced almost all scientific fields. Most of the synthetic data generation solutions are task-specific or unscalable while the others are expensive, based on commercial games, or unreliable. In this chapter, a new rendering engine called *Silver* is presented in detail. Photo-realism, diversity, scalability, and full 3D virtual world generation at run-time are the key aspects of this synthetic data generator. The photo-realism was approached by utilizing the state-of-the-art High Definition Render Pipeline (HDRP) of the Unity game engine. In parallel, the Procedural Content Generation (PCG) concept was employed to create a full 3D virtual world at run-time, while the scalability of the system was attained by taking advantage of the modular approach followed as we built the system from scratch. *Silver* can be used to provide clean, unbiased, and large-scale training and testing data for various computer vision tasks.

The source code of the complete synthetic data generator and a sample dataset are both available at `https://github.com/lsmcolab/Silver`. At the same time, a video displaying a sample 3D virtual world generated by *Silver* is provided at `https://youtu.be/Ktlx5bgJLXE`.

4.1 Introduction

When a problem can be described by a limited set of rules, computers perform very well and usually surpass human performance in terms of speed and accuracy. However, when the problem is too hard to be formulated, computers fail dramatically. Human reasoning and intelligence are still very far from being understood or formulated. The vast research done in this field still only scratches the surface (Colom et al., 2010; Lisman, 2015; Papo, 2015). Visual perception is one of the most complex tasks that the human brain performs accurately and efficiently. As visual perception is important for human daily activities, it is critical for the machine (agent) to perceive the environment, reason, plan, and then interact to achieve its goal. The machine could be as simple as an Optical Character Recognition (OCR) program or as complex as an autonomous car or robot on another planet. Failure of such agents could lead to deaths or injuries, damage to the environment or properties, failure of missions, and loss of millions of dollars.

The recent great success of ANN in solving complex problems motivated many researchers to apply it to machine perception problems too. In parallel to this, the great advancement in chip design, microelectronics, and the introduction of General-Purpose Graphics Processor Architectures like the General Purpose Graphics Processing Unit (GPGPU), facilitated training deep neural networks with millions of parameters. These deep ANNs with their high degree of non-linearity, help in approximating the real functions or phenomena behind complex vision problems (like semantic segmentation, instance segmentation, and object recognition) in a much more accurate way. Unfortunately, training these deep learning models requires a great amount of data together with their corresponding annotations or ground-truths. Finding, collecting, and annotating suitable data is cumbersome, time-consuming, error-prone, expensive, and subject to privacy issues. Perhaps, the lack of diverse, high quality, and precisely labeled data can be attributed to the previously mentioned reasons. Unfortunately, these factors cause many major data quality issues in the field of computer vision and they clearly present an obstruction toward the aim of optimal performance computer vision models in practice.

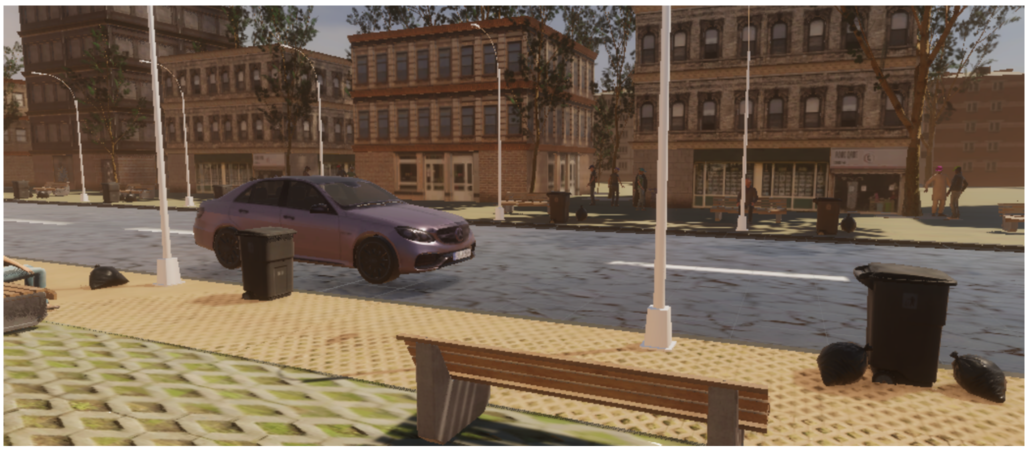


Figure 4.1: Sample scene generated using Silver.

The promising solution seems to be in the leading-edge game engines like Unity (2024), Unreal Engine (2024), and Cry Engine (2024). Leveraging the powerful tools of the Unity game engine, we present a system that creates three-dimensional, photo-realistic virtual worlds, procedurally at run-time. A sample scene generated by our synthetic data generator is shown in Figure 4.1. The system currently supports various computer vision tasks such as semantic segmentation, instance segmentation, depth estimation, and others. It allows researchers, with no computer graphics background, to generate large-scale synthetic datasets for training or testing their own computer vision models.

Our main contributions can be summarized as follows:

- To the best of our knowledge, *Silver* is a pioneering work that uses HDRP of the Unity game engine to build a 3D virtual world at run-time for synthetic data generation;
- Unlike other frameworks, *Silver* considers data generation as an online problem which opens the door for using synthetic data with online learning algorithms;
- This is the first work that considers generating a photo-realistic, full-city, procedurally and at run-time.

4.2 Synthetic Datasets and Common Limitations

The currently available computer vision synthetic datasets seem to cover a wide set of the main computer vision applications, such as:

- Visual object tracking;
- Semantic segmentation;
- Instance segmentation;
- Depth estimation;
- Action recognition;
- Optical flow.

Some of these datasets provide videos (Gaidon et al., 2016; Ros et al., 2016; Roberto de Souza et al., 2017; Butler et al., 2012) while others (Richter, Vineet, et al., 2016; Shafaei and James J Little, 2016; Hurl, Czarnecki, and Waslander, 2019) provide simple snapshots depending on the application or the task. The resolution of these synthetic datasets is restricted by the method used to generate it. Some of the methods allow only fixed resolutions (Richter, Hayder, and Koltun, 2017; Shafaei and James J Little, 2016) while others permit generating the dataset at various resolutions (Cheung et al., 2016; Butler et al., 2012).

Important characteristics of synthetic datasets are as follows:

- Diversity;
- Visual complexity;
- Realism.

4.2.1 Diversity

Synthetic datasets can be used for training and testing purposes. For training, diversity is needed to avoid over-fitting to the visual features of the synthetic world. On the other hand, for testing, diversity is required to represent a good proxy of the real world and the possible standard and rare scenarios. The real training and testing data are collected with the aim to be a good proxy of the actual world. Sometimes the space of the problem could be easy to describe and to collect suitable data that represents – approximately – this space. Unfortunately, predicting the problem space for most of the computer vision tasks is extremely hard or simply unfeasible. Thus, datasets with the highest diversity are always expected to improve the overall performance.

4.2.2 Visual complexity

The real world is full of details that are very hard to be captured even by the state-of-the-art simulators or rendering engines. However, generating synthetic data at a high-level of visual complexity is very important. Moreover, some specific details could be more critical for some computer vision tasks while not for some others. Thus, special care is needed in deciding the level of details the synthetic data should attain and which scene elements are more important than others.

4.2.3 Realism

The ultimate goal of models trained on synthetic data is to be tested on the real-world dataset. However, the most common problem that causes such models to fail or not to perform well is the domain gap (domain shift) problem. Many different approaches are suggested to mitigate this problem. However, improving the photo-realism of synthetic data is, nevertheless, the major approach. It should be noted that realism should not only be limited to photo-realism. However, the concept should be extended to animation, camera motion, object locations in the scene, object frequency, object scales, dynamic objects interactions with the static elements and other dynamic objects, to name a few.

The limitations of most of the observed synthetic datasets shown in Table 4.1 is the lack of adverse weather conditions and limited times of the day (hence, illumination). In addition to that, the camera behaviours are quite limited, unrealistic, or cinematic. These problems in fact simplify the actual issue, and cause computer vision models trained on these datasets to over-fit to these situations.

Table 4.1: **Synthetic datasets for computer vision tasks.** Synscapes (Tsirikoglou et al., 2017; Wrenninge and Unger, 2018), Virtual Kitti (Gaidon et al., 2016), Synthia (Ros et al., 2016), PHAV (Roberto de Souza et al., 2017), GTA-V (Richter, Vineet, et al., 2016), MPI-Sintel (Butler et al., 2012), SOMASet (Barbosa et al., 2018), LCrowdV (Cheung et al., 2016), VIPER (Richter, Hayder, and Koltun, 2017), UBC3V (Shafaei and James J Little, 2016), and PreSIL (Hurl, Czarnecki, and Waslander, 2019).

Name	CV Tasks	Size	Videos	Resolutions	Other Useful Info
Synscapes	Instance and semantic segmentation Depth Map, 2d and 3d bounding boxes Camera's position and field of view Occlusion and truncation Scene metadata	25K Frames	No	1440X720 2048X1024	Unbiased path tracing for rendering and Monte Carlo-based light transport simulation
Virtual Kitti	Object detection and multi-object tracking Scene-level and instance-level semantic segmentation Optical flow, and depth estimation	21,260 Frames	Yes	1242X375	Unity 5 environments
Synthia SYNTHIA-Rand & SYNTHIA-Seqs	Semantic segmentation Depth Map Car ego-motion	213K Frames 200K Frames	No Yes	960X720	Unity Virtual New York City 13 classes
PHAV	Action recognition Semantic segmentation Instance segmentation Depth map Vertical and horizontal optical flow	6M Frames	Yes	340X256	39,982 videos for more than 1,000 examples for each action of 35 categories
GTA-V	Semantic segmentation	25K Frames	No	1914X1052	Grand Theft Auto V Compatible with CamVid and CityScapes
MPI-Sintel	Optical flow	1628 Frames	Yes	1024X436 (Any)	Effect of resolution on Optical Flow Based on the open-source animated film Sintel
SOMASet	Person reidentification	100K Frames	No	400X200	Makehuman and Blender
LCrowdV	Bounding box, flow estimation, pedestrians count Crowd density, population, lighting conditions Background scene, camera angles Agent personality and noise level	20M Frames	Yes	Any	Unreal game engine Based on procedural modeling and rendering techniques
VIPER	Optical flow, semantic instance segmentation, Object detection and tracking Object-level 3d scene layout Visual odometry	254K Frames	Yes	1920X1080	Grand Theft Auto V Inject a middleware between the game and its supporting graphics library
UBC3V	Single or multiview depth-based pose estimation	6M Frames	No	512X424	Utilize Kinect sensors
PreSIL	Depth information, point clouds Semantic segmentation 2d and 3d labels for object detection	50K Frames	No	1920X1080	LiDAR simulator within Grand Theft Auto V

4.3 Silver Framework

Using the Unity game engine, we built a system that is able to generate a full city procedurally, randomly, at run-time and with high degree of photo-realism. The city is populated with humans, cars, trees, birds and so on. The system runs at around 30 Frames Per Second (FPS) on Ubuntu 18.04 with a GeForce GTX 1080 Ti GPU. As described in Figure 4.2, it consists of four main components:

- Scene generator;
- Camera controller;
- Weather and time unit;
- Ground-truth extractor.

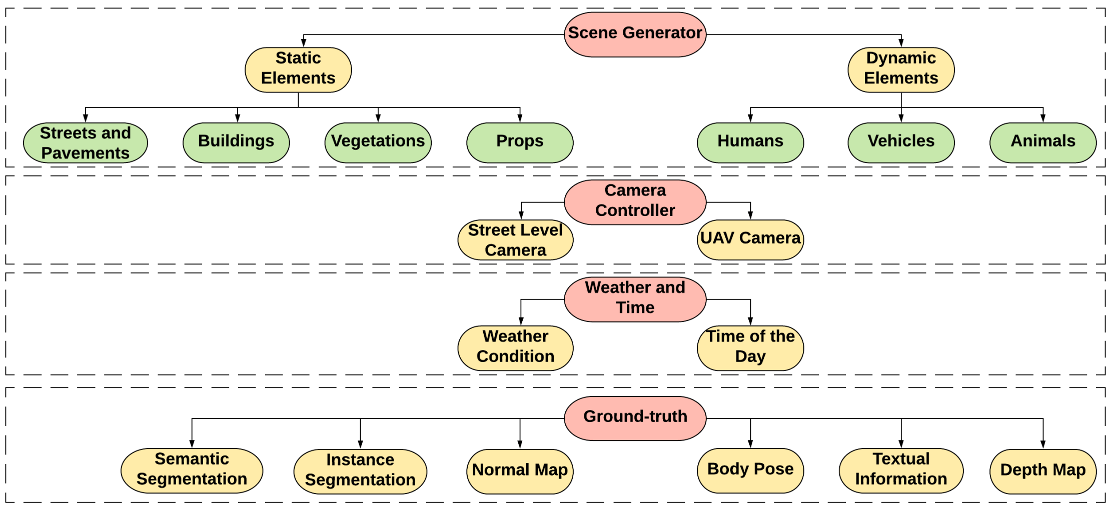


Figure 4.2: Bird's-eye view of the system architecture.

4.3.1 Scene generator

Starting from the given parameters, *Silver* initially creates the static part of the 3D virtual world. Once the static part of the scene is completed, the dynamic part of the scene is initiated. In the following, we discuss static and dynamic scene elements generators in detail.

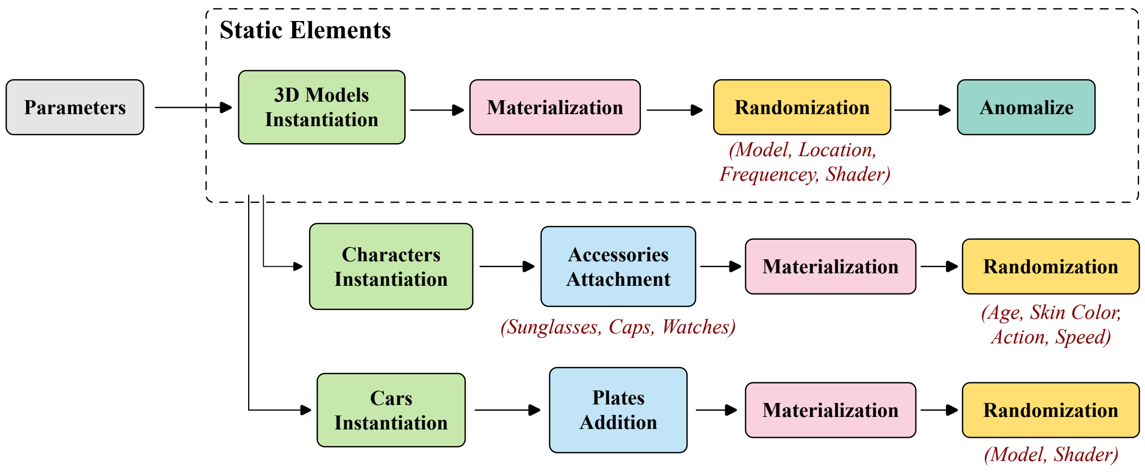


Figure 4.3: Flowchart describing the scene creation in our novel simulator.

4.3.1.1 Static elements generator

First, the street length and number of intersections (crosses) are randomly determined using a uniform distribution. The street length could vary based on predefined parameters or within a certain range and the number of crosses could also be determined randomly within a specified range. Following this, the buildings are created where buildings' locations, types and frequency are set at random. The locations of buildings are constrained by the street layout and the positions of intersections. The types of buildings include residential, commercial, and industrial, and their frequencies are based on probability distributions and other user-defined parameters. After that, the other scene elements like benches, trash containers, and bags, trees, and other elements are created.

4.3.1.2 Dynamic elements generator

Initially, the characters generator retrieves the locations of buildings and benches, and instantiates characters based on the required density and number of characters. At this stage, only avatars are created but they are not animated yet.

Animating an avatar typically involves adjusting the translations and rotations of each bone, of the character, to achieve a desired pose at a specific moment in time. This process can resemble a playback if the motions have been prerecorded, although modern approaches can be much more sophisticated than simple playback (M. C. Wibowo, Nugroho, and A. Wibowo, 2024; Lam and Fong, 2023). We use the Microsoft Rocketbox Avatar Library (Gonzalez-Franco, Ofek, et al., 2020) to define the character avatar. By utilizing this library, *Silver* allows us to instantiate virtual characters with different appearances to populate the scene. The animations are selected based on character's pose and action such as standing, sitting, walking, running, and talking. Character animations were adopted from Mixamo[1]. Mixamo offers a vast collection of motion capture-based animations for various actions and poses, ranging from simple movements like walking and running to more complex interactions. These actions are captured with the help of professional motion actors using advanced techniques. *Silver* selects appropriate animations based on the pose of each character to animate them realistically within the scene.

Then, the number of cars and models are selected at random. Additionally, car shader attributes, namely, *Smoothness*, *Metallic* and *BaseColour*, are all randomized at run-time to give different visual appearances even to the same car model. After that, the plates of the cars are selected at random from a set collected manually from the web. The above main processes are summarized in Figure 4.3.

4.3.2 Camera controller

The camera unit is made to be independent of the other scene elements to support the scalability of the system. Thus, the camera unit can function effectively regardless of the complexity of the scene. This independence of the camera unit from other scene elements makes it straightforward to integrate or adapt different elements, contexts, and scenarios, allowing for flexibility and efficient operation. Furthermore, this makes the system versatile and capable

[1]`https://www.mixamo.com`

of accommodating different camera types and recording settings, such as those involving Unmanned Aerial Vehicles (UAVs) or individuals carrying cameras. This adaptability contributes to the diversity and richness of the synthetic data generated by our data generator.

Silver includes two different camera types. One simulates a camera placed on a UAV or drone. While the second imitates a person carrying a camera and recording others. *Cinemachine* was utilized for that reason since it gives unlimited sets of behaviours that enrich the diversity of the generated synthetic data in terms of the camera view angle, transition, and many others.

4.3.3 Weather and time unit

The Unity game engine supports creating *Skybox*, which is a 6-sided cube that is rendered behind all scene objects. We deployed the High Dynamic Range Imaging (HDRI) images from PolyHaven[2] to create two realistic sets of skyboxes. One incorporated the day-time skyboxes from which one skybox at random will be chosen if the simulation time is a day-time. While the second includes the nighttime skyboxes from which one will be selected at random when the nighttime is being simulated. For the time simulation, other actions are taken to increase photo-realism and enrich the diversity. For example, the locations of the sun and the moon are changed randomly. The skybox is rotated at random and the sky emission is set at random as well. After that, the street lights are turned on and off based on the time of the day. The main processes involved in simulating the time-of-day are shown in Figure 4.4.

The weather condition component is essential in simulating the main conditions that could present some challenges for deep learning models. *Silver* currently allows researchers to generate four different weather conditions, namely, sunny, foggy, rainy and snowy. A parameter is used to specify the severeness (slight, very, extreme) of the weather condition. However, the weather is randomized within the chosen

[2]`https://polyhaven.com`

severeness level in a range that is still meaningful and reflects the severeness degree. The aim was to add further diversity even for a given weather severeness level.

After that, for rainy and snowy weathers, the Unity particle system is used to simulate both snow and rain drops. The density of these drops per unit volume is controlled by the severeness level. At the same time, the ground shader is changed to simulate water drops collision or to imitate snow accumulation on the ground. Following this, trees animations are changed (speed up or slow down) based in the severeness of the weather being simulated to give a sense of wind interaction with trees. The foggy weather condition is supported by the Unity games engine. Thus, only simple parameters tuning was required to simulate realistic foggy weather condition. The main processes of simulating the four weather conditions are shown in Figure 4.4.

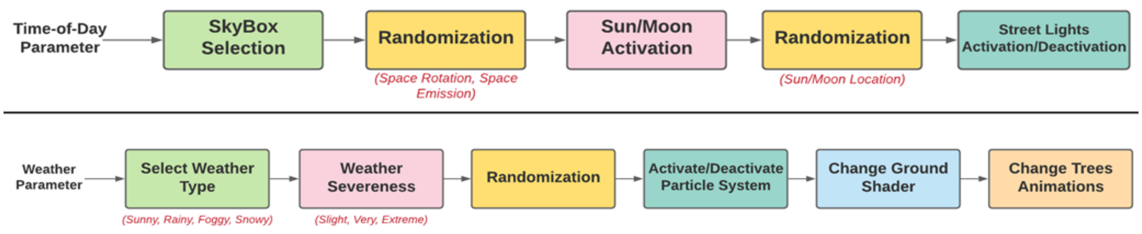


Figure 4.4: Time-of-day and weather simulation flow charts.

4.3.4 Ground-truth extractor

One of the main goals of *Silver* is to generate synthetic data with annotations for different computer vision tasks.

A custom rendering pass and a simple shader are defined to extract the *depth maps* for each frame. The vertex position and texture coordinates are retrieved then converted to screen space. After that, the depth information is linearized and scaled. *Normal maps* are simply extracted using graph shader and a custom pass volume. The world space normal vector is retrieved and encoded as the base colour for the lit shader.

For *instance segmentation*, the objects being created in the scene are retrieved and for each object its mesh is drawn again for each frame. For *semantic segmentation*, however, each category (class) of objects are given a different tag at the creation process. Then, all objects with one tag are given the same colour. It should be noted that Unity supports around 10K tags. Thus, the number of classes can reach 10K which is much larger than the number of semantic classes in most of the datasets (order of 10).

For the *human body pose* information, the avatar is accessed at each frame and the main 14 joints coordinates are provided. The world space locations of these joints are transformed to screen space locations and saved for each character.

4.4 System Evaluation

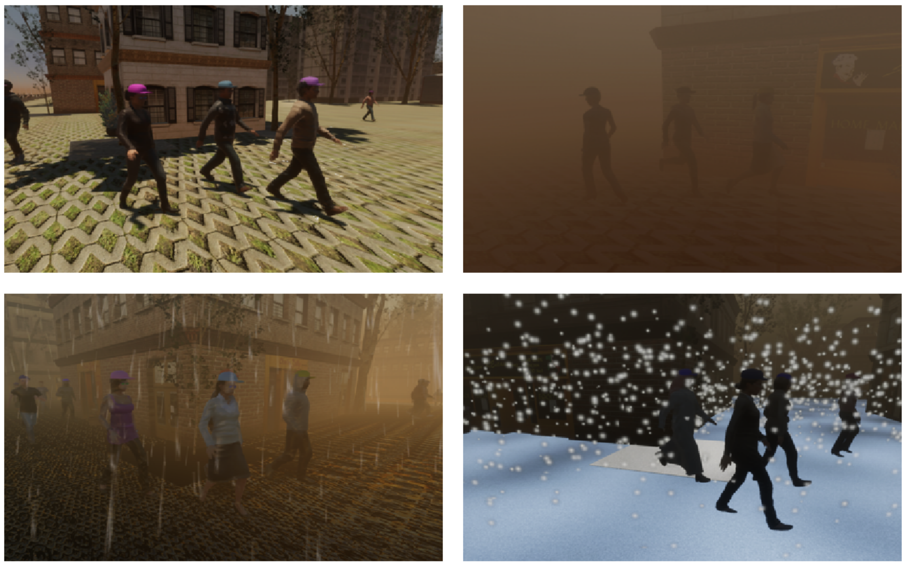
The system was evaluated both qualitatively and quantitatively. In Section 4.4.1 the quality and the variation of the generated images are discussed. On the other hand, time complexity and memory utilization are addressed in Section 4.4.2.

Table 4.2: Variations for some of the main attributes the proposed engine supports.

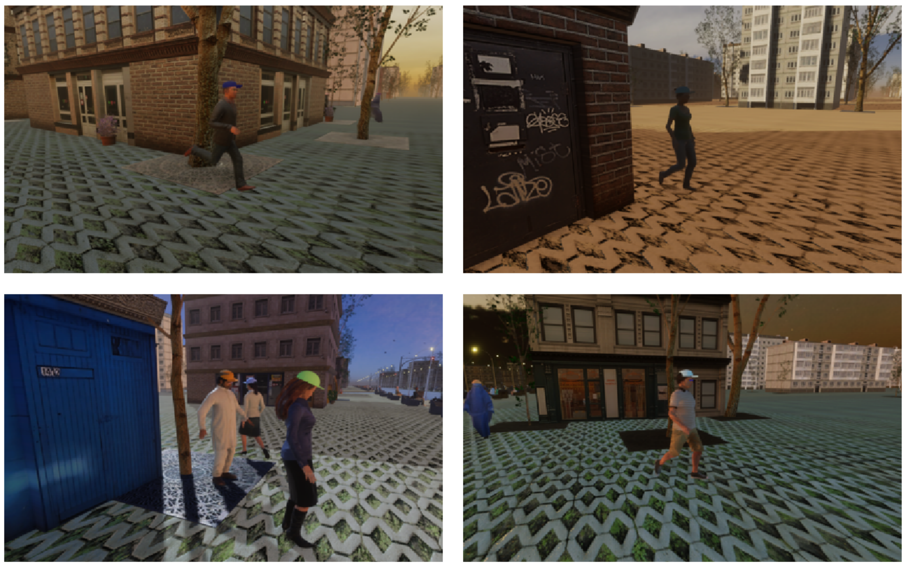
Attribute	Variations	Attribute	Variations	Attribute	Variations	Attribute	Variations
People	Gender - Age	Building	Shops	Weather	Normal	Human Accessories	Glasses
	Colour - Ethnicity		Condominiums		Rainy		Hats
	Height - Speed				Snowy		Hand watches
	Actions				Foggy		
Car	Model	Street	Length	Time of the Day	Day time	Vegetation	Trees
	Colour		Traffic Lights		Night		Plants
	Reflectiveness		Cross Locations				
	Plates		Cross Numbers				
Sky	Sun/Moon/Stars Location	Lighting	Traffic Lights	Prop	Waste Bin	Animal	Birds
	Stars Contrast		Street Lights		Waste Bags		
	Clouds Location				Benches		

4.4.1 Diversity and photo-realism

To avoid over-fitting to the visual features of the synthetic world, the content of the environment is diversified by including a wide set of 3D models, textures, animations,



(a) Different weather conditions.



(b) Different times-of-the-day.

Figure 4.5: Sample frames from sequences rendered at different weather conditions (a) and times-of-the-day (b).

weather conditions, illumination and lighting, and recording set-ups. A sample of the key variations *Silver* currently supports is illustrated in Table 4.2. In parallel to that, example frames from the supported weather conditions and times-of-the-day are shown in Figure 4.5.

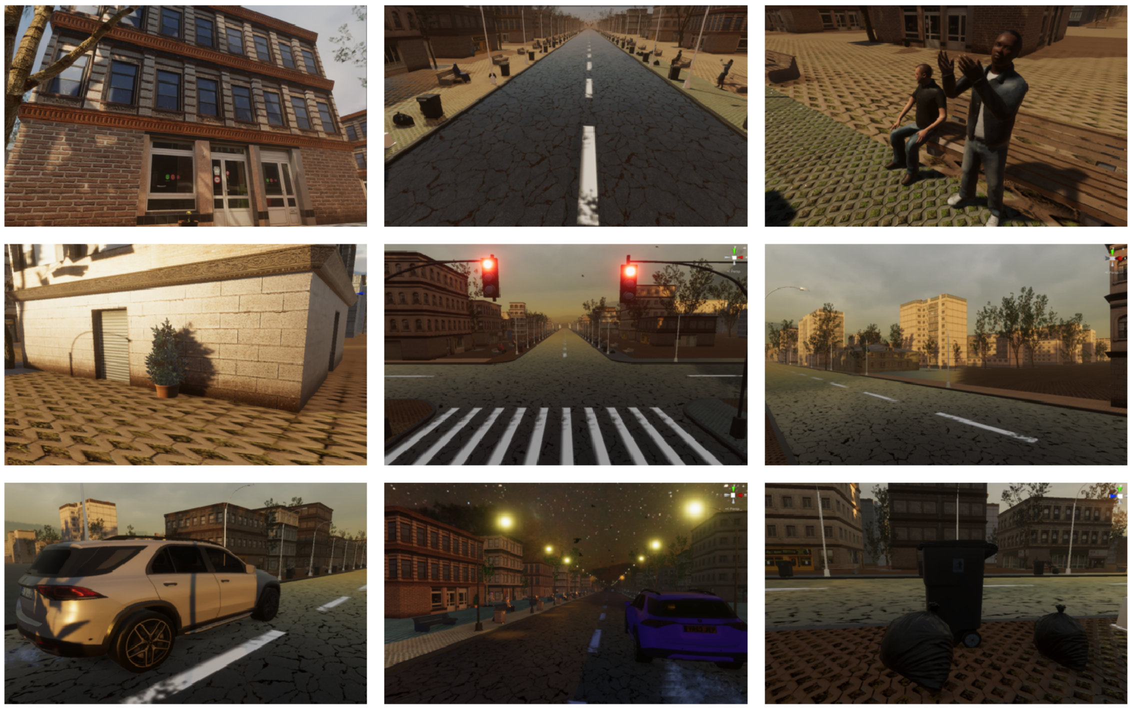


Figure 4.6: Photo-realism of *Silver*

Silver utilizes HDRP of Unity game engine to develop a photo-realistic 3D virtual world for synthetic data generation. HDRP is based on the Scriptable Rendering Pipeline (SRP). Generally, it is intended for high visual fidelity applications. The other key feature of HDRP is the Physical Light Unit (PLU) that relies on real-life lighting measurable values. All of these attributes together contribute to the final photo-realistic rendering that is shown in Figure 4.6.

Utilizing HDRP for the purpose of synthetic data generation is one of the key contributions of our work in this chapter. As shown in Figure 4.7 the synthetic data generators PHAV and Synthia generate less photo-realistic synthetic data as compared to *Silver* and Synscapes. Non-realistic weather simulation, low polygon 3D models, and unrealistic characters and vehicles are crucial issues behind the lack of photo-realism. An addition to this, the core problem is utilizing a non-physically based rendering pipeline which makes the interaction of the light with the materials clearly unrealistic. On the other hand, the main advantage of our system in comparison with Synscapes is the capability of *Silver* to simulate both different weather conditions and diverse nighttimes. Additionally, *Silver* diversifies buildings types, characters accessories, animations, age and skin color.

Table 4.3 details some key features of *Silver* as compared to other synthetic data generation systems. At the same time, it shows that these systems present a partial solution to the problem. Most of them are task specific, achieve diversity but neglect photo-realism or vice versa. For these reasons, *Silver* is developed to present a good solution for the synthetic data generation task.

Table 4.3: A comparison between *Silver* features and three state-of-the-art synthetic data generators, namely, Synscapes (Tsirikoglou et al., 2017; Wrenninge and Unger, 2018), Synthia (Ros et al., 2016), and PHAV (Roberto de Souza et al., 2017).

Simulator	Computer Vision Tasks					Weather Conditions				Times-of-Day		Public Availability	Photo-realism
	Semantic Segmentation	Instance Segmentation	Depth Estimation	Surface Normals Estimation	Pose Estimation	Normal	Rainy	Foggy	Snowy	Day-time	Nighttime	/	/
Synscapes	✓	✓	✓	-	-	✓	-	-	-	✓	-	-	✓
Synthia	✓	-	✓	-	-	✓	✓	-	✓	✓	✓	-	-
PHAV	✓	✓	✓	-	✓	✓	✓	✓	-	✓	✓	-	-
Silver	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

4.4.2 Scalability

One main goal of our work, presented in this chapter, is to make the system scalable and extendable to a range of different problems. In other words, adding new 3D models for human characters, buildings, trees, cars, or new animations, or textures, or camera setup should be straightforward. At the same time, extending the system to support new computer vision tasks shall not require major modifications to the other system components. This was achieved by following a modular approach in building *Silver* where each component is independent of the unrelated ones.

To dissect the complexity of the system, the following six components were analysed: *Triangles, Vertices and Batches Count, CPU Time, FPS*, and *SetPass Calls*. The first three represent the total number of triangles, vertices and batches

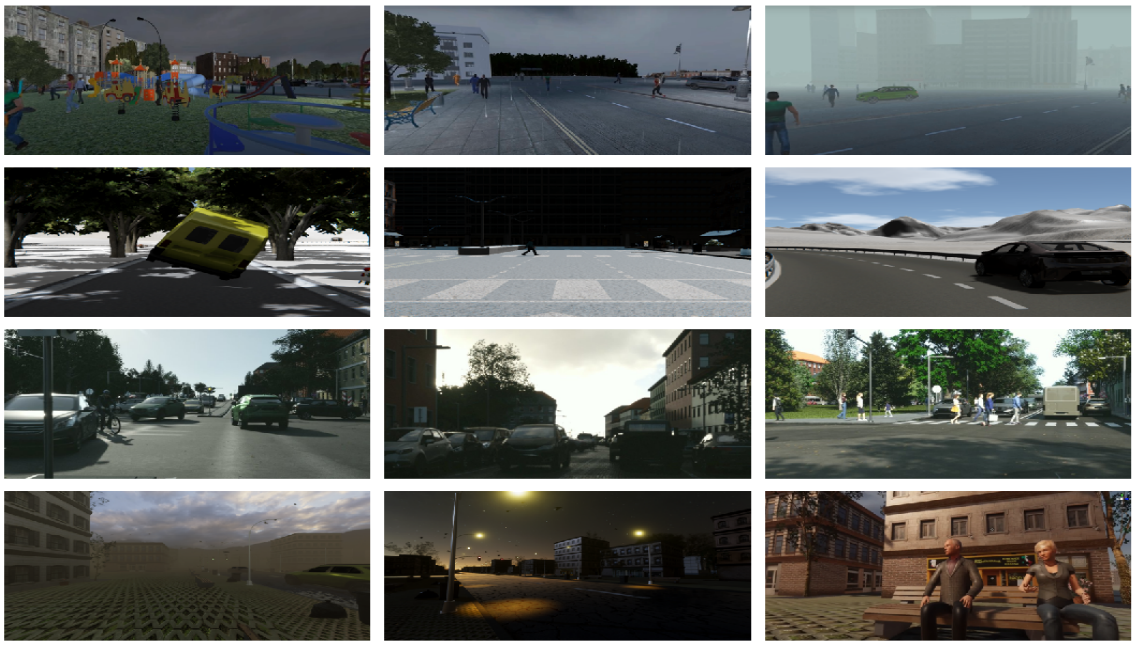


Figure 4.7: Visual comparison between *Silver* and three synthetic data generators in terms of photo-realism. First three rows show PHAV (Roberto de Souza et al., 2017), Synthia (Ros et al., 2016), and Synscapes (Tsirikoglou et al., 2017; Wrenninge and Unger, 2018), respectively, while final row presents our system *Silver*.

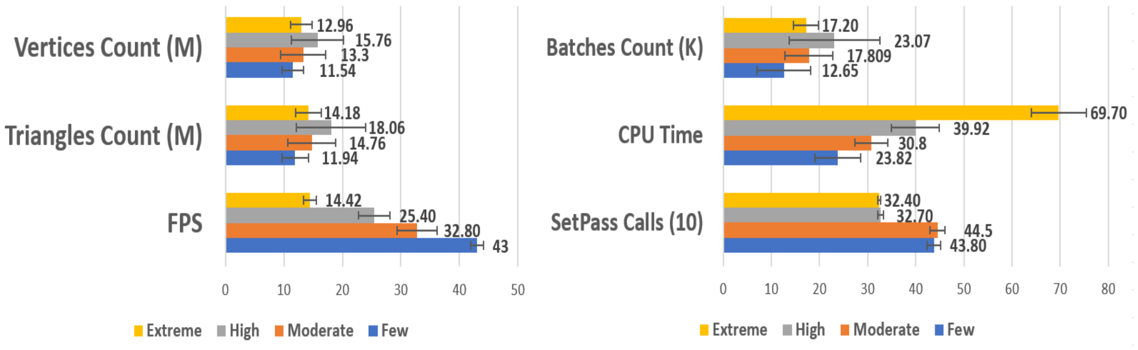


Figure 4.8: System complexity when varying number of characters.

(static, dynamic, and instance types) Unity processes for a single frame. CPU Time illustrates the time required to process and render a single frame. FPS, however, provides the number of frames rendered per second. The higher the value, the smoother the system will run. On the other hand, SetPass Calls shows how many times Unity changes the shader pass as it renders one frame.

Figure 4.8 shows the average and standard deviation values for these six metrics taken for 5 iterations. Scalability was studied for four crowdedness levels, namely Few (N≤20), Moderate (N∼50), High (N∼100), and Extreme (N∼300). N represents the number of generated pedestrians. Figure 4.8 clearly shows that the number of *Triangles*, *Vertices*, *Batches*, and *SetPass Calls* all are independent of the system load, i.e. generating more characters. *FPS* decreased and *CPU Time* increased as expected since more characters were generated.

4.4.3 Sample dataset

Our system automatically provides ground-truth for depth estimation, normal map, semantic segmentation, instance segmentation and body pose estimation. Figure 4.9 shows one RGB frame with its corresponding ground-truths for the mentioned computer vision tasks. In addition to other textual information describing the time of the day, the weather condition, and other useful information.

To show one particular usage of *Silver*, a sample dataset was generated and released publicly. The dataset contains sample sequences spanning the main supported attributes. The ability to generate different scenes and the process of sampling models, materials, and animations are thought to minimize the visual similarity even among a large number of sequences.

4.5 Concluding Remarks

In this chapter, we presented synthetic data generator called *Silver*, which allows researchers to generate 3D photo-realistic virtual worlds procedurally and at run-time. No computer graphics knowledge is required, no privacy issues are involved, and no manual annotation is needed. The system is released for research usage and researchers are welcomed to adopt it to their own needs. A sample dataset was generated to show the capability of the system.

This chapter was not meant yet to provide proof of the usability of the synthetic

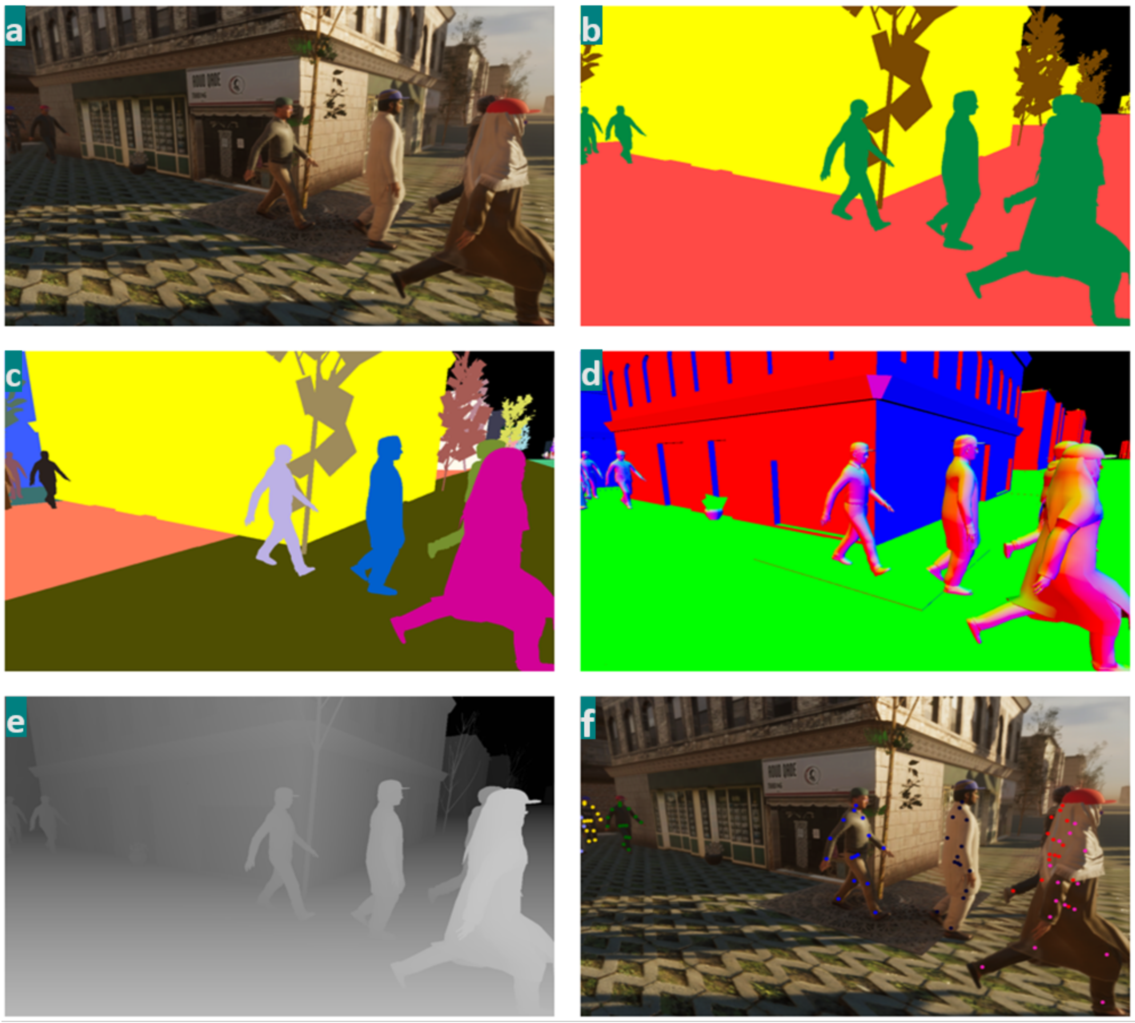


Figure 4.9: *Silver* provides RGB frames (a) with pixel-level accurate ground-truths for semantic segmentation (b), instance segmentation (c), normal maps (d), depth maps (e), and human body pose information (f).

data the system is able to generate. However, it was dedicated to representing the system, its main features, design decisions, and the reasons behind them. In Chapters 5 and 6, we are going to study the usability of the generated synthetic data for two major computer vision tasks i.e., semantic segmentation and video stabilization. It should be noted that *Silver* can be utilized to generate full datasets with rare and challenging attributes. The automatic annotation process ensures accuracy. At the same time, it provides unbiased ground truth since no human is involved in the annotation process.

Chapter 5

Synthetic-Aware Semantic Segmentation

In every block of marble I see a statue as plain as though it stood before me, shaped and perfect in attitude and action. I have only to hew away the rough walls that imprison the lovely apparition to reveal it to the other eyes as mine see it.

Michelangelo

The work presented in this chapter was published as a conference paper in Kerim, Chamone, et al. (2022).

Recent semantic segmentation models perform well under standard weather conditions and sufficient illumination but struggle with adverse weather conditions and nighttime. Collecting and annotating training data under these conditions is expensive, time-consuming, error-prone, and not always practical. Usually, synthetic data is used as a feasible data source to increase the amount of training data. However, just directly using synthetic data may actually harm the model's performance under normal weather conditions while getting only small gains in adverse situations. Therefore, we present a novel architecture specifically designed for using synthetic training data for domain adaptation. We propose a simple yet powerful addition to DeepLabV3+ by using weather and time-of-the-day supervisors trained with multi-task learning, making it both weather and nighttime aware, which improves its mIoU accuracy by 14 percentage points on the ACDC dataset while maintaining a score of 75% mIoU on the Cityscapes dataset.

The synthetic dataset, code, and our extended version of the *Silver* simulator are all publicly available under this chapter's GitHub repository at `https://github.com/lsmcolab/Semantic-Segmentation-under-Adverse-Conditions`.

5.1 Motivation

Understanding the environment using visual data has been an active research problem since the early beginning of computer vision. It started to attract even more researchers with the great advancement in autonomous cars (Teichmann et al., 2018; Kumar et al., 2021; Wiseman, 2022), human-computer-interaction (Ren and Bao, 2020; Nazar et al., 2021; Yubin Liu, Sivaparthipan, and Shankar, 2022), and augmented reality (Baroroh, Chu, and L. Wang, 2021; Costa, Petry, and Moreira, 2022; Chiang, Shang, and Qiao, 2022).

Semantic segmentation is at the core of these applications, with the data-driven supervised learning methods dominating this field, achieving state-of-the-art results (Ronneberger, Fischer, and Brox, 2015; L.-C. Chen, Y. Zhu, et al., 2018; Y. Yuan et al., 2019; Fu et al., 2019; Hengshuang Zhao et al., 2017). Training these models on real data requires large-scale human annotated images, which is expensive and time-consuming, especially for images taken under challenging weather and illumination conditions such as fog and nighttime. For instance, a person takes about 90 minutes to annotate an image from the Cityscapes dataset (Cordts et al., 2016), which contains only daylight and clear weather conditions, while it exceeds three hours for the Adverse Conditions Dataset with Correspondences (ACDC) (Sakaridis, D. Dai, and Van Gool, 2021) dataset.

Despite the success of recent semantic segmentation models in clear weather and standard illumination conditions, these methods struggle with adverse conditions (*e.g.*, rainy, foggy, snowy, and nighttime), which degrade the feature extraction process. Falling rain and snow particles change the visual appearance of objects, partially occlude them, and cause distortion on the camera sensor, while fog works as a low-pass filter, removing high-frequency components. Nighttime is even more problematic because of the dramatic change in the light distribution and other severe artifacts, such as lens flare, bright spots, and chromatic aberration. Yet, few works have tried to investigate the effect of weather conditions and nighttime in semantic segmentation (Guo et al., 2020; Lei et al., 2020; Alshammari, Akcay, and Breckon, 2020; Q. Xu et al., 2021; X. Ma et al., 2022). Although they achieve remarkable results, they are limited to one weather condition only and are too narrow in their scope.

In this chapter, we propose a novel training procedure to address the issues in the semantic segmentation under adverse conditions and in the annotation efforts, simultaneously. We leverage synthetic data to produce ground-truth images at no human annotation effort and create a new dataset, the AWSS, which is composed of images specially generated by our extended version of the *Silver* simulator (see Chapter 4). To reduce the gap between synthetic and real, our approach combines synthetic and real images by alternating their batches at training time as illustrated in Figure 5.1.

We also propose the Weather-Aware Supervisor (WAS) and the Time-Aware Supervisor (TAS), which are trained jointly with the main module to improve the feature extraction. Our main module derives from the DeepLabV3+ which contains the powerful atrous convolutions that increase the receptive field while not increasing the dimensions of feature maps and computation cost. Thus, better performance at low computation. Unlike the current methods that work only with a single weather condition, our approach can handle the three main ones, i.e., rainy, foggy, and snowy, as well as nighttime images. The results show that our novel model achieves state-of-the-art results under adverse weather conditions (0.49 mIoU on ACDC)

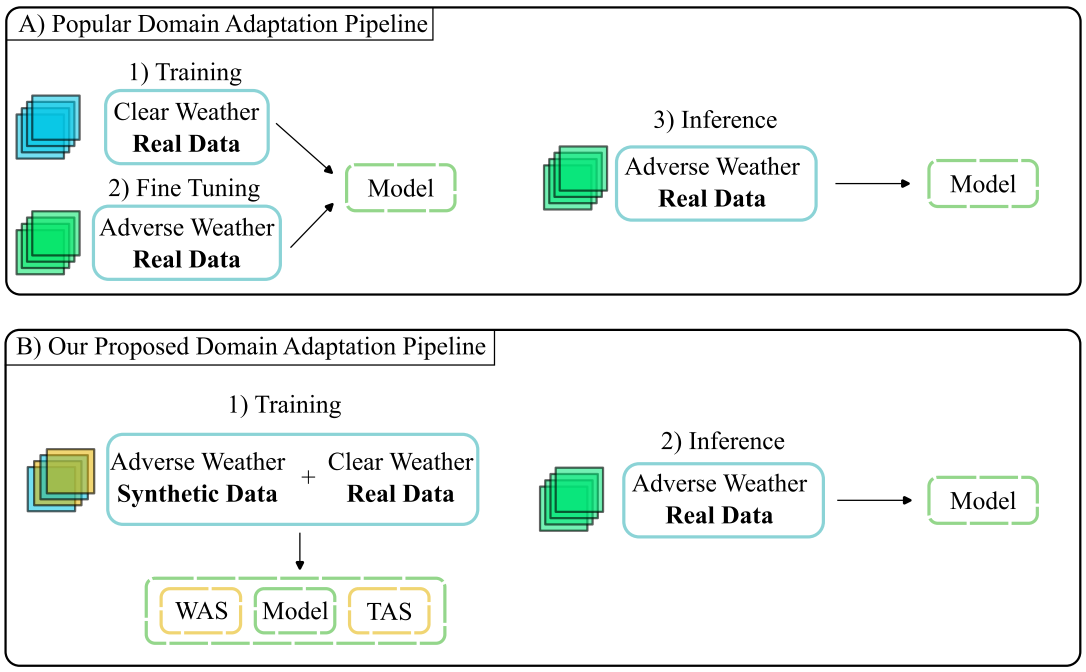


Figure 5.1: **Existing domain adaptation vs. our proposed pipeline.** Unlike other approaches, our pipeline utilizes synthetic data, Weather-Aware-Supervisor (WAS), and Time-Aware-Supervisor (TAS) to handle standard-to-adverse domain adaptation. Leveraging our synthetic-aware training procedure, we train our weather and daytime-nighttime aware architecture, simultaneously, on synthetic adverse weather and real normal weather data.

while it maintains adequate performance under standard conditions (0.75 mIoU on Cityscapes).

In summary, our contributions are three-fold:

- A novel synthetic-aware training procedure that can be used to train on both synthetic and real data simultaneously. In particular, we significantly improve DeepLabV3+ (L.-C. Chen, Y. Zhu, et al., 2018) robustness on adverse conditions by making its encoder both weather and nighttime aware;

- We extend the *Silver* simulator (see Chapter 4) to generate more photo-realistic and diverse adverse weather conditions and increase the supported semantic segmentation classes;

Table 5.1: **Comparison among synthetic semantic segmentation datasets.** Our proposed dataset, named AWSS, is composed of photo-realistic pixel-wise annotated images under standard and adverse conditions.

	Weather Conditions				Times-of-Day		Photo-realism	Public Availability
	Normal	Rain	Fog	Snow	Daytime	Nighttime	/	/
GTA-V (Richter, Vineet, et al., 2016)	✔	✔	-	-	✔	-	✔	✔
Synscapes (Tsirikoglou et al., 2017; Wrenninge and Unger, 2018)	✔	-	-	-	✔	-	✔	✔
Virtual KITTI (Gaidon et al., 2016)	✔	✔	✔	-	✔	-	-	✔
Synthia (Ros et al., 2016)	✔	✔	-	✔	✔	✔	-	-
SHIFT (T. Sun et al., 2022)	✔	✔	✔	-	✔	✔	✔	✔
AWSS (Our Dataset)	✔	✔	✔	✔	✔	✔	✔	✔

- Leveraging our extended version of *Silver*, we generate a new synthetic semantic segmentation dataset, the AWSS, composed of photo-realistic annotated images spanning foggy, rainy, and snowy weather conditions and nighttime attributes.

5.2 The AWSS Dataset

There have been many synthetic datasets proposed for the semantic segmentation problem. However, they are usually non-photo-realistic such as Synthia (Ros et al., 2016) and Virtual KITTI (Gaidon et al., 2016), limited in diversity such as GTA-V (Richter, Vineet, et al., 2016) and Synscapes (Tsirikoglou et al., 2017; Wrenninge and Unger, 2018) as clearly demonstrated in Table 5.1. Recently, SHIFT (T. Sun et al., 2022) dataset was introduced, which is photo-realistic and diverse similar to our generated synthetic dataset but does not cover the snowy weather.

We extend *Silver*, our developed simulator that was explained in Chapter 4, to generate adverse weather photo-realistic images along with their corresponding ground-truth for the semantic segmentation task. We generate the Adverse Weather Synthetic Segmentation (AWSS) dataset, which comprises 1,250 images with a resolution of 1,200 × 780 pixels and spans normal, rainy, foggy, and snowy weather conditions at daytime and nighttime. It follows the same conventions,

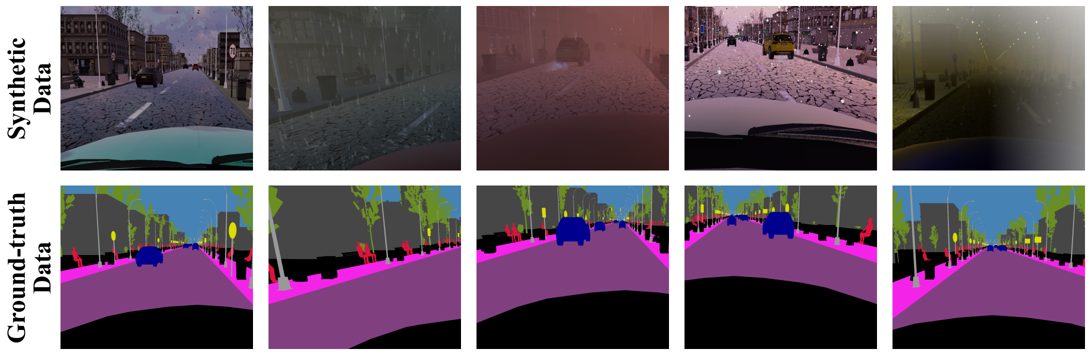


Figure 5.2: **Samples from AWSS dataset.** The generated AWSS synthetic dataset spans normal, rainy, foggy, snowy, and nighttime attributes.

i.e., classes definitions and color encoding, as Cityscapes (Cordts et al., 2016) and ACDC (Sakaridis, D. Dai, and Van Gool, 2021) datasets. However, we limit the number of classes to 10:

- *Road* and *Sidewalk*;
- *Building* and *Pole*;
- *Traffic Light* and *Traffic Sign*;
- *Vegetation* and *Sky*;
- *Person* and *Car*.

Figure 5.2 shows sample images from the AWSS dataset spanning various standard and challenging attributes.

5.3 Extensions to the Silver framework

Silver is based on the Unity game engine (Unity, 2024). It allows users to create 3D virtual worlds by only specifying a set of scene descriptive parameters like the weather condition, time-of-the-day, number of cars and humans, camera type, and lens artifacts. The simulator achieves photo-realism by using the recent High Definition Rendering Pipeline (HDRP). In addition, the simulator applies a set of

Procedural Content Generation (PCG) concepts to generate, populate, and control the scenes. For more details, please refer to Chapter 4.

5.3.1 Adverse conditions

The original simulator can simulate standard and adverse weather conditions at daytime and nighttime but with a limited photo-realism and diversity. For each weather condition, we diversify weather severeness, time-of-the-day, and other scene elements if not specified. Based on the environment being simulated, scene elements materials, shaders and textures are selected from a predefined large set. We customize and integrate Procedural Terrain (Rocks and (HDRP), 2022) with Adobe Substance materials (Material, 2022) to simulate photo-realistic snow accumulation on ground, mud, mold, wet surfaces, and water puddles. Water drops splashes on the ground are simulated by customizing the Unity particle system. Rain splash intensity is controlled by the rain weather severeness which is sampled from a uniform distribution. Additionally, we simulate slightly foggy weather condition once heavy rain is simulated. For nighttime simulation, street lights are turned on and their intensity is randomized. Some of these lights are flickered or turned off to increase diversity.

5.3.2 Dash camera mode

Initially *Silver* simulates Unmanned Aerial Vehicle (UAV) and first-person view cameras. However, most existing semantic segmentation datasets like Cityscapes (Cordts et al., 2016) and ACDC (Sakaridis, D. Dai, and Van Gool, 2021) datasets are recorded using a dash camera mounted on a car. To generate the AWSS dataset, we developed the dash camera mode to facilitate this task. Furthermore, to increase view angle diversity, we simulate vertical and horizontal lens shifts.

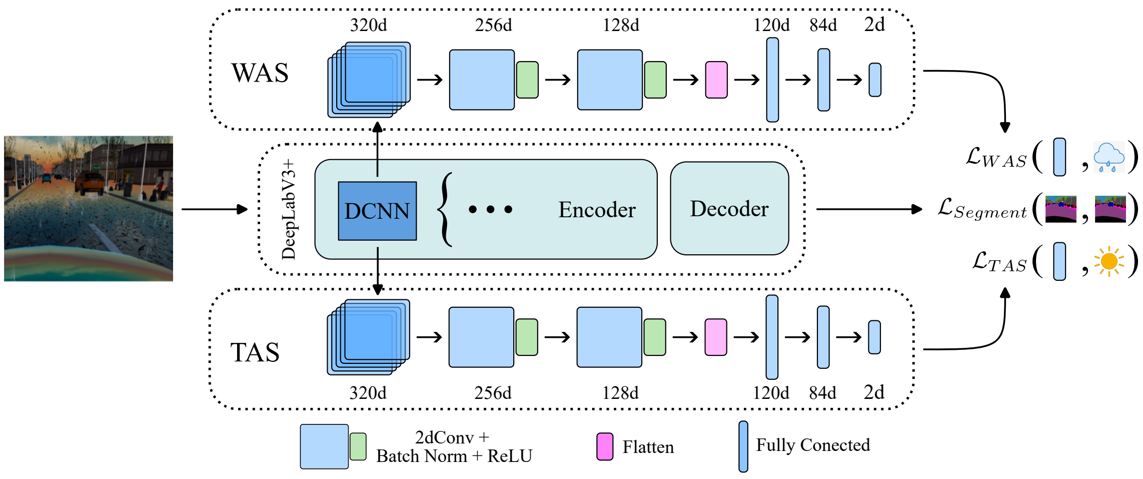


Figure 5.3: **An overview of our proposed architecture.** DCNN of DeepLabV3+ (L.-C. Chen, Y. Zhu, et al., 2018) is forced to learn weather and daytime-nighttime specific and roboust features by the means of multi-task learning. WAS and TAS branches learn to predict weather and daytime-nighttime, respectively. At the same time, they guide the encoder and specifically DCNN to learn extracting robust features under adverse and standard conditions.

5.3.3 Semantic segmentation automatic ground-truth

The simulator supports semantic segmentation automatic ground-truth generation. However, the number of semantic classes was limited to 4 classes: humans, ground, buildings, and trees. We extend the number of supported classes by adding new elements to the scene like traffic signs and modify the road mesh into road and sidewalk. At the same time, we customize the ground-truth generation pipeline to match Cityscapes (Cordts et al., 2016) color codes and conventions. With our extension, *Silver* now can provide semantic segmentation ground-truth for 10 classes, as specified earlier in this section.

5.4 Methodology

We aim to reduce the domain shift in adverse weather conditions while not acquiring additional real data. Hence, we propose a novel training approach that leverages synthetic data, while making the architecture aware of the weather condition and nighttime. Our proposed architecture is trained on both synthetic and real data simultaneously (see Figure 5.3).

The methodology is based on three components:

- Adding two simple networks WAS and TAS that work as supervisors to teach the model to learn weather and nighttime specific features;
- The full-model is trained using multi-task learning where the baseline learn semantic segmentation and WAS and TAS learns to predict weather condition and day-night, respectively;
- Our proposed model is trained on images from synthetic domain $\mathcal{D}_{adv-synth}$ and real domain $\mathcal{D}_{stand-real}$ in alternating fashion to ensure that the model learn to extract adverse weather features only from synthetic data which presents a proxy of the adverse real domain $\mathcal{D}_{adv-real}$. At the same time, it does not overfit to synthetic data and still ensure that the architecture other components leverage real data. Throughout this chapter, $\mathcal{D}_{stand-real}$, $\mathcal{D}_{adv-real}$, and $\mathcal{D}_{adv-synth}$ are represented by Cityscapes (Cordts et al., 2016), ACDC (Sakaridis, D. Dai, and Van Gool, 2021), and AWSS datasets, respectively.

5.4.1 Weather and nighttime aware encoder

We use the DeepLabV3+ (L.-C. Chen, Y. Zhu, et al., 2018) architecture because of its powerful encoder-decoder architecture. Originally, it is assumed that the encoder will learn how to extract low-level and high-level features independent of weather and illumination conditions. This strategy prevents the model from learning how to extract weather-specific features, resulting in low-quality features being fed to the decoder. The problem becomes even harder without training samples under these conditions.

To alleviate this problem, we focus on the Deep Convolutional Neural Network (DCNN) part of DeepLabV3+. We leverage multi-task learning to enforce the encoder to learn weather and time specific features. We add two simple identical models Weather-Aware-Supervisor (WAS) and Time-Aware-Supervisor (TAS). Each model is composed of two 3×3 atrous 2D convolutions with a rate of 2 and padding of 6. Each convolution is followed by a batch normalization and a rectified linear unit (ReLU). After this, the feature map is flattened and fed to 3 fully connected layers. The last layer predicts the weather for WAS and the daytime-nighttime for TAS. It is worth noting that WAS and TAS are only activated in the training process to guide the feature extraction learning process.

5.4.2 Multi-task learning to improve semantic segmentation

In the original implementation of DeepLabV3+ (L.-C. Chen, Y. Zhu, et al., 2018), the output of DCNN is passed to the remaining part of the encoder and to the decoder. In our implementation, we also feed the output of DCNN to WAS and TAS. The total objective to train the new architecture is defined as:

$$\min_{\theta} \mathcal{L} = \mathcal{L}_{Segment} + \alpha \times \mathcal{L}_{WAS} + \beta \times \mathcal{L}_{TAS}, \tag{5.1}$$

where $\mathcal{L}_{Segment}$ is the original loss used to train DeepLabV3+ (L.-C. Chen, Y. Zhu, et al., 2018), $\mathcal{L}_{WAS}$ and $\mathcal{L}_{TAS}$ are the cross-entropy losses utilized to train WAS and TAS, respectively. α and β are scalars to ensure numerical stability during the training and to give more emphasis to the main loss, i.e., $\mathcal{L}_{Segment}$.

It should be noted that each loss is back-propagated separately. $\mathcal{L}_{Segment}$ is back-propagated over all the architecture except WAS and TAS. On the other hand, $\mathcal{L}_{WAS}$ and $\mathcal{L}_{TAS}$ are back-propagated only to DCNN.

5.4.3 Synthetic-aware training

Training on source domain and fine-tuning on the target domain is a well-known approach to mitigate the domain gap (Tzeng et al., 2017). However, it is not practical as it requires annotated real data from the target domain which may not be always affordable. Furthermore, training the model on data from one distribution and then forcing the model to learn a new distribution limits the ability of the network to learn and may not converge to a global minima.

Thus, we propose training the modified DeepLabV3+ (L.-C. Chen, Y. Zhu, et al., 2018) on data from both synthetic and real distributions simultaneously and from scratch. For that aim, we train in alternating fashion: one batch from $\mathcal{D}_{stand-real}$ and next batch from $\mathcal{D}_{adv-synth}$. At the same time, since the aim is to learn how to extract useful features under adverse conditions, we freeze DCNN weights when training on a batch from $\mathcal{D}_{stand-real}$ and update them for a batch from $\mathcal{D}_{adv-synth}$. It is worth noting that all other weights are updated for data from both domains. This strategy encourages the encoder to leverage synthetic data to better learn feature extraction for the target domain while it ensures that the decoder is learning how to interpret both features to perform segmentation task under standard and adverse conditions.

5.5 Experiments

Datasets. For training, we use two datasets: AWSS dataset and the training split of Cityscapes (Cordts et al., 2016) dataset. For evaluation, we use validation splits of Cityscapes and ACDC (Sakaridis, D. Dai, and Van Gool, 2021) datasets. The three datasets follow the same conventions and color codes. Cityscapes comprises 2975 training images and 500 validation images. It is captured in urban scenes under normal weather conditions in the daytime. ACDC validation split comprises 506 images spanning rainy, foggy, snowy weather conditions and nighttime attributes.

Implementation details. Experiments are conducted on a Tesla V100 GPU. For all experiments, we keep the default parameters of the authors. For our adopted DeepLabV3+ architectures, we use a batch size of 4 while we keep all other parameters the same as DeepLabV3+. For DeepLabV3+ baseline, our architecture, and all ablation study experiments, we train for $30K$ iterations. We set $\alpha = \beta = 10^{-5}$, as these values achieved the best results.

Baselines. To analyse the robustness of recent semantic segmentation methods under adverse conditions, we use DeepLabV3+ (L.-C. Chen, Y. Zhu, et al., 2018), HRNet (Y. Yuan et al., 2019), DANet (Fu et al., 2019), and PSPNet (Hengshuang Zhao et al., 2017).

Evaluation metric. We use the common Mean Intersection over Union (mIoU) (L.-C. Chen, Y. Zhu, et al., 2018; Y. Yuan et al., 2019; Fu et al., 2019; Hengshuang Zhao et al., 2017) on the validation sets of Cityscapes and ACDC similar to (P.-R. Chen et al., 2020; Xie et al., 2022; Musat et al., 2021).

5.6 Results

Before discussing our architecture results, we will discuss how the domain shift degrades the state-of-the-art, and the improvements achieved by fine-tuning on synthetic data.

5.6.1 Standard-Adverse domain shift

As shown by results in Table 5.2, the performance of recent methods clearly degrade under adverse weather conditions and at nighttime (see rows *Baseline*). Additionally, it seems that snow and nighttime represent a clear challenge for recent models. Snow causes a drastic change in scene appearance: falling snow particles, snow on pavements and other scene elements makes these objects considerably different compared to what the model learned in the training phase. Thus, the

Table 5.2: **mIoU results for our approach Vs. standard domain adaptation methods.** Training our proposed weather and nighttime-aware architecture on both Cityscapes (Cordts et al., 2016) and AWSS, improves the performance on ACDC (Sakaridis, D. Dai, and Van Gool, 2021) dataset and achieves adequate performance on Cityscapes (Cordts et al., 2016). Best results are **bolded**. Fnt stands for Fine-Tuned.

		ACDC					Cityscapes
		Rain	Fog	Snow	Night	Overall	Overall
DeepLabV3+ (L.-C. Chen, Y. Zhu, et al., 2018)	Baseline	0.41	0.46	0.36	0.17	0.35	0.78
	FnT on AWSS	0.44	0.48	0.47	0.19	0.39	0.59
HRNet (Y. Yuan et al., 2019)	Baseline	0.46	0.42	0.41	0.09	0.35	0.75
	FnT on AWSS	0.47	0.49	0.35	0.14	0.36	0.51
DANet (Fu et al., 2019)	Baseline	0.47	0.57	0.44	0.21	0.42	0.82
	FnT on AWSS	0.48	0.58	0.48	0.26	0.45	0.74
PSPNet (Hengshuang Zhao et al., 2017)	Baseline	0.49	0.54	0.43	0.20	0.41	**0.86**
	FnT on AWSS	0.52	0.56	0.46	0.18	0.43	0.86
Ours	Full-Model	**0.57**	**0.60**	**0.50**	**0.27**	**0.49**	0.75

model struggles to segment these elements. Similarly, nighttime scenes with the radical decrease in illumination present a major challenge for segmentation methods.

5.6.2 Domain adaptation using synthetic data

Transfer learning is usually applied to handle a domain shift. However, although it improves the performance on the target domain, it generally degrades the performance on the source domain. As shown in Table 5.2 (FnT on AWSS), we can improve the performance of each semantic segmentation model. For some attributes like night and snow, the improvement was more than 50% (e.g. HRNet (Y. Yuan et al., 2019) under night). Generally, each semantic segmentation model was able to leverage AWSS to improve its performance for each adverse attribute. However, when evaluating these fine-tuned models on the original domain

(Cityscapes), we see a clear degradation in performance. This degradation was more severe for some models like HRNet (Y. Yuan et al., 2019) while it was slight for PSPNet (Hengshuang Zhao et al., 2017)

Table 5.3: **Per-class mIoU results on ACDC (Sakaridis, D. Dai, and Van Gool, 2021) dataset.** Our model achieves the best overall results on ACDC (Sakaridis, D. Dai, and Van Gool, 2021). It maintains the best results on *Road*, *Sidewalk*, *Building*, and *Person* classes. Best and second best results are **bolded** and <u>underlined</u>, respectively.

	Road	Sidewalk	Building	Pole	Tr. Light	Tr. Sign	Vegetation	Sky	Person	Car	Overall
DeepLabV3+	<u>0.71</u>	<u>0.22</u>	0.31	0.18	0.22	0.29	**0.72**	0.38	0.24	0.23	0.35
HRNet	0.55	0.16	0.44	0.14	0.28	0.24	0.66	**0.72**	0.07	0.19	0.35
DANet	0.68	0.11	0.19	<u>0.28</u>	**0.54**	**0.67**	0.26	0.65	<u>0.29</u>	**0.53**	<u>0.42</u>
PSPNet	0.63	0.12	<u>0.60</u>	**0.30**	<u>0.48</u>	<u>0.41</u>	0.62	0.61	0.21	0.17	0.41
Ours	**0.79**	**0.40**	**0.63**	0.25	0.26	0.33	<u>0.69</u>	<u>0.66</u>	**0.32**	<u>0.52</u>	**0.49**

5.6.3 Weather and night aware architecture

While the previous solution is simple, the improvement on the target domain was limited, and the performance on the source domain was sharply degraded. As a remedy, our architecture-based solution achieves the best results on the target domain and it maintains an adequate performance on the source domain. As reported in Table 5.2, making the model aware of the weather condition and daytime-nighttime attributes of the images in the training phase helps the model to learn how to extract more efficient features under both standard and challenging scenarios. Qualitative results are shown in Figure 5.4. Furthermore, per-class results are demonstrated in Table 5.3, our model achieves the best results on *Road*, *Sidewalk*, *Building*, and *Person* semantic classes. The largest improvement was on the *Sidewalk* which is around 82% improvement compared to DeepLabV3+, the best performing baseline on this class. As expected, snow and rain changes the visual appearance of this class significantly. This is because of snow accumulation, footsteps on snow,

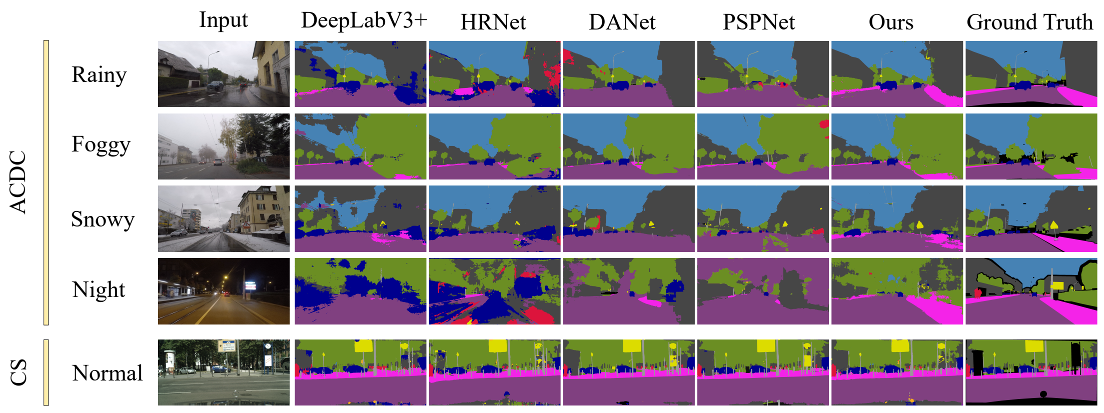


Figure 5.4: **Visual comparison between baselines and our approach.** Segmentation results are shown on ACDC (Sakaridis, D. Dai, and Van Gool, 2021) and Cityscapes (Cordts et al., 2016) dataset, respectively.

rain splash and mud, in addition to light reflection due to wet surface when raining.

5.7 Ablation Studies

To understand the effect of each design decision, we perform several experiments.

5.7.1 Training data type

We train the baseline model on AWSS from scratch (see Table 5.4 first row). As expected, training on synthetic data alone does not achieve satisfactory results due to domain gap between synthetic and real data. Thus, this suggests that AWSS can be used as complementary to the real data and not as an alternative. On the other hand, training the model from scratch on standard weather will perform well on these conditions but will fail under challenging conditions (refer to Table 5.4 second row).

Table 5.4: **Ablation analysis of weather and time awareness on performance.** Making the DeepLabV3+ weather and time aware improved the performance significantly at both normal weather, i.e. Cityscapes (Cordts et al., 2016) (CS), and adverse weather, i.e. ACDC (Sakaridis, D. Dai, and Van Gool, 2021), scenarios. Best and second best results are **bolded** and <u>underlined</u>, respectively.

	Training Mode	ACDC					Cityscapes
		Rain	Fog	Snow	Night	Overall	Overall
	scratch on AWSS	0.24	0.25	0.26	0.11	0.22	0.27
Baseline	scratch on CS	0.41	0.46	0.36	0.17	0.35	**0.78**
	scratch on CS and fine-tuned on AWSS	0.44	0.48	0.47	0.19	0.39	0.59
	scratch on CS and AWSS	0.41	0.43	0.38	0.19	0.35	0.69
Ours	scratch on CS and AWSS + Weather Aware	<u>0.49</u>	<u>0.55</u>	<u>0.47</u>	<u>0.20</u>	<u>0.43</u>	*0.73*
	scratch on CS and AWSS + Weather and Nighttime Aware	**0.57**	**0.60**	**0.50**	**0.27**	**0.49**	<u>0.75</u>

5.7.2 Training strategy

As shown in Table 5.4 third row, the standard method of transfer learning (fine-tuning the last layers on the target domain) improves the performance on the target domain but degrades the performance on the source domain.

5.7.3 Weather-Time awareness

Our approach achieves the best results under adverse conditions while still maintaining a satisfactory performance under standard conditions. Making the model synthetic aware and training the model without weather and nighttime-awareness achieve better results on the source domain but low performance on the target domain, compared to fine-tuning experiment. Adding the weather awareness to the model, i.e WAS, improves the performance at standard and adverse conditions. All adverse weather attributes were improved clearly as expected but the night attribute maintained a slight improvement. Finally, making the model aware of nighttime too, boosts significantly the performance under nighttime. Interestingly, it improves also the performance of the other weather conditions too. This is expected as TAS and WAS teachers allow the model to learn weather specific and nighttime-specific robust features which enables the model to achieve better results under these challenging conditions.

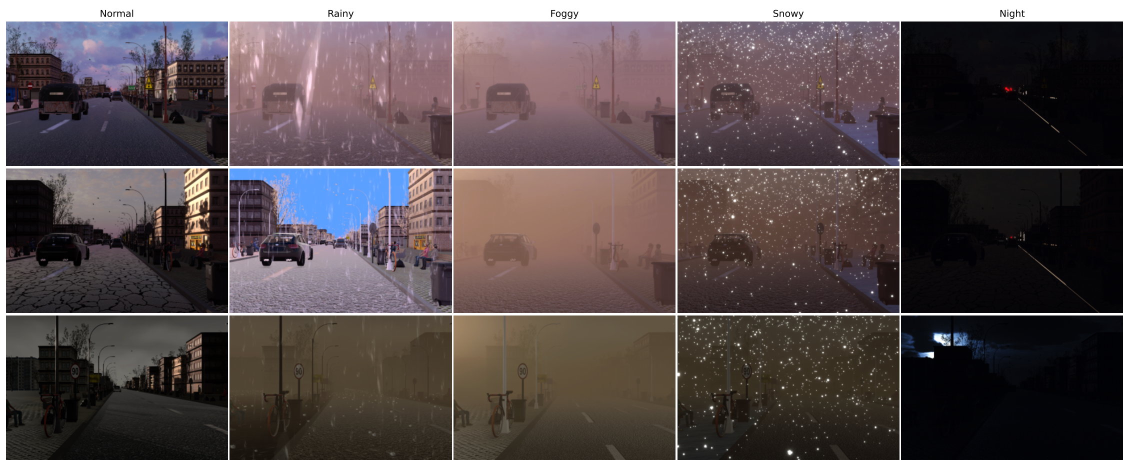


Figure 5.5: **Varying weather conditions for the same scene and viewpoint.** Images depicted in one row were generated from the same scene and viewpoint. Each column represents varying weather conditions and time of day.

5.7.4 Encoder features visualization

First, we utilize our *Silver* simulator to generate a small dataset. The dataset is composed of 200 images: ten frames long four sequences for each condition captured from the same scene and viewpoint but under normal and adverse conditions as shown in Figure 5.5. Each row represents a different scene. The adverse weather conditions include rainy, foggy, and snowy conditions. We also generate images in the daytime and nighttime.

We use T-distributed Stochastic Neighbor Embedding (T-SNE) (Van der Maaten and G. Hinton, 2008) to visualize the high-dimensional low-level and high-level features learned by *DeepLabV3+'s* encoder and compare them to the ones learned by our model. Low-level and high-level features are leveraged by the decoder to predict

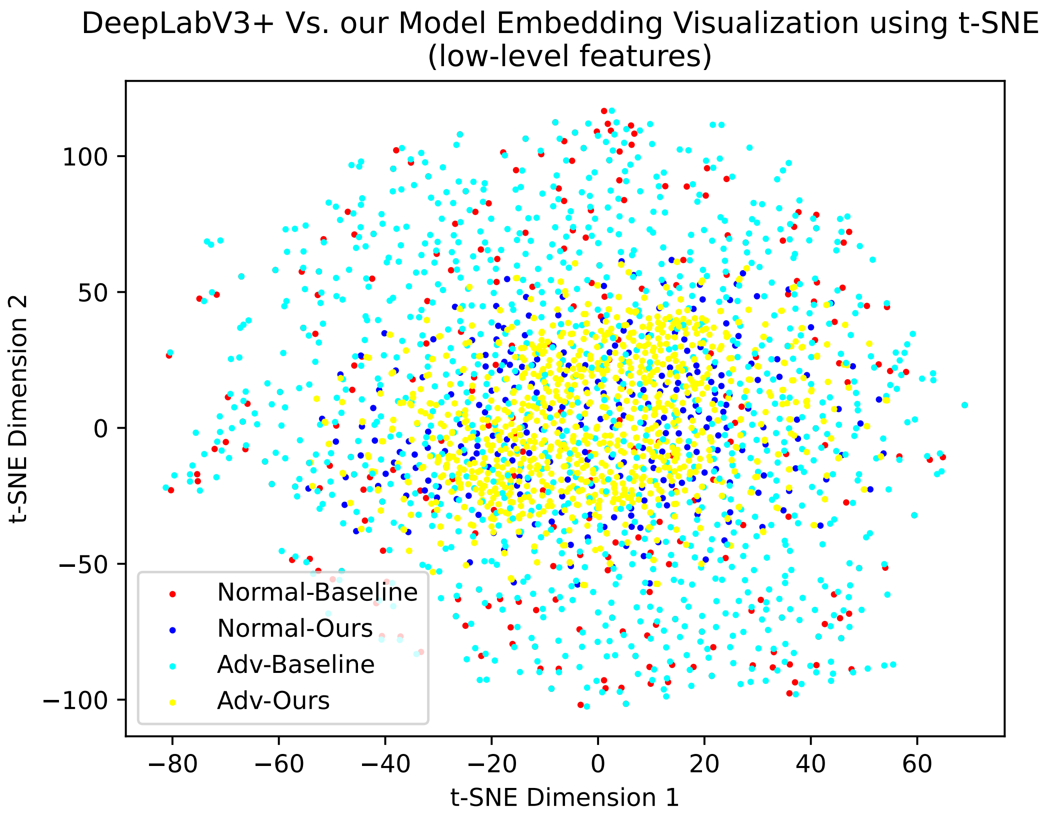


Figure 5.6: **Visualisation of low-level features learned by *DCNN* of *DeepLabV3+* compared to our model.** The DCNN of our model learns more weather and time-invariant features compared to the baseline.

the final semantic segmentation of a given RGB image. Therefore, visualizing these features provides insight into whether the learned features by our model are normal-adverse conditions-invariant as we predicted.

Figure 5.6 demonstrates the embeddings of both models at normal and adverse conditions. Our model helps the *DCNN* of the encoder to learn more weather-invariant features at this stage. Next, we also visualize high-level features learned by the encoder of both models. Figure 5.7 shows that adverse weather high-level features, learned by our encoder, are yet more invariant compared to those of the baseline as expected.

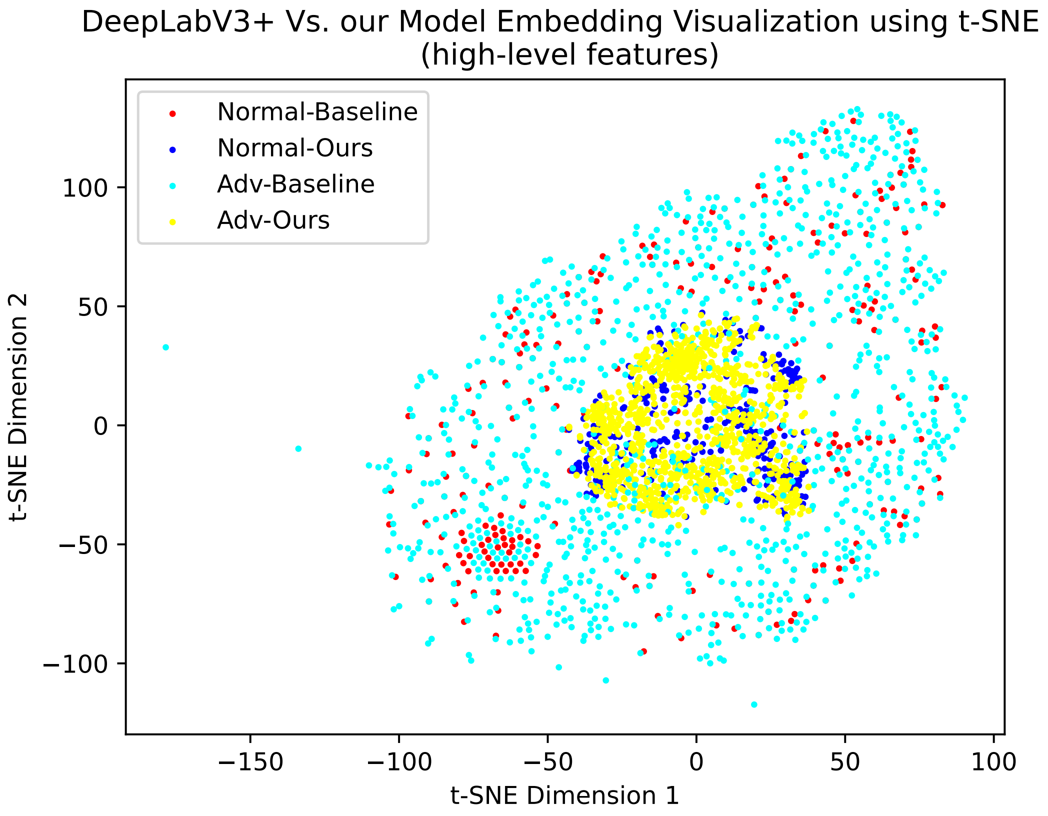


Figure 5.7: **Visualisation of high-level features learned by the encoder of *DeepLabV3+* compared to our model.** Our model's encoder learns, clearly, more weather and time-invariant features.

A key point to consider in this scope is that our model was designed and trained with a focus on real images for testing purposes. Although synthetic data was utilized during the training phase, the model's optimization primarily targeted real-world data rather than synthetic ones. Therefore, what was just illustrated with synthetic data can be also generalized to real data. This explains the good results obtained by our modified version of *DeepLabV3+* under adverse conditions.

5.8 Concluding Remarks

In this chapter, we introduced a novel synthetic dataset, the AWSS, that covers various adverse conditions. We showed that fine-tuning four state-of-the-art semantic segmentation models improves performance under adverse conditions but degrades performance under standard conditions. Our proposed solution shows that making the model aware of the synthetic data and utilizing a weather-aware-supervisor and time-aware-supervisor achieves the best results under adverse weather conditions while maintaining adequate performance under standard conditions.

Chapter 6

Synthetic-Aware Video Stabilization

The biggest risk is not taking any risk. In a world that's changing quickly, the only strategy that is guaranteed to fail is not taking risks

Mark Zuckerberg

The work presented in this chapter was published as a conference paper in Kerim, Ramos, et al. (2024).

Stabilization plays a central role in improving the quality of videos. However, current methods perform poorly under adverse conditions. In this chapter, we propose a synthetic-aware adverse weather video stabilization algorithm that dispenses real data for training, relying solely on synthetic data. Our approach leverages specially generated synthetic data to avoid the feature extraction issues faced by current methods. To achieve this, we present a novel data generator to produce the required training data with an automatic ground-truth extraction procedure. We also propose a new dataset, VSAC105Real, and compare our method to five recent video stabilization algorithms using two benchmarks. Our method generalizes well on real-world videos across all weather conditions and does not require large-scale synthetic training data. Implementations for our proposed video stabilization algorithm, extended version of the generator for video stabilization task, and datasets are available at `https://github.com/A-Kerim/SyntheticData4VideoStabilization_WACV_2024`.

6.1 Motivation

Over the past several years, we have witnessed an explosion of videos being recorded and shared on the Internet. However, most shared videos are unedited and shaky, which makes them unpleasant to watch. Therefore, video stabilization techniques became essential in the video processing pipeline, gaining momentum as more unedited videos are being created and shared. Recent video stabilization approaches perform well under standard conditions but struggle under adverse ones. Moreover, collecting training videos in these adverse conditions is hard, dangerous, and time-consuming. Furthermore, creating photo-realistic synthetic videos simulating these conditions is complex, expensive, and poses certain limitations because of the domain gap problem between synthetic and real domains.

The training data bottleneck mentioned above causes many video stabilization methods to be essentially non-learning-based, commonly adopting affine or homography matrix estimation in the camera motion estimation step to extract the camera trajectory. Usually, feature extraction, description, and matching are involved in this process. Feature extractors like SIFT (Lowe, 2004) and the learning-based ones, like R2D2 (Revaud et al., 2019) and ASLFeat (Luo et al., 2020), perform well under standard conditions, but they may fail under challenging conditions, such as foggy, rainy, and snowy weather, as well as nighttime scenes. For instance, raindrop and snowflake particles along with texture-depleted scenes in fog or darkness pose a clear challenge to extract robust features. Failing to estimate the camera trajectory accurately, in the motion estimation stage, propagates the error to later steps of the process, decreasing the quality of the stabilized video.

Synthetic data have shown great progress in the field of computer vision, captivating numerous researchers who seek to apply it to diverse computer vision problems (Yilin Liu, Xue, and H. Huang, 2021; Kerim, Celikcan, et al., 2021; Shafaei, James J Little, and Schmidt, 2016a; Tsirikoglou, 2022; Piano et al., 2023). Most importantly, synthetic data holds the promise of addressing the lack of suitable data for training supervised learning models. However, in this thesis, we argue that the potential of synthetic data lies not only in the amount of generated data for training but also in how we design and use this data jointly with our methods.

In this chapter, we present a novel synthetic-aware video stabilization method that leverages synthetic data and achieves state-of-the-art results using only a small-scale synthetic dataset. Using specially designed synthetic videos for training, our algorithm can bypass the feature extraction step, commonly adopted by video stabilization methods, and thus be more robust to adverse weather conditions. We leverage the Unity engine to build a simulator that creates three-dimensional photo-realistic virtual worlds procedurally at run-time. The system automatically diversifies essential scene attributes like weather conditions, time of the day, and crowdedness. For more details, please refer to Chapter 4.

We also extend our simulator,*Silver*, to generate the required ground-truth training data for our learning-based video stabilization method. We also introduce VSAC105Real, a new evaluation dataset with real adverse weather videos, since available benchmarks lack these weather conditions.

Our proposed method does not require any real data for training and is more robust than the state-of-the-art methods across different weather conditions. To the best of our knowledge, this is the pioneering work addressing video stabilization in adverse weather, utilizing synthetic videos. Despite supervised learning-based approaches (M. Wang et al., 2018; Y.-L. Liu et al., 2021; J. Yu and Ramamoorthi, 2020; M. K. Ali, S. Yu, and Kim, 2020) being able to learn parameters like cropping window, sensitivity, and even to extract discriminative features, there is no sufficient labeled data for obtaining high-quality results in any condition. Sourcing, collecting, and annotating relevant data is cumbersome, time-consuming, error-prone, costly, and subject to privacy issues.

Hence, our main contributions are three-fold:

- A novel synthetic-aware video stabilization method, achieving state-of-the-art results on real videos while trained only on synthetic videos;
- A new synthetic data generator capable of producing specially designed training videos;
- A new video stabilization dataset, VSAC105Real, composed of real videos spanning foggy, rainy, snowy weather, and nighttime attributes.

6.2 Methodology

Let $V = \{v_1, \dots, v_N\}$ be a shaky video composed of N frames. Our approach aims to generate a stabilized version $V' = \{v'_1, \dots, v'_N\}$ while preserving the original camera movement made by the recorder. Our method has two stages:

- Motion estimation;
- Trajectory smoothing.

In the Motion estimation stage, we train a motion estimation network using the ground-truth data from generated synthetic videos to estimate an affine transformation matrix $\mathbf{A}_i$ for every consecutive frames v_i and v_{i+1}. Then, in the trajectory smoothing stage, we calculate the camera trajectory $\hat{T} = \{\hat{\tau}_1, \dots, \hat{\tau}_N\}$ from the estimated parameters $\hat{\mathbf{x}}_j$ for the pair of frames (v_j, v_{j+1}), and after smoothing $\hat{T}$, we warp and crop frames using transformations retrieved from the smoothed trajectory $\tilde{T} = \{\tilde{\tau}_1, \dots, \tilde{\tau}_N\}$. Figure 6.1 shows the pipeline.

6.2.1 Motion estimation

The first stage of our proposed pipeline estimates the camera motion throughout the video. Most existing 2D-based stabilization approaches apply key-point feature extraction and tracking to solve this task (K.-Y. Lee et al., 2009; Grundmann,

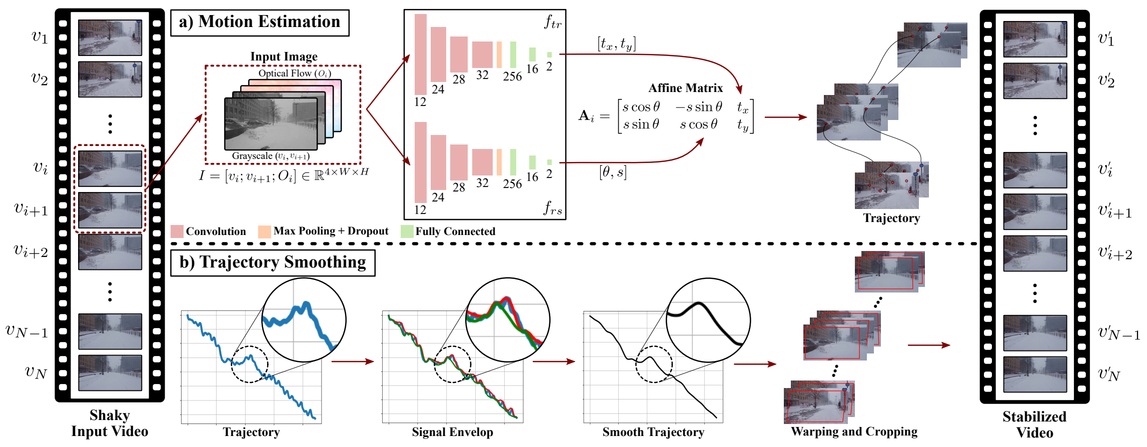


Figure 6.1: **Video stabilization pipeline.** Our method estimates the translation, rotation, and scale for each pair of frames of the shaky video. After computing the camera trajectory, upper (red) and lower (green) bounds are found and averaged, and the Savitzky-Golay filter is applied to smooth the trajectory. Finally, warping and cropping are performed.

Kwatra, and Essa, 2011; S. Liu, Y. Wang, et al., 2012). However, both steps may fail under adverse weather conditions due to repetitive textures and partial occlusions caused by rain and snow particles or textureless objects under foggy weather or at night. To overcome this issue and properly recover the camera motion in V, we propose estimating parameters t_x, t_y, θ, and s of an affine transformation matrix

$$\mathbf{A} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \end{bmatrix} \tag{6.1}$$

for every consecutive pair of frames using synthetic data for deep estimation. Thus, we abdicate the feature extraction procedure entirely since, using our proposed engine, we can generate the ground-truth affine transformation needed for training as described in Section 6.3.

We use two identical networks for estimating the parameters, where $W = H = 256$ is the center-cropped image width and height:

- $f_{tr} : \mathbb{R}^{4\times W\times H} \rightarrow \mathbb{R}^2$, which estimates the x and y translations $\mathbf{x}^{tr} = [t_x, t_y]$,
- $f_{rs} : \mathbb{R}^{4\times W\times H} \rightarrow \mathbb{R}^2$, which predicts the rotation angle and scale $\mathbf{x}^{rs} = [\theta, s]$.

Note that f_{tr} and f_{rs} share the same architecture but not the same weights. Both networks consist of a feature extractor implemented as four convolutional layers, a pooling and a dropout layer, and a regressor, which is a fully connected network composed of three linear layers that process the extracted features to estimate the parameters. Figure 6.1-a shows the number of output channels of each layer. For each training step, we feed the networks with an input $I = [v_i; v_{i+1}; O_i] \in \mathbb{R}^{4 \times W \times H}$, where $v_i, v_{i+1} \in \mathbb{R}^{W \times H}$ are two consecutive grayscale frames from the input video V and $O_i \in \mathbb{R}^{2 \times W \times H}$ is the dense Optical Flow map for the pair (v_i, v_{i+1}). Then, we estimate the parameters for $\mathbf{A}_i$ as $\hat{\mathbf{x}}^{tr} = f_{tr}(I)$ and $\hat{\mathbf{x}}^{rs} = f_{rs}(I)$. To optimize the networks f_{tr} and f_{rs}, we train separately each one using the MSE loss.

A key contribution of this chapter is the usage of specially designed synthetic data to learn an affine transformation matrix. Let $\mathcal{P}_i = \{\mathbf{p}_1, \dots, \mathbf{p}_K\}$ denote the 2D coordinates of K marked points at the frame v_i from a generated synthetic video. Since we can control the camera motion during the synthetic video generation, we can analytically determine the new positions of the marked points in frame v_i as they transition to frame v_{i+1}. With these $2K$ points in frames v_i and v_{i+1}, we can compute an affine transformation $\mathbf{A}_i$ with 4 degrees of freedom using $\mathcal{P}_i$ and $\mathcal{P}_{i+1}$, then use it as the ground truth for training f_{tr} and f_{rs}. We detail the process of ground truth generation in Section 6.3.

Finally, with $\hat{\mathbf{x}} = [\hat{\mathbf{x}}^{tr}, \hat{\mathbf{x}}^{rs}] = [\hat{t}_x, \hat{t}_y, \hat{\theta}, \hat{s}]$, the estimated parameters for each video frame pair, we compute the estimated camera trajectory $\hat{T} = \{\hat{\tau}_1, \dots, \hat{\tau}_{N-1}\}$, where

$$\hat{\tau}_i = \sum_{j=1}^{i} \hat{\mathbf{x}}_j \tag{6.2}$$

and $\hat{\mathbf{x}}_j$ represents the estimated parameters for the pair of frames (v_j, v_{j+1}). It is important to note that, similar to Grundmann, Kwatra, and Essa (2011), we do not directly use $\hat{t}_i$ to warp the shaky images. We warp the frames by applying the smoothed affine transformations composed of the smoothed translation, rotation, and scale parameters.

6.2.2 Trajectory smoothing

The next step after estimating the shaky camera trajectory is smoothing it. Unlike other methods, which tackle the camera trajectory smoothing as an optimization problem (Grundmann, Kwatra, and Essa, 2011), we deploy the Savitzky-Golay filter (Savitzky and Golay, 1964) on the averaged envelop of the shaky camera trajectory to smooth it, as described in the sequel.

Given the camera trajectory $\hat{T}$, we first calculate the extremes of $\hat{T}$ by applying the first-order discrete derivative. Then, we interpolate the trajectory maxima ($\hat{T}_{max}$) and minima ($\hat{T}_{min}$) values to extract the upper and lower envelopes, respectively. Quadratic interpolation presented the best results in our experiments since it makes smooth interpolations and tends to stay within the ranges of the interpolation points. The final upper and lower signal envelopes are represented as $E_{up} = \{e_1^{up}, \ldots, e_{N-1}^{up}\}$ and $E_{low} = \{e_1^{low}, \ldots, e_{N-1}^{low}\}$, respectively.

After obtaining the envelopes, we apply the Savitzky-Golay filter on the average envelop $\bar{E} = (E_{up} + E_{low})/2$ to remove the unwanted sudden camera shakiness and create the smooth camera trajectory $\tilde{T} = \{\tilde{t}_1, \ldots, \tilde{t}_{N-1}\}$ as shown in Figure 6.1-b. The Savitzky-Golay filter smooths the digital signal by fitting a low-degree polynomial to consecutive signal points using linear least squares. This strategy has an advantage over other techniques as it preserves the signal tendency. Thus, $\tilde{T}$ still maintains the properties of $\hat{T}$ while ensuring a smooth camera transition over time. After that, we calculate the difference between both trajectories $\delta_T = \hat{T} - \tilde{T}$. Then, the smoothed affine transformation parameters $\tilde{X}$ are calculated as $\tilde{X} = \hat{X} - \delta_T$, where $\tilde{X} = \{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{N-1}\}$ and $\hat{X} = \{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_{N-1}\}$, with $\tilde{\mathbf{x}}_i$ being the smoothed parameters $\tilde{t}_x$, $\tilde{t}_y$, $\tilde{\theta}$, and $\tilde{s}$.

At last, we warp and crop the video frames to compose the final video. For each video frame v_i, we compute its warped version $\tilde{v}_i$ by applying a transformation matrix to every pixel. Formally, we retrieve $\tilde{\mathbf{x}}_i$ from the smoothed transformations $\tilde{\mathbf{X}}$ and use the smoothed parameters $\tilde{t}_x$, $\tilde{t}_y$, $\tilde{\theta}$, and $\tilde{s}$ to compose the smoothed affine matrix $\tilde{\mathbf{A}}_i$. Then, we crop the warped frames using a predefined virtual cropping

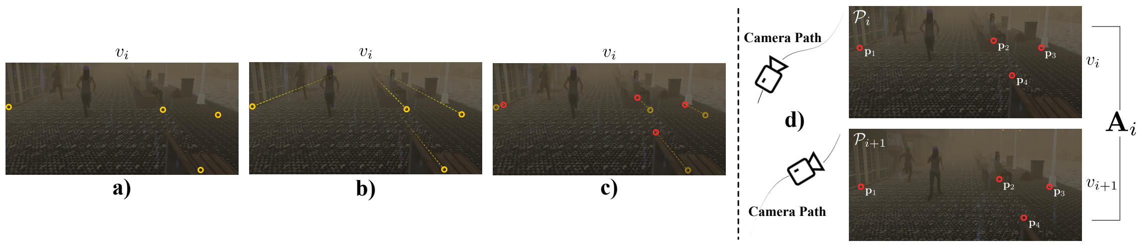


Figure 6.2: **Ground-truth Generation.** K points are randomly sampled from the screen space (yellow circles in a). From each of these points, we cast rays to infinity in the 3D scene space (dashed yellow lines in b) and create hypothetical objects at the intersection of these rays with the scene (red circles in c). We obtain the affine transformation matrix A_i using the coordinates of the hypothetical objects in screen space $\mathcal{P}_i$ and $\mathcal{P}_{i+1}$ since they remain stationary in the scene from the frame v_i to v_{i+1}.

window similar to Grundmann, Kwatra, and Essa (2011) to generate the stabilized video.

6.3 Synthetic Data and Ground truth Generation

6.3.1 Synthetic data generation

There are many synthetic data generators like CARLA (Dosovitskiy, Ros, et al., 2017) and UrbanScene3D (Yilin Liu, Xue, and H. Huang, 2021) that simulate photo-realistic and diverse 3D worlds in the literature. However, generating special data in such engines is cumbersome, and they do not support video stabilization. Therefore, in this chapter, we introduce a new extension to our synthetic data generator which was presented in Chapter 4. This extension to Silver makes it capable of filling this gap and creating the required training data for this task.

Our generator supports other computer vision tasks like semantic and instance segmentation, depth, and pose estimation. However, this chapter focuses on its usability for the video stabilization task. We show that more vital than photo-

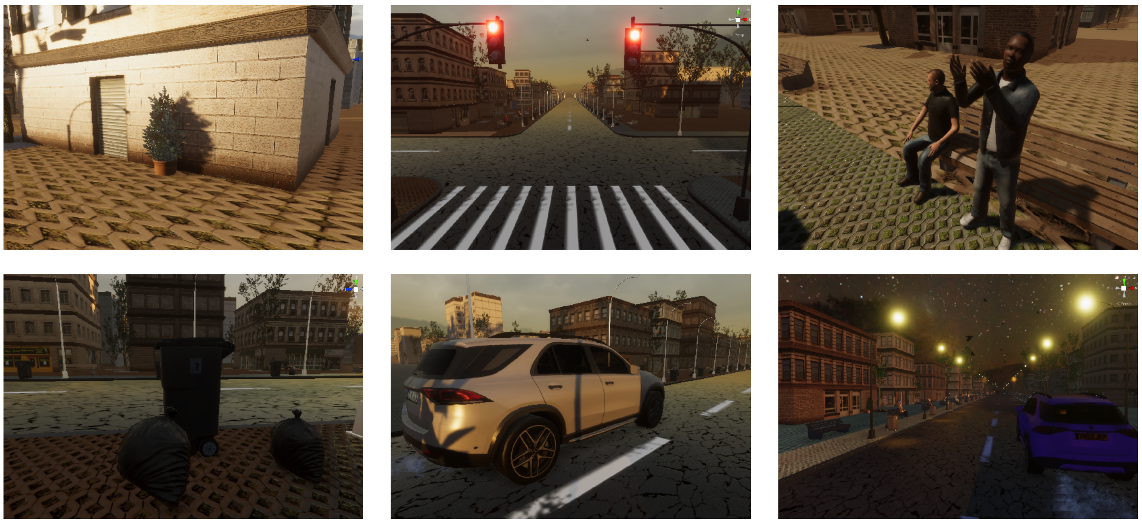


Figure 6.3: Samples from the procedurally generated scenes using the proposed synthetic data generator.

realistic and diverse 3D scenes is designing computer vision models targeted at using synthetic data and generating the appropriate data for these models. We control scene aspects in virtual worlds and generate more suitable training data for supervised learning algorithms. A shaky synthetic video is recorded after procedurally creating a 3D virtual world sampled from a predefined set of 3D models, materials, and animations. Note that for each video, a new virtual world is created to diversify the training data. Figure 6.3 demonstrates examples of the generated scenes.

To introduce shakiness to the recording camera, we create noise using a predefined noise profile asset. The amplitude and frequency of the noise are randomly sampled from a uniform distribution. The noise is applied to change the translation and rotation of the recording agent camera.

6.3.2 Ground-truth generation

The goal of our networks proposed in Section 6.2.1 is to infer the affine transformation matrix, **A**, given two consecutive frames. Since finding, collecting, and annotating data is a complex and expensive process, we generate and use synthetic videos to obtain the ground truth affine transformation matrices to supervise the training process.

Our idea works as follows: we first create stationary hypothetical labeled objects in the 3D world scene; then, we record their coordinates in the screen space of the recording camera. In that way, we can guarantee that the coordinates of these objects in frames (v_i, v_{i+1}) correspond to exactly the same static elements in the 3D world seen by the recording camera at frames v_i and v_{i+1}.

In other words, we create a number of invisible objects and save their coordinates in the camera space for each frame. For each frame, a number of random points are sampled from the screen camera space. Then, a ray is cast from each of these points to infinity. At each ray's intersection point with the scene, a hypothetical invisible object is created. The object remains stationary for a number of seconds before being destroyed. For each frame, the object's position in world space is transferred to the camera space. Figure 6.2 demonstrates how these hypothetical objects are created. If the hypothetical object is not in the camera view, or if it exceeds its time limit duration β, it is removed. Each object is given a Unique Identifier (UID) over its lifetime. Later in the post-processing stage, for every two consecutive frames (v_i, v_{i+1}) using the UIDs of these hypothetical objects and their screen locations, we calculate the ground-truth affine transformation for each pair. The process is further clarified as described in Algorithm 1. Generating myriads of hypothetical objects is possible. However, our algorithm provides a solution to create only objects in the field view of the recording agent, improving the overall performance.

Algorithm 1: Affine Transformation Ground-truth Generation

```
Require: $N_{frames}$, $T$
Ensure : Per frame file containing camera screen locations of stationary
         hypothetical objects.
$N_{frames}$;                              ▷ Total number of frames to generate
$T$;                                       ▷ Sampling period
while Recording do
    for $ID_{frame} = 0$; $ID_{frame} < N_{frames}$; $ID_{frame}$++ do
        if $ID_{frame} \% T == 0$ then
            $Points \leftarrow samplePoints$(K);   ▷ Sampling $K$ points on camera screen
             space
            foreach $point \in Points$ do
                $M \leftarrow castRayToInfinity$(point);   ▷ Cast a ray from point to
                 infinity
                $point_{inters} \leftarrow findIntersPoint(M)$;   ▷ Intersection point between
                 the ray with scene objects
                $O \leftarrow createHypotheticalObject(point_{inters})$;
                $O.UID \leftarrow assignObjectUID(O)$;   ▷ Assign $O$ a unique identifier
                $O.ScreenPos \leftarrow Cam.WorldToScreen(O.WorldPos)$;   ▷ Transfer
                 coordinates from world to screen space
                while O is visible and within its lifetime do
                    $Save(O.ScreenPos)$;
                    $WaitFewFrames()$;
                $Destroy(O)$;
        else
            $WaitFewFrames()$;
```

6.4 Experiments

In this section, we present the experimental setup used to conduct the experiments and the comparison results.

6.4.1 Experimental setup

6.4.1.1 Synthetic datasets

Using our proposed generator, we create two synthetic training datasets: VSNC35Synth and VSAC65Synth.

- VSNC35Synth dataset includes 35 videos at 24 fps with an average of 400 frames per video; it covers only videos in normal weather conditions. The average number of frames was set to 400 to match the number of frames in other datasets;
- VSAC65Synth dataset consists of 65 videos, including normal and adverse weather condition videos. The classes span normal, rainy, foggy, and snowy weather conditions at daytime and nighttime.

6.4.1.2 Real dataset

The available video stabilization benchmarks such as DeepStab (M. Wang et al., 2018), Stabfr (L. Zhang et al., 2018), Selfie Video (J. Yu and Ramamoorthi, 2018), and S. Liu, L. Yuan, et al. (2013) exclusively contain videos under normal weather condition and at a sufficient illumination. To assess the performance of the state-of-the-art video stabilization methods under foggy, rainy, snowy, and nighttime conditions, we created the VSAC105Real dataset. Our dataset is composed of videos collected from YouTube using search queries like “Fog”, “Rain”, “Snow”, “Night”, “Adverse”, and “Severe”. We manually inspected all the videos and selected the ones with shaking camera movement. Then, we cut the videos to

Table 6.1: **Dataset statistics.** Comparison among the available video stabilization datasets and VSAC105Real dataset.

Dataset Name	#Videos	Average #Frames	Total #Frames
DeepStab (M. Wang et al., 2018)	61	714	43,585
Stabfr (L. Zhang et al., 2018)	45	471	21,200
Selfie Video (J. Yu and Ramamoorthi, 2018)	33	251	8,308
S. Liu, L. Yuan, et al. (2013)	144	578	83,257
VSAC105Real	105	737	77,477

ensure continuous temporal and query attributes. VSAC105Real dataset comprises 105 videos spanning normal, rainy, foggy, snowy, and nighttime attributes.

Table 6.1 shows a comparison among different video stabilization datasets and VSAC105Real dataset. The VSAC105Real dataset has the advantage in terms of the average number of frames. Moreover, it includes a diverse set of challenging attributes where videos are evenly distributed across the classes, i.e., 21 videos per class. Furthermore, a visual comparison among VSAC105Real and other video stabilization datasets is depicted in Figure 6.4.

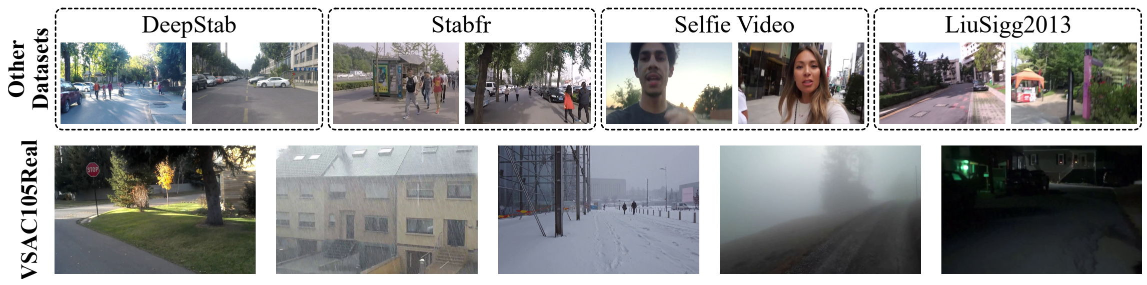


Figure 6.4: **VSAC105Real versus other datasets.** On top, each dashed box shows frames from other datasets. Our proposed dataset, VSAC105Real, is at the bottom. It includes more diverse and challenging attributes as compared to the other datasets.

6.4.1.3 Evaluation metrics

To evaluate our approach, we use three metrics commonly used to evaluate video stabilization algorithms (Y.-L. Liu et al., 2021; Choi and Kweon, 2020; M. Wang et al., 2018; J. Yu and Ramamoorthi, 2020) and a new proposed one:

- Stability score. It assesses the smoothness of the stabilized video; the higher the value the better. It is computed as the average between Stability Average Translation and Stability Average Rotation Scores. To compute this score, we estimate the homography matrix between v_i and v_{i+1} to obtain the translation and rotation arrays. Following this, we calculate their Fast Fourier Transform (FFT). Finally, we obtain the score by calculating the ratio between the 2nd through 6th frequency components and all frequency components. Note that the 0th frequency component is neglected;

- Distortion score. It measures the global distortion caused by a given video stabilization method. It fits a homography matrix between the original and stabilized videos. Then, it finds the anisotropic scaling among these frames; the closer to one, the better;

- Cropping ratio. It describes the ratio of the remaining frame's area after stabilization to the original one;

- Success rate. We also measure the success rate, which computes the ratio of videos that were successfully processed and yielded a distortion score lower than or equal to one.

6.4.1.4 Baselines

We evaluate five state-of-the-art video stabilizers on two real datasets: VSAC105Real and Selfie Video (J. Yu and Ramamoorthi, 2018). The baselines span non-learning based (Grundmann, Kwatra, and Essa, 2011), supervised (StabNet (M. Wang et al., 2018)), and unsupervised (DIFRINT (Choi and Kweon, 2020)) video stabilization

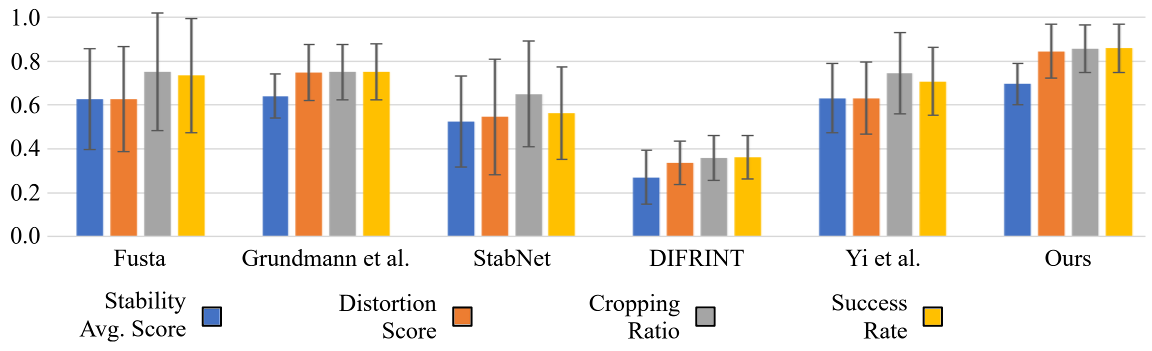


Figure 6.5: **Comparison across different weather conditions in the VSAC105Real dataset.**

methods. Furthermore, we compare our method to J. Yu and Ramamoorthi (2020), which heavily relies on optical flow since we use optical flow, and FuSta (Y.-L. Liu et al., 2021) because it also uses CNNs for video stabilization close to our approach.

6.4.1.5 Implementation details

We trained our method using only the synthetic data provided by our proposed simulator, i.e., VSNC35Synth. The hypothetical object's time limit duration was empirically set to $\beta = 1$ second. We trained the t_x and t_y translation prediction model (f_{tr}) and rotation θ and scale s prediction model (f_{rs}) for 65 and 2 epochs, respectively, using batches of size $M = 40$. After 10 epochs, we decrease the learning rate of f_{tr} to 1e−5.

Our architecture is fully implemented in PyTorch, and the training procedure takes 33 hours on a Tesla V100 GPU. For the smoothing step, a window length, i.e., number of coefficients, equal to 51 with 1$^{\text{st}}$ order polynomial were used as parameters to the Savitzky-Golay filter as they give better results. We used FlowNet2 (Ilg et al., 2017; Reda et al., 2017) as the optical flow estimator.

6.4.2 Results

Figure 6.5 and Table 6.2 show the comparison among the competitors across different weather conditions in the VSAC105Real dataset. Our method presented the best values, on average, in terms of stability average and distortion scores, cropping ratio, and success rate. In this thesis, we argue that our video stabilization model's superiority relates to the accurate affine transformation matrix estimation and the smoothing stage. Our algorithm preserves the frame content while compensating for camera shakiness.

We should highlight that our method achieved the highest success rate compared to the competitors. All baselines failed to stabilize most of the shaky videos in foggy weather conditions due to the nature of participating media. Fog works as a low-pass filter that removes high-quality features. Most video stabilizers depend on resilient features to estimate the camera trajectory. Even though our model was not trained on foggy weather conditions, it learned useful features from both raw images and optical flow.

Table 6.2 shows the results of comparing our method to several video stabilization approaches. As can be seen, our method presented the best values on average in comparison to all the baselines in terms of stability average, distortion, cropping ratio, and success rate. Even though our method did not surpass the baselines at each class individually, it still achieved competitive results. Every baseline performs badly in at least one class, while our method is more robust across classes, hence holding the final best results on the VSAC105Real.

Preserving the content while compensating for camera shakiness is another important feature of our algorithm. Our method achieved the best results as compared to other state-of-the-art methods. The superiority of our method is linked to the accurate affine transformation matrix estimation and the smoothing stage.

Table 6.2: **Comparison across different weather conditions in the VSAC105Real dataset.** Our method presents the best average values in comparison to the other competitors for all metrics. **Bold** indicates the best and <u>underline</u> second best.

Metric		**Method**	**Weather Condition**					**Average**
			Fog	Night	Normal	Rain	Snow	
Stability	Avg. Score ↑	FuSta (Y.-L. Liu et al., 2021)	0.226	0.683	0.715	0.679	0.824	0.626 ±0.231
		Grundmann, Kwatra, and Essa (2011)	0.642	0.549	0.620	0.580	0.809	<u>0.640</u> ±0.101
		StabNet (M. Wang et al., 2018)	0.201	0.469	0.620	0.577	0.753	0.524 ±0.207
		DIFRINT (Choi and Kweon, 2020)	0.121	0.212	0.321	0.247	0.446	0.270 ±0.122
		J. Yu and Ramamoorthi (2020)	0.401	0.682	0.665	0.572	0.834	0.631 ±0.159
		Our Model	0.606	0.619	0.728	0.687	0.835	**0.695** ±0.093
Distortion	Score *	FuSta (Y.-L. Liu et al., 2021)	0.202	0.692	0.725	0.712	0.798	0.626 ±0.240
		Grundmann, Kwatra, and Essa (2011)	0.740	0.617	0.762	0.667	0.952	<u>0.748</u> ±0.128
		StabNet (M. Wang et al., 2018)	0.111	0.518	0.790	0.597	0.710	0.545 ±0.264
		DIFRINT (Choi and Kweon, 2020)	0.367	0.219	0.351	0.270	0.476	0.337 ±0.099
		J. Yu and Ramamoorthi (2020)	0.372	0.729	0.654	0.593	0.804	0.631 ±0.165
		Our Model	0.719	0.746	0.952	0.809	0.997	**0.845** ±0.124
Cropping	Ratio ↑	FuSta (Y.-L. Liu et al., 2021)	0.286	0.810	0.905	0.800	0.950	<u>0.751</u> ±0.267
		Grundmann, Kwatra, and Essa (2011)	0.759	0.618	0.760	0.663	0.948	0.750 ±0.127
		StabNet (M. Wang et al., 2018)	0.278	0.579	0.875	0.667	0.850	0.650 ±0.242
		DIFRINT (Choi and Kweon, 2020)	0.399	0.234	0.392	0.280	0.490	0.359 ±0.102
		J. Yu and Ramamoorthi (2020)	0.476	0.810	0.842	0.650	0.947	0.745 ±0.184
		Our Model	0.762	0.760	0.945	0.820	0.999	**0.857** ±0.109
Success	Rate ↑	FuSta (Y.-L. Liu et al., 2021)	0.280	0.816	0.905	0.762	0.905	0.734 ±0.261
		Grundmann, Kwatra, and Essa (2011)	0.762	0.619	0.762	0.667	0.952	<u>0.752</u> ±0.128
		StabNet (M. Wang et al., 2018)	0.238	0.524	0.667	0.571	0.810	0.562 ±0.212
		DIFRINT (Choi and Kweon, 2020)	0.429	0.238	0.381	0.286	0.476	0.362 ±0.099
		J. Yu and Ramamoorthi (2020)	0.480	0.815	0.762	0.619	0.857	0.707 ±0.155
		Our Model	0.765	0.762	0.950	0.814	1.000	**0.858** ±0.110

↑*Higher is better* **Better closer to* 1

Table 6.3: **Comparison in the Selfie Video dataset (J. Yu** and **Ramamoorthi, 2018)**. **Bold** indicates the best, <u>underline</u> second best, and *italic* the third.

Method	Stability Avg. Score↑	Distortion Score∗	Cropping Ratio↑	Success Rate↑
FuSta (Y.-L. Liu et al., 2021)	<u>0.818</u>	*0.777*	**0.970**	**0.970**
Grundmann, Kwatra, and Essa (2011)	0.727	<u>0.828</u>	0.848	0.848
StabNet (M. Wang et al., 2018)	0.763	0.680	*0.917*	0.667
DIFRINT (Choi and Kweon, 2020)	**0.827**	0.691	0.912	*0.915*
J. Yu and Ramamoorthi (2020)	0.770	0.739	0.909	0.909
Our Model	*0.787*	**0.933**	<u>0.939</u>	<u>0.939</u>

↑*Higher is better* ∗*Better closer to* 1

We also evaluate our model on the Selfie Video dataset (J. Yu and Ramamoorthi, 2018), which contains videos under normal weather conditions and standard illumination. Results are presented in Table 6.3. Even though our model was trained from scratch solely on our synthetic data, it achieved the best distortion score.

6.4.3 Ablation study

We analyzed the design options and showed the effectiveness of each component of our pipeline. The results are reported in Table 6.4.

First, we assess a variant that uses a single CNN instead of two, as we proposed in our final model. As a result, the network was unable to converge well. One problem could be that the translation losses were larger than others. However, even with loss weighting applied, the network could not learn well (*Single Network* row). We hypothesize that the reason is the large value range between translation (i.e., 0 to image height/width) and scale (i.e., 0 to 1). Even with normalization, a single network struggled to back-propagate a meaningful error.

To emphasize the advantages of using our learning-based model for affine

Table 6.4: **Ablation study.** Performance for different design choices (best in **bold**).

Variant	Stability Avg. Score↑	Distortion Score∗	Cropping Ratio↑	Success Rate↑
Single Network	0.443	0.540	0.557	0.543
SIFT	0.576	0.650	0.670	0.667
No Optical Flow	0.678	0.781	0.829	0.800
Directed Smoothing	0.675	0.828	0.844	0.838
More Data	0.690	0.793	0.850	0.829
Complete Model	**0.695**	**0.845**	**0.857**	**0.840**

↑*Higher is better* ∗*Better closer to* 1

transformation matrix estimation over applying SIFT, we apply SIFT to find the affine transformation matrix while keeping the smoothing part of our model intact. As expected, SIFT did not perform well (*SIFT* row). Standard feature extractors like SIFT struggle to extract robust features under adverse conditions. Rain and snow particles, low illumination at night, and foggy weather make finding and matching features rather hard. That leads to inaccurate affine transformations and, thus, low-quality stabilized videos.

To evaluate our proposed smoothing algorithm's advantages over l_1 directed smoothing, as done in Grundmann, Kwatra, and Essa (2011), we apply l_1 directed smoothing on the predicted camera trajectory while keeping our learning-based model for the affine transformation matrix estimation. As expected, the model does not perform very well as compared to using our proposed smoothing algorithm (*Directed Smoothing* row) because our model considers more sophisticated camera paths and is not limited to constant, linear, and parabolic motions like Grundmann, Kwatra, and Essa (2011).

To highlight the importance of optical flow in affine transformation learning, we train a variant using only grayscale images, i.e., $I = [v_i; v_{i+1}]$. As expected, excluding optical flow reduced video stabilization quality (see *No Optical Flow* row).

To study the impact of our training synthetic data on video stabilization quality,

Table 6.5: **Affine matrix estimation**. Comparison among different methods for affine matrix estimation on CA-Unsupervised dataset (Jirong Zhang et al., 2020) using l_2 distance (best in **bold**).

Method	RT↓	LT↓	LL↓	SF↓	LF↓	Average↓
ORB (Rublee et al., 2011) + RANSAC (Fischler and Bolles, 1981)	9.24	14.63	12.27	11.36	7.20	10.94
ORB (Rublee et al., 2011) + MAGSAC (Barath, Matas, and Noskova, 2019)	10.11	19.79	12.48	11.86	7.85	12.42
ORB (Rublee et al., 2011) + LMEDS	9.78	40.11	12.02	10.84	7.01	15.95
SIFT (Lowe, 2004) + RANSAC (Fischler and Bolles, 1981)	10.63	11.70	13.37	11.75	6.44	10.78
SIFT (Lowe, 2004) + MAGSAC (Barath, Matas, and Noskova, 2019)	10.75	10.97	12.99	11.09	6.35	10.43
SIFT (Lowe, 2004) + LMEDS	10.47	9.72	13.04	10.14	5.88	9.85
Supervised (DeTone, Malisiewicz, and Rabinovich, 2016)	8.39	9.33	8.63	10.29	5.92	8.51
Unsupervised (Jirong Zhang et al., 2020)	7.05	7.60	6.84	7.42	**3.84**	6.55
Our Model	**4.55**	**5.58**	**5.68**	**5.17**	9.73	**6.14**

↓ *Lower is better*

we trained our model on more data, including normal and adverse weather and nighttime videos. The training was performed using the VSAC65Synth dataset from scratch with no real data. The results indicated no significant improvement over the method trained on VSNC35Synth (*Complete Model* row). Therefore, a few synthetic videos with accurate ground truth are sufficient to train the model.

To investigate the accuracy of our estimated affine transformation matrix, we conduct experiments comparing our learning-based affine transformation estimation with several estimation approaches, including the traditional ones like ORB and SIFT with RANSAC, MAGSAC, and LMEDS for outliers rejection, the supervised, and the unsupervised ones. Traditional approaches estimate the affine transformation directly, but supervised and unsupervised methods are designed to estimate the homography matrix. Thus, we extract the affine transformation from their estimated homography for a fair comparison. We utilize the dataset of (Jirong Zhang et al., 2020), which contains 4,200 pairs of images where each image-pair includes six matching human-annotated pairs of points. The dataset covers:

- Regular texture (RT);
- Low texture (LT);
- Low light (LL);
- Small foregrounds (SF);
- Large foregrounds (LF).

We use l_2 distance to measure the error between the warped and ground-truth points similar to (Jirong Zhang et al., 2020; Ye et al., 2021).

Table 6.5 demonstrates the generalizability of our affine estimation model, which is better on both standard and challenging conditions. The dataset used in this experiment includes challenging images, such as low-texture images similar to images under foggy or snowy weather conditions and images at low illumination similar to the ones at nighttime. These results demonstrate the ability of our proposed affine estimator of learning to extract robust features under challenging conditions.

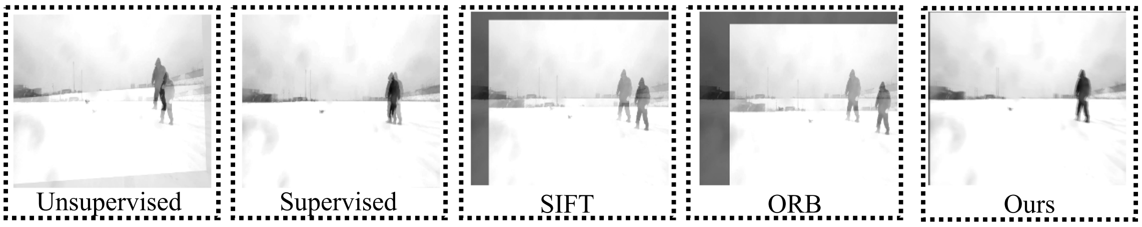


Figure 6.6: Qualitative comparison for affine transformation estimation.

Figure 6.6 shows a qualitative comparison among our model, traditional (e.g., SIFT and ORB), supervised (DeTone, Malisiewicz, and Rabinovich, 2016), and unsupervised (Jirong Zhang et al., 2020) methods, demonstrating examples of low texture pair of images. While other methods fail under such challenging conditions because they were manually tuned for standard settings or not trained with data under adverse conditions, our method performs well because it learned how to extract resilient features using our specially designed synthetic data.

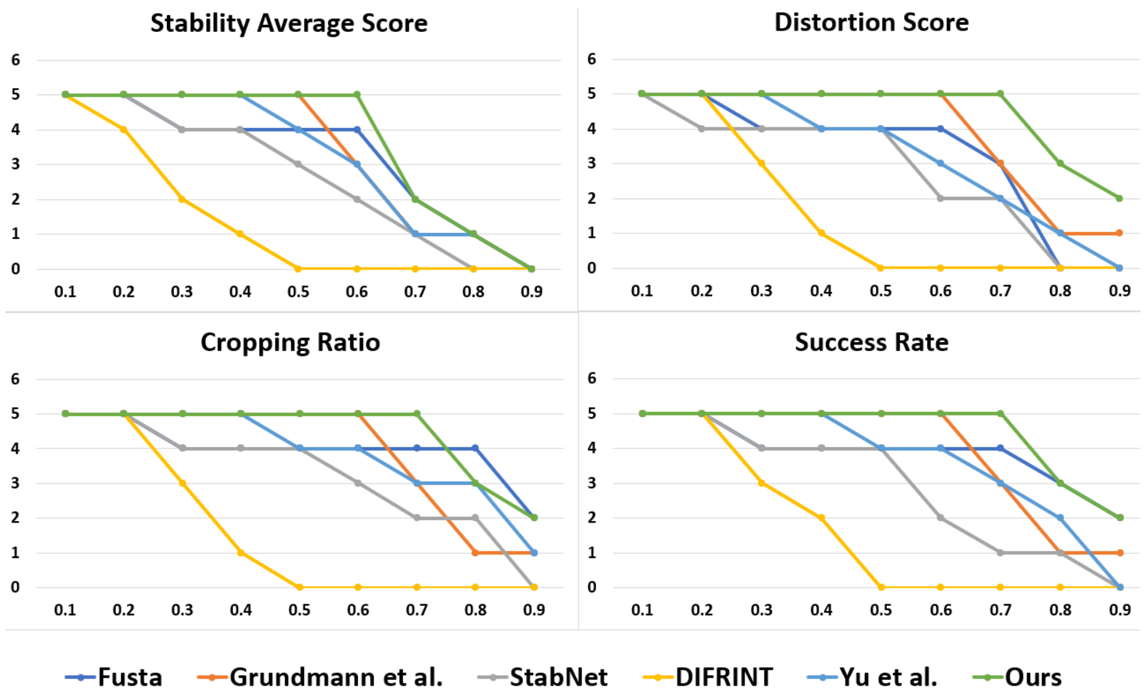


Figure 6.7: **Classes coverage by threshold.** Number of times the metrics values for classes in Figure 6.5 are above each threshold. Our method presents the highest metrics coverage, achieving the best results compared to other video stabilization methods.

Additionally, we present in Figure 6.7 a new metric that computes the coverage of classes. The figure shows the number of classes (i.e., fog, night, normal, rain, and snow) whose output values achieved a result above different thresholds (i.e., 0.1 to 0.9 with a 0.1 step). As expected, our method presents the highest coverage as the threshold values increase, achieving the best results compared to other video stabilization methods.

6.4.4 Computational time analysis

Our proposed method requires two image frames sampled closely enough to accurately estimate the camera trajectory $\hat{T}$ given a shaky video $V = \{v_1, v_2, \ldots, v_N\}$. Thus, to evaluate our proposed video stabilization's performance and computation cost under a low frame rate, we performed the following set of experiments. We analysed the performance of our proposed video stabilization model under three frame rates:

- High-frame rate;
- Mid-frame rate;
- Low-frame rate.

6.4.4.1 High-frame rate

This is the original setup of our video stabilizer. It is assumed that the videos are captured at 24 to 30 fps. Under this setup, we sample every 2 consecutive frames, i.e., v_i and v_{i+1} where $i = 1, 2, 3, \ldots, N - 1$, and then we generate the optical flow. Then, we estimate the affine transformation and smooth the predicted camera trajectory $\hat{T}$ as explained earlier.

6.4.4.2 Mid-frame rate

For every 4 frames from the shaky video V, we skip three frames, i.e., we sample v_i and v_{i+4} where $i = 1, 5, 9, 13, \ldots, N - 4$. Then, we calculate the optical flow for the frames v_i and v_{i+4}. After that, we estimate the affine transformation and smooth the predicted camera trajectory.

6.4.4.3 Low-frame rate

Similarly, we skip 7 frames for every 8 frames from the shaky video. We sample v_i and v_{i+8} where $i = 1, 9, 17, 25, \ldots, N - 1$. Then, we repeat the steps mentioned earlier.

Table 6.6 shows the results on VSAC105Real dataset. As expected, performing video stabilization at lower frame rates requires fewer computations. The computation complexity comes from two main factors: optical flow and affine

Table 6.6: **Computational Time Analysis:** Higher sampling rate gives more stable videos but requires more computations.

	2 consecutive frames	1 at every 4 frames	1 at every 8 frames
Stability Avg. Score ↑	**0.695**	0.505	0.420
Distortion Score ∗	**0.845**	0.784	0.640
Avg. Time per Frame (Sec) ↓	0.022	0.010	**0.007**

↑Higher is better ↓Lower is better ∗Better closer to 1

transformation estimations. For a video of 10 frames, 9, 2, and 1 estimation(s) are required for optical flow and affine transformation for high, mid, and low frame rates, respectively.

While the computation cost is reduced with lower frames, the quality of the stabilized videos degrades. Lower frame rates make the camera path estimation noisier and, thus, the final stabilized video shakier and more distorted. It should be noted that for the main problem addressed in this chapter, 24 to 30 fps is the standard frame rate. Considering different frame rates may not be suitable given the dynamics of the scene.

6.5 Discussion

The results demonstrate the advantages of using synthetic data with a specially designed ground truth and architecture. In this chapter, we argue that the main factors behind achieving good results using only a small amount of synthetic data are:

- Accurate ground truth;
- High quality images and plausible scene composition;
- Diversity;
- Video stabilization algorithm.

Subsequently, we will delve into a detailed examination of each, providing a more in-depth discussion.

Accurate ground truth. Most supervised computer vision algorithms are trained using data collected and annotated manually for this task. However, video stabilization is more challenging as collecting ideal training data is not feasible. Some approaches utilize two cameras using a mechanical stabilizer to generate the required ground truth. The main issue is that the scene is captured by two different cameras and from two different view angles. Thus, the task of video stabilization becomes harder for the model to learn. Our novel approach for ground-truth generation achieves accurate ground truth. Thus, it helps learning video stabilization to converge. Please note that corrupt and noisy labels are well-known issues in computer vision (Song et al., 2022; P. Chen et al., 2019; Van Horn et al., 2015).

High quality images and plausible scene composition. Our synthetic data is composed of high quality image, where the scenes comprise plausible configurations. These two properties mitigate the domain gap between the synthetic and real domains. Thus, our model generalizes well on real videos.

Diversity. The key reason behind achieving good results using small size synthetic dataset can also be due to the diversity of the generated videos. The *Silver* simulator applies various domain randomization techniques, so the attributes of the generated scenes are highly diverse.

Video stabilization algorithm. Another key factor is using two separate networks and leveraging optical flow information to help with the video stabilization task.

6.6 Concluding Remarks

Recent video stabilization methods struggle under adverse conditions. In this chapter, we proposed a synthetic-aware video stabilization method that requires only synthetic data for training, surpassing all other baselines. We also provided one real dataset for video stabilization under adverse conditions and two synthetic datasets for training produced by our novel synthetic data generator.

Chapter 7

Usability of Synthetic Data

Usability is like love. You have to care, you have to listen, and you have to be willing to change. You'll make mistakes along the way, but that's where growth and forgiveness come in.

Jeffrey Zeldman

The work presented in this chapter has been submitted to CVPR-2024 (Kerim, Marcolino, et al., 2024). The paper is currently under review.

Supervised machine learning methods require large-scale training datasets to perform well in practice. Synthetic data has been showing great progress recently and has been used as a complement to real data. However, there is still a great urge to assess the usability of these synthetically generated data. For that aim, we propose a novel training procedure and usability metric that disentangles photorealism from diversity. We show that our metric is a simple yet effective way to rank synthetic images based on their usability. Furthermore, we propose a new pipeline for generating synthetic data by integrating Large Language Models with Stable Diffusion. The quantitative results show we can achieve similar or better results by training on 50% less synthetic data. Additionally, we quantitatively assess the impact of photorealism on synthetic data usability. We perform an extensive set of experiments by evaluating six different architectures on three different datasets to assess the effectiveness of our metric and approach.

7.1 Motivation

The advancement in ML has been mostly influenced and attributed to large-scale annotated training data availability. State-of-the-art computer vision models (J. Xu et al., 2022; Junayed et al., 2022; W. Wang et al., 2023) were usually trained on large-scale datasets, such as MS COCO (Lin et al., 2014), ADE20K (B. Zhou, Hang Zhao, Puig, Xiao, et al., 2019; B. Zhou, Hang Zhao, Puig, Fidler, et al., 2017), and ImageNet (Deng et al., 2009). However, collecting and annotating these large-scale datasets is cumbersome and time-consuming. Furthermore, the annotation process imposes privacy problems, ethical issues, and the human-labeled annotations can be biased and noisy.

Synthetic data generation has seen remarkable advances in recent years, thanks to methods such as Generative Adversarial Networks (Jin, F. Tan, S. Jiang, et al., 2020; Iglesias, Talavera, and Díaz-Álvarez, 2023; Kang et al., 2023), Diffusion Models (Croitoru et al., 2023; Carlini et al., 2023; Bansal et al., 2023), and simulators (Dosovitskiy, Ros, et al., 2017; S. Shah et al., 2018; C. Ma, Y. Zhou, and Zhiqiang Li, 2020). These methods enable the creation of diverse synthetic data that can complement or replace real data in various applications. Moreover, synthetic data can help overcome the challenges and limitations of real data, such as scarcity, privacy, bias, and cost. Although these approaches can generate large-scale datasets, the generated images are not always realistic and useful for training or testing machine learning models. While it is an easy task for humans to assess the photorealism and diversity of a small set of images, it is hard to assess the usability of a few synthetic images and rather hard for a large-scale dataset.

Visually appealing images are not necessarily effective to train models. Similarly, diverse synthetic images are not certainly useful images, too. A model trained on visually appealing images may struggle when faced with images that have variations

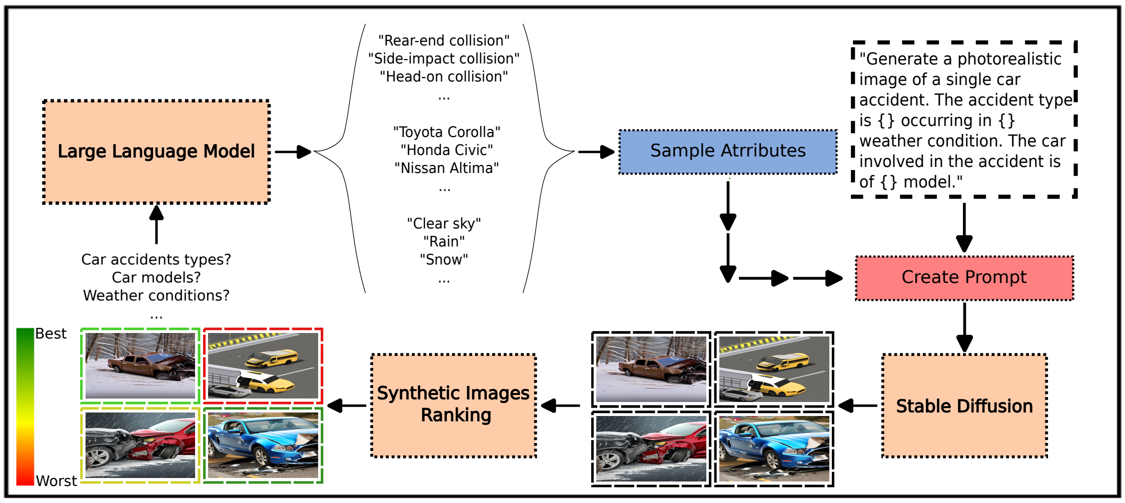


Figure 7.1: **Synthetic data generation pipeline.** Our method leverages LLM and stable diffusion to automatically generate diverse, photorealistic, and large-scale training data.

in lighting and camera parameters which are common in practical scenarios but may not be present in aesthetically pleasing images. In contrast, overly diverse synthetic images may introduce unrealistic variations that do not reflect the true distribution of real-world data. Thus, it can confuse the model, making it less effective in practice.

As noted by Katayama et al. (2022) and Man and Chahl (2022), a useful synthetic image is the one that is photorealistic and diverse. Photorealism is essential to bridge the domain gap with the real domain. Additionally, training machine learning models on diverse data is fundamental for improving the generalizability and robustness of these models in practice (Gong, Zhong, and W. Hu, 2019; S. Yang et al., 2022; Kariyappa and Qureshi, 2019).

In response, this chapter introduces a novel synthetic data generation pipeline and usability metric that untangles the essential components of photorealism and diversity for image classification problem. We address the significance of selecting the right images for training ML models, emphasizing the balance between photorealistic representation and diverse content.

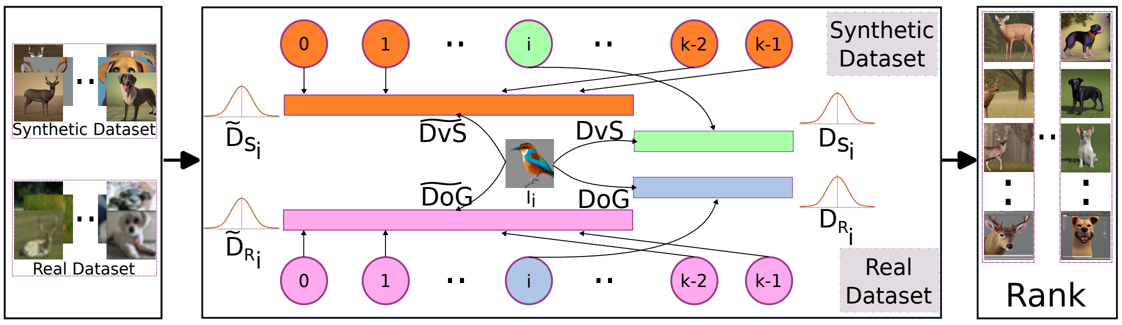


Figure 7.2: **Synthetic Images Ranking.** Each synthetic image I_i is compared to the real and synthetic images in its class i and to both real and synthetic datasets excluding its class images, then a score is assigned to the image based on Eq. 7.1.

Hence our main contributions are three-fold:

- A new pipeline for automatically generating synthetic data by integrating large language models with Stable Diffusion which offers a diverse and photorealistic collection of images suitable for effective ML models training;
- Quantitative evaluation of the impact of photorealism on the usability of synthetic images;
- A new metric for assessing the usability of synthetic images which captures the strengths and limitations of using them for training computer vision models.

7.2 Methodology

In our approach, We propose using Stable Diffusion (Rombach et al., 2022) with LLMs specifically Chat Generative Pre-trained Transformer (ChatGPT-3.5) (OpenAI, 2023) to generate synthetic training images. Then, we provide a metric to select the most useful ones. Figure 7.1 illustrates the main components of our approach. The modular nature of our approach allows the synthetic data generation pipeline and usability metric to be utilized separately.

7.2.1 Synthetic data generation pipeline

We propose a novel synthetic data generation pipeline that leverages the power of LLMs and DMs to create diverse and high-quality datasets. The pipeline consists of three phases.

I) Attribute extraction using LLM. LLMs are pre-trained on large-scale and diverse datasets, making them a good fit for feature extraction.

Thus, our methodology initiates by leveraging the advanced capabilities of LLMs to perform attribute extraction. Specifically, we employ LLM to extract the main attributes that are crucial and pivotal for the problem. For instance, when generating car accidents synthetic dataset, we employ LLM to identify relevant attributes, such as popular car colors, prevalent car models, and diverse weather conditions.

This process of attribution extraction provides us with a comprehensive pool of potential strong attributes that will be leveraged in the second phase of our pipeline.

II) Attribute sampling and prompt creation. To construct the final prompts for the DMs to generate the required images, we randomly sample attributes from the pre-existing pool of attributes acquired in the preceding phase. These sampled attributes are employed as input parameters to our DM prompt template, which is designed to guide the DM model in generating realistic and contextually relevant data. The random sampling process ensures diversity in the generated datasets, thereby making them more representative of real-world scenarios.

III) Data generation with DM. We leverage Stable Diffusion-V2 model (Rombach et al., 2022) to generate the required dataset tailored to this task-specific requirements using our carefully engineered prompt. The DM was trained on LAION-5B dataset (Schuhmann et al., 2022) which contains more than 5.85 billion image-text pairs to understand the context provided by the prompts and produce synthetic data that aligns with the specified attributes and conditions. This extensive training enables the DM to effectively comprehend the contextual details encoded in the prompts, facilitating the generation of synthetic data that is more useful and practical.

By leveraging the combined power of LLMs and DMs, we can efficiently generate high-quality data. This innovative and streamlined pipeline opens up new possibilities for training and evaluating machine learning models.

7.2.2 Synthetic data usability assessment

In addressing the challenges of evaluating synthetic data for model training, our motivation for introducing a new metric stems from the limitations of existing approaches, which often involve complex hyperparameter tuning and lack universality. Our proposed metric aims to streamline this process by offering a more accessible and widely applicable assessment tool. By considering the interplay of semantic context and visual fidelity, our metric responds to the evolving needs of the computer vision community, providing a comprehensive and relevant evaluation framework for the usability of synthetic data in model training.

The score S for each synthetic image is calculated as:

$$\mathcal{S} := \frac{DvS + \widetilde{DvS}}{DoG + \widetilde{DoG}}, \tag{7.1}$$

where DvS and $\widetilde{DvS}$ represent in-class and in-dataset diversity scores of the synthetic image, respectively. While DoG and $\widetilde{DoG}$ refer to the in-class and in-dataset synthetic-real domain gaps, respectively.

7.2.2.1 Diversity

To assess the diversity of the synthetic image, we use:

$$DvS(I_i) :=\mid \mu_i^s - \mu_{ci}^s \mid + Tr(\sigma_i^s + \sigma_{ci}^s - 2\sqrt{\sigma_i^s * \sigma_{ci}^s}), \tag{7.2}$$

where μ_i^s and σ_i^s refer to the mean and covariance of the synthetic, $(.)^s$, image I that belongs to class i; μ_{ci}^s and σ_{ci}^s refer to the mean and covariance of the set of all images in class i; Tr refers to the trace of the matrix.

In this context, we define $DvS(I_i)$ as the diversity score of the synthetic image I_i within its class. Specifically, we evaluate the diversity of I_i by comparing it to all other synthetic images within the same class. A higher score indicates greater diversity within the class, as formulated in Eq. 7.2. Our metric also ensures that the diversity of image I_i extends beyond its class, encompassing factors such as background, camera orientation, weather conditions, and other attributes not directly related to the object of interest as shown in Eq. 7.3.

$$\widetilde{DvS}(I_i) :=\mid \mu_i^s - \widetilde{\mu^s}_{ci} \mid + Tr(\sigma_i^s + \widetilde{\sigma^s}_{ci} - 2\sqrt{\sigma_i^s * \widetilde{\sigma^s}_{ci}}), \tag{7.3}$$

where $\widetilde{\mu_{ci}^s}$ and $\widetilde{\sigma_{ci}^s}$ refer to the mean and covariance of all images in all classes of the synthetic dataset except class i.

7.2.2.2 Photorealism and synthetic-to-real gap

Similarly, we evaluate the synthetic-to-real domain gap of each synthetic image based on the given real dataset. It is assumed that both synthetic and real datasets have an equal number of classes k.

$$DoG(I_i) :=\mid \mu_i^s - \mu_{ci}^r \mid + Tr(\sigma_i^s + \sigma_{ci}^r - 2\sqrt{\sigma_i^s * \sigma_{ci}^r}), \tag{7.4}$$

where μ_{ci}^r and σ_{ci}^r refer to the mean and covariance of all the real, $(.)^r$, images in class i.

$DoG(I_i)$ represents the domain-gap score of the synthetic image I_i in its class. This ensures that nonphotorealistic images will be given lower scores and thus less likely to be used in the training. Similarly, to assess how close the synthetic image I_i is to the real images in general including attributes such as background, lighting, and camera parameters, we define $\widetilde{DoG}$ as

$$\widetilde{DoG}(I_i) :=\mid \mu_i^s - \widetilde{\mu^r}_{ci} \mid + Tr(\sigma_i^s + \widetilde{\sigma^r}_{ci} - 2\sqrt{\sigma_i^s * \widetilde{\sigma^r}_{ci}}). \tag{7.5}$$

Table 7.1: Prompt templates used to guide the Stable Diffusion model (Rombach et al., 2022) to generate the synthetic datasets.

Dataset	*Prompt Template*
SP-Car-2	"Generate a photorealistic image of a single car accident. The accident type is {} occurring in {} weather condition. The car involved in the accident is of {} color and is {} model. Capture the scene with meticulous attention to detail, realism, and visual impact."
SA-Car-2	"Generate a highly stylized and non-photorealistic image of a single car accident. The accident type is {} occurring in {} weather condition. The car involved in the accident is of {} color and is {} model. Apply unique and exaggerated artistic effects, such as vibrant color splashes, abstract shapes, and bold brushstrokes."
SP-CIFAR-10	"Generate a photorealistic image of a single {} item. Capture the scene with meticulous attention to detail, realism, and visual impact."
SA-CIFAR-10	"Generate a highly stylized and non-photorealistic image of a single {} item. Apply unique and exaggerated artistic effects, such as vibrant color splashes, abstract shapes, and bold brushstrokes, to create an image that diverges significantly from traditional photographic realism. Emphasize the use of strong contrasts, surreal textures, and artistic distortions."
SP-Birds-525	"Generate a photorealistic image of a single {} bird. Capture the scene with meticulous attention to detail, realism, and visual impact."
SA-Birds-525	"Generate a highly stylized and non-photorealistic image of a single {} bird. Apply unique and exaggerated artistic effects, such as vibrant color splashes, abstract shapes, and bold brushstrokes, to create an image that diverges significantly from traditional photographic realism. Emphasize the use of strong contrasts, surreal textures, and artistic distortions."

7.2.3 Synthetic images ranking

In this chapter, we argue that carefully selecting training samples is more important than training on large-scale synthetic datasets. Few noisy training samples can drastically impact the overall performance of ML models (Al-Gethami, Al-Akhras, and Alawairdhi, 2021; Liang, X. Liu, and Yao, 2022; Karimi et al., 2020). Our metric helps to filter noisy and irrelevant synthetic images based on four aspects:

- Diversity in class;
- Diversity in dataset;
- Photorealism in class;
- Photorealism in dataset.

Figure 7.2 illustrates how each synthetic image is given a score and ranked using our metric shown in Eq. 7.1.

7.3 Prompts Engineering

To generate the synthetic datasets i.e., *SP-Car-2*, *SP-CIFAR-10*, *SP-Birds-525*, *SA-Car-2*, *SA-CIFAR-10*, and *SA-Birds-525*, we employed a set of carefully designed prompt templates to guide the Stable Diffusion model (Rombach et al., 2022) in generating the required images. These prompt templates are outlined in Table 7.1. Each template was carefully engineered to influence the synthesis process, ensuring diversity and adherence to desired visual properties. The versatility of our approach is evident in the range of datasets created, covering various domains, problems, and scenarios.

Leveraging ChatGPT-3.5 (OpenAI, 2023), we curated a comprehensive set of attributes that encapsulate diverse aspects of the problem and task. For instance, we present the attributes pool specifically designed for synthetic car accidents datasets (*SP-Car-2* and *SA-Car-2*) in Table 7.2. Please note that other attributes pools are similarly created using the same approach. These attributes play a pivotal role in simulating realistic, diverse, and challenging scenarios and guiding the Stable Diffusion model (Rombach et al., 2022) to capture these scenarios in the generated synthetic images for effectively training supervised computer vision models.

7.4 Experiments

We evaluate our approach through binary and multi-labeled image classification tasks, offering fundamental insights into the classification models' ability to handle complex scenarios with multiple categories when trained on synthetic data. The classification task is foundational in computer vision, providing a robust assessment of our synthetic data generation pipeline and usability metric.

Our approach is fully implemented in PyTorch on a Tesla V100 GPU. The data generation, training, and evaluation experiments took approximately 20 days using 15 GPUs.

Table 7.2: The attributes pool created by *ChatGPT* (OpenAI, 2023) for synthetic car accidents datasets (i.e., *SP-Car-2* and *SA-Car-2*).

Popular Car Models	*Popular Car Colors*	*Car Accident Types*		*Weather Conditions*
Toyota Corolla	White	Rear-end collision	Reckless driving accident	Clear sky
Honda Civic	Black	Side-impact collision (T-bone)	Failure to yield collision	Partly cloudy
Ford F-Series	Gray	Head-on collision	Improper lane change collision	Overcast
Chevrolet Silverado	Silver	Single-vehicle accident	Backing up collision	Cloudy
Toyota Camry	Blue	Multi-vehicle pileup	Animal-related accident	Rain
Honda Accord	Red	Intersection collision	Driver error accident	Thunderstorm
Nissan Altima	Brown	Parking lot collision	Mechanical failure accident	Snow
Volkswagen Golf	Green	Rollover	Construction zone accident	Fog
BMW 3 Series	Yellow	Run-off-road collision	Red light violation collision	Mist
Mercedes-Benz C-Class	Orange	Sideswipe collision	Wrong-way driving accident	Haze
Ford Mustang	Beige	Pedestrian-involved accident	Loss of control collision	Drizzle
Chevrolet Camaro	Bronze	Bicycle-involved accident	Emergency vehicle collision	Sleet
Toyota RAV4	Turquoise	Motorcycle-involved accident	Train crossing collision	Freezing rain
Jeep Wrangler	Maroon	Weather-related accident	School bus-involved accident	Tornado

7.4.1 Real datasets

To assess the performance of a wide range of image classifiers and evaluate the utility of our synthetic training data, we use three publicly available real datasets: Car Accidents-2 (*R-Car-2*), CIFAR-10 (Krizhevsky, G. Hinton, et al., 2009) (*R-CIFAR-10*), and Birds-525 (Piosenka, 2023) (*R-Birds-525*). We partition each dataset into 90% train-validate split and 10% test split.

7.4.2 Synthetic datasets

Using our synthetic data generation pipeline, we generate two sets of datasets: Photorealistic and Artistic. Artistic versions of the datasets were generated to assess the impact of photorealism on the utility of synthetic data. First, we generated three photorealistic datasets: Synthetic Photorealistic Car Accidents-2 (*SP-Car-2*), Synthetic Photorealistic CIFAR-10 (*SP-CIFAR-10*), and Synthetic Photorealistic Birds-525 (*SP-Birds-525*); second, three artistic datasets: Synthetic Artistic Car Accidents-2 (*SA-Car-2*), Synthetic Artistic CIFAR-10 (*SA-CIFAR-10*),

Table 7.3: Synthetic datasets statistics.

Synthetic Dataset Statistics				
Artistic	Photorealistic	# Classes	# Images	Resolution
SA-Car-2	*SP-Car-2*	2	844	768×768
SA-CIFAR-10	*SP-CIFAR-10*	10	$50,000$	768×768
SA-Birds-525	*SP-Birds-525*	525	$89,250$	768×768

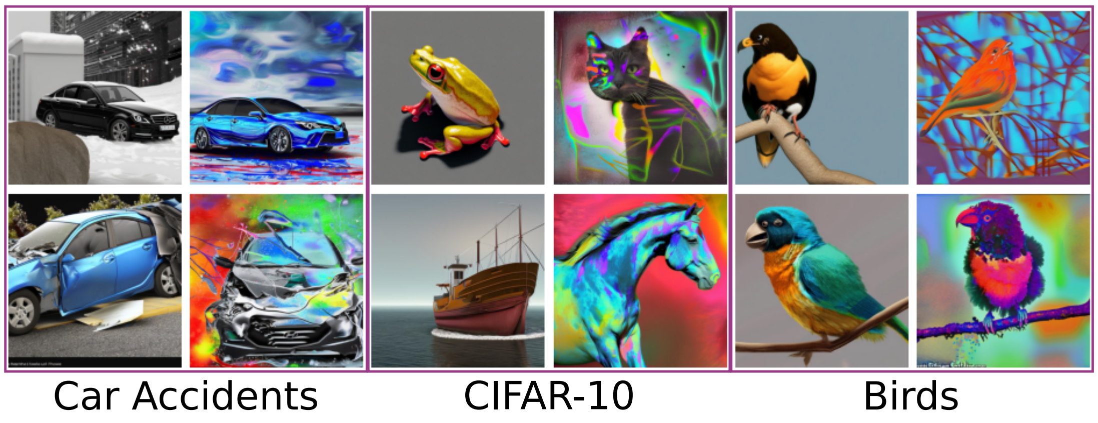


Figure 7.3: Samples from the synthetically generated datasets using stable diffusion. Photorealistic and artistic samples are shown in the first and second columns of each dataset, respectively.

and Synthetic Artistic Birds-525 (*SA-Birds-525*).

Photorealistic and artistic versions are identical in number of images per class and image resolution. Table 7.3 shows the number of classes, images, and resolutions. Furthermore, Figure 7.3 shows random samples from each dataset.

7.4.3 Image classification methods

We evaluate six prominent and state-of-the-art image classifiers spanning various architectures: *AlexNet* (Krizhevsky, Sutskever, and G. E. Hinton, 2012) which represents a simple and foundational architecture, *EfficientNet* (M. Tan and Le, 2019) which is an example of computationally efficient model, *ViT* (Dosovitskiy, Beyer, et al., 2020) and *SwinTansformer* (Z. Liu et al., 2021) which represent transformers, *VGG* (Simonyan and Zisserman, 2014) which represents a well-established baseline in the field, and *REGNet* (Radosavovic et al., 2020) which represents models with a focus on regularization techniques.

Table 7.4: Accuracy of six classification architectures across various training modes on three real datasets: *R-Car-2*, *R-CIFAR-10*, and *R-Birds-525*. *Synth** means the synthetic images were selected using our metric. The variable n denotes the total number of images employed in the fine-tuning experiments (best in **bold**).

	Model	*Synth(Artistic)*	*Synth(Photorealistic)*	*Real*	*Synth + Real*	*Synth** *+ Real*
		n	n	n	$0.5n + 0.5n$	$0.25n + 0.5n$
R-Car-2	*AlexNet*	87 (−4% ↓)	85 (−7% ↓)	**91**	88 (−3% ↓)	89 (−2% ↓)
	EfficientNet	84 (−10% ↓)	68 (−27% ↓)	93	94 (+1% ↑)	93 (——)
	ViT	86 (−4% ↓)	81 (−10% ↓)	90	91(+1% ↑)	93 (+3% ↑)
	SwinTansformer	86 (−5% ↓)	74 (−19% ↓)	91	92(+1% ↑)	92 (+1% ↑)
	VGG	91 (−3% ↓)	82 (−13% ↓)	94	**95** (+1% ↑)	94 (——)
	REGNet	87 (−7% ↓)	71 (−24% ↓)	94	96 (+2% ↑)	94 (——)
R-CIFAR-10	*AlexNet*	29 (−62% ↓)	21 (−73% ↓)	77	75 (−3% ↓)	76 (−1% ↓)
	EfficientNet	26 (−68% ↓)	37 (−54% ↓)	80	84 (+5% ↑)	83 (+4% ↑)
	ViT	69 (−24% ↓)	71 (−22% ↓)	91	90 (−1% ↓)	91 (——)
	SwinTansformer	54 (−40% ↓)	64 (−29% ↓)	90	89(−1% ↓)	90 (——)
	VGG	35 (−55% ↓)	38 (−51% ↓)	77	79(+3% ↑)	80 (+4% ↑)
	REGNet	31 (−60% ↓)	15 (−81% ↓)	78	83(+6% ↑)	84 (+8% ↑)
R-Birds-525	*AlexNet*	6 (−91% ↓)	13 (−80% ↓)	66	46 (−30% ↓)	68(+3% ↑)
	EfficientNet	19 (−80% ↓)	21 (−78% ↓)	94	87 (−7% ↓)	91 (−3% ↓)
	ViT	25 (−73% ↓)	34 (−64% ↓)	94	87 (−7% ↓)	92 (−2% ↓)
	SwinTansformer	17 (−82% ↓)	27 (−71% ↓)	93	88 (−5% ↓)	89 (−4% ↓)
	VGG	25 (−72% ↓)	31 (−66% ↓)	90	85 (−6% ↓)	86 (−4% ↓)
	REGNet	13 (−86% ↓)	20 (−78% ↓)	91	81 (−11% ↓)	89 (−2% ↓)

7.4.4 Results

7.4.4.1 Photorealism and synthetic data usability

We use two different prompt templates to generate our synthetic datasets. The first one guides the DM to generate artistic and nonrealistic synthetic images. The prompt used contains keywords such as:

- “Highly stylized”;
- “Non-photorealistic”;
- “Exaggerated artistic effects”;
- “Bold brushstrokes”.

In contrast, to generate the photorealistic images, we used keywords, such as:

- “Photorealistic”;
- “Meticulous attention to detail”;
- “Realism”.

The same attributes sampling process is followed for both photorealistic and artistic versions of the dataset.

To assess the photorealism of the generated images, we use the FID metric as shown in Table 7.5. We use all images for *Car-2* dataset and randomly sample 500 images from each class for *CIFAR-10* and *Birds-525* datasets. We compare each class in the real dataset to its corresponding class in artistic and photorealistic synthetic datasets. As expected, photorealistic synthetic datasets are always visually more similar to the real data.

However, training on realistic synthetic data is not always better than training on artistic images as shown in Table 7.4. Thus, in this experiment, we quantitatively show that photorealism does not imply better usability. Photorealism may limit the diversity of the generated images using DMs. In contrast, artistic images give the

Table 7.5: **Visual similarity between synthetic and real datasets**. FID scores between the classes show that photorealistic images are more similar to the real data compared to artistic ones, as expected.

	Car Accidents		*CIFAR-10*									
	Accident	*Intact*	*Airplane*	*Automobile*	*Bird*	*Cat*	*Deer*	*Dog*	*Frog*	*Horse*	*Ship*	*Truck*
Real-ArtisticSynth	7	5	33	15	13	25	11	25	16	10	22	14
Real-PhotoSynth	2	3	11	6	8	4	7	4	10	4	6	4

	Birds-525											
	BlueThroated PipingGuan	*Cerulean Warbler*	*Visayan Hornbill*	*BandTailed Guan*	*Mallard Duck*	*Rock Dove*	*AfricanPied Hornbill*	*Philippine Eagle*	*Malachite Kingfisher*	*Barn Swallow*	*RoseBreasted Cockatoo*	*Takahe*
Real-ArtisticSynth	7	10	9	8	9	9	10	6	23	11	11	14
Real-PhotoSynth	4	5	4	4	5	6	6	2	4	8	9	6

generative model more freedom to diversify style, color palette, scene composition, brushwork, scene elements proportions, and textures. Additionally, the usability of synthetic images also depends on the essence of the problem. For example, a binary classification task is more easy to learn compared to a multi-classification problem.

Moreover, the visual similarity among the classes of the same dataset also plays an essential role in controlling the usability of synthetic images in practice. As shown in Table 7.6, the classes in the *R-Birds-525* dataset are visually more similar and harder to classify. Thus, photorealism is more important and can lead to better usability. In contrast, diversity is more important in other problems such as car accident classification where intact cars are essentially different from cars with accidents. This explains why non-photorealistic images yield better results than photorealistic ones for the car accident classification task. Thus, we cannot say that more diversity of synthetic images will always mean better usability.

7.4.4.2 Usability metric for selecting training data

As we have seen so far, we need a new metric to assess the usability of synthetic images. We utilize our usability metric shown in Eq. 7.1 to rank synthetic images and select the most useful ones for training. We conducted qualitative and quantitative experiments to illustrate the applicability of our approach. In Figure 7.4 we illustrate the images that are deemed to be most and least suitable images for training based

Table 7.6: **Comparative Analysis of *FID* Scores between *R-Birds-525* and *R-CIFAR-10* Datasets.** *FID* scores among 10 randomly sampled classes from *R-Birds-525* dataset versus *R-CIFAR-10*. *R-Birds-525* dataset shows relatively low *FID* scores compared to *R-CIFAR-10*. Thus, the classes are visually more similar and harder to classify.

		R-Birds-525									
		Class 1	*Class* 2	*Class* 3	*Class* 4	*Class* 5	*Class* 6	*Class* 7	*Class* 8	*Class* 9	*Class* 10
R-CIFAR-10	*Class* 1		2	1	3	4	2	2	5	4	1
	Class 2	8		1	4	5	3	5	4	3	3
	Class 3	16	4		4	6	3	4	5	5	1
	Class 4	3	29	10		3	7	8	5	4	5
	Class 5	5	1	14	4		8	6	5	2	7
	Class 6	3	3	2	19	8		4	7	5	2
	Class 7	4	0	5	1	15	4		5	6	2
	Class 8	4	2	4	8	4	12	8		4	6
	Class 9	9	4	8	4	10	4	9	1		6
	Class 10	12	11	16	20	16	30	18	1	11	

on our metric.

Image background: The best images have photo-realistic, diverse, and visually intricate backgrounds. In contrast, to simple, plane, minimal details backgrounds in less usable images according to our metric. This is clear in Figure 7.4 specially for least usable synthetic images for *SP-CIFAR-10* and *SP-Birds-525* datasets. We can see these images have mostly single-color and abstract backgrounds. In contrast, more realistic, detailed, and problem-relevant backgrounds are shown in more usable synthetic images.

Weather condition: It is clear from Figure 7.4 specially for *SP-Car-2* dataset that our metric is able to select the most diverse examples from the training dataset. For example, snowy weather condition were given higher score as they are photo-realistic and less frequent in the dataset.

Object of interest: Our metric filtered images with distortions in the object of

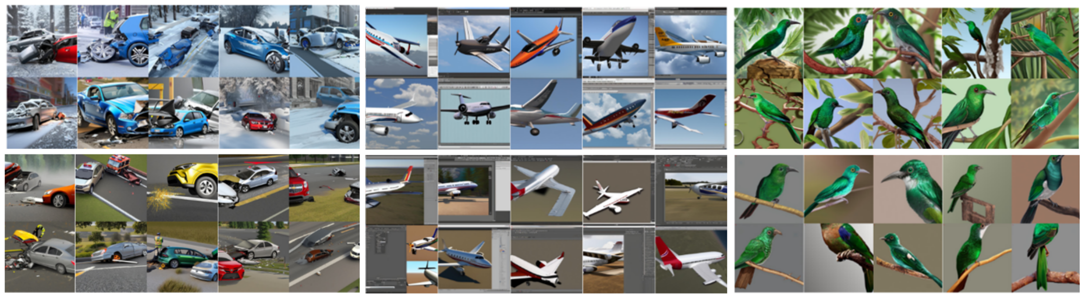


Figure 7.4: The best (first row) and worst (second row) synthetic images from the three datasets: *SP-Car-2* (first col), *SP-CIFAR-10* (second col), and *SP-Birds-525* (third col) according to our usability metric.

interest as clearly seen from *SP-CIFAR-10* in Figure 7.4.

The quantitative results shown in Table 7.4 support the qualitative results too. Fine-tuning on 50% less synthetic training data selected using our approach achieves comparable or better results compared to fine-tuning on 2 times more randomly selected synthetic images.

7.4.4.3 Comparative analysis on usability metrics

To further investigate the usability of our approach, we compare our metric to *SSIM, PSNR, IS,* and *FID* as shown in Table 7.7. The six architectures are trained from scratch on 50% synthetic and 50% real data. The synthetic images are selected from the photorealistic synthetic datasets using these metrics.

Our metric achieves the best results on all datasets for all the architectures (except $SwinTransformer$ on *R-CIFAR-10*). This illustrates the superiority of our approach in ranking synthetic images and effectively training a wide set of classifiers for three different problems.

We also support our experiments with qualitative results as shown in Figure 7.5. It shows the most usable synthetic images ranked by *SSIM, PSNR, IS, FID,* and our metric on *SP-Car-2, SP-CIFAR-10,* and *SP-Birds-525*. Our method selects more photorealistic and diverse training samples as compared to other approaches. Figure 7.5 shows that *SSIM* and *PSNR* give higher scores for more abstract images.

While *IS* and FID give photorealistic images higher scores with no guarantee that these images are diverse.

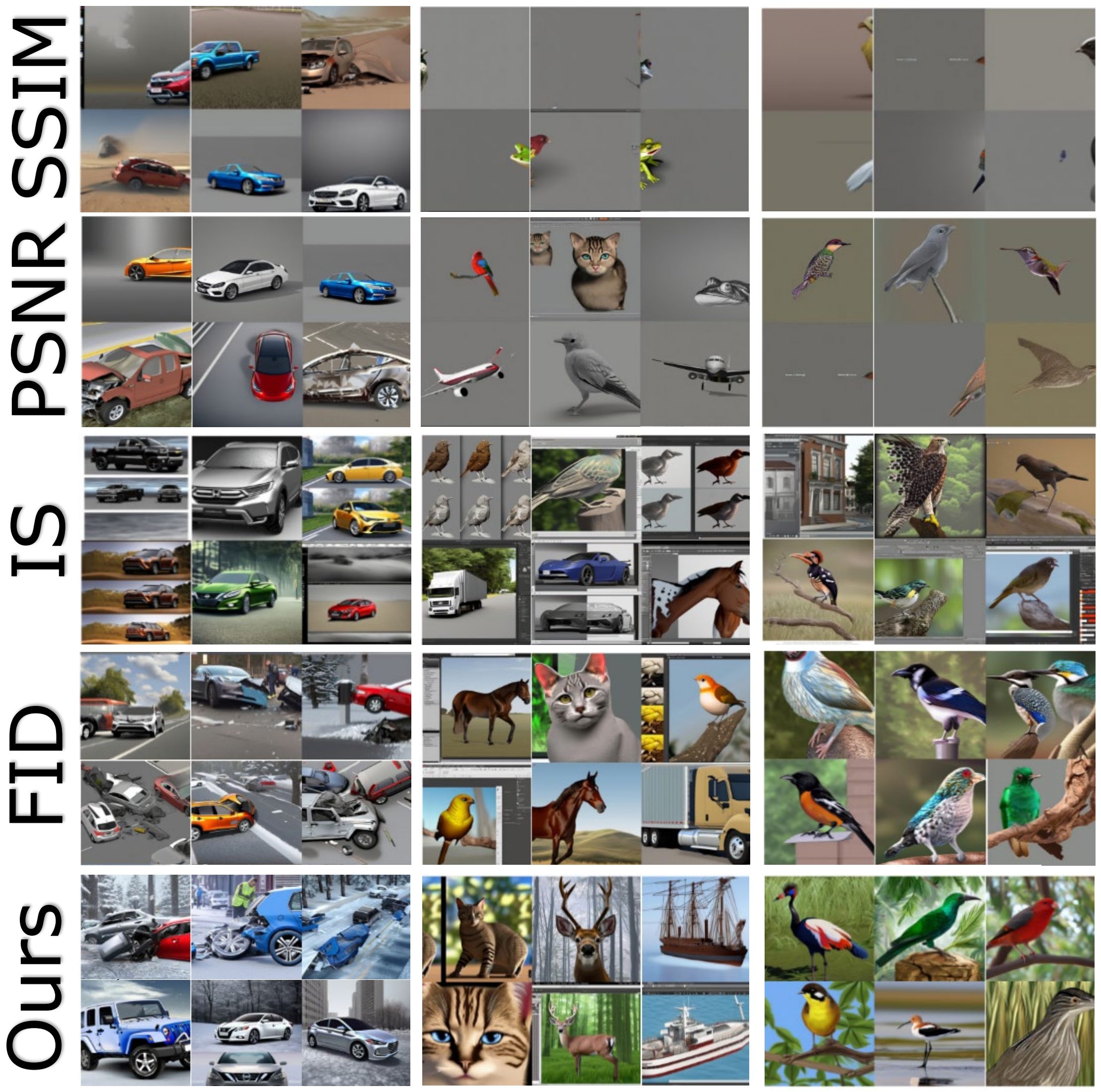


Figure 7.5: Qualitative comparison among most usable synthetic images ranked by SSIM, PSNR, IS, FID, and our proposed metric on *SP-Car-2*, *SP-CIFAR-10*, and *SP-Birds-525*.

Table 7.7: Quantitative comparison among different usability metrics *SSIM, PSNR, IS, FID*, and our metric on three datasets: *R-Car-2, R-CIFAR-10,* and *R-Birds-525* (best in **bold**).

	Model	*SSIM*	*PSNR*	*IS*	*FID*	*Ours*
R-Car-2	*AlexNet*	61	50	57	58	**64**
	EfficientNet	50	50	51	52	**65**
	ViT	62	62	63	63	**65**
	SwinTransformer	64	65	65	65	**68**
	VGG	59	58	60	60	**68**
	REGNet	61	63	65	66	**70**
R-CIFAR-10	*AlexNet*	59	57	60	59	**65**
	EfficientNet	30	23	26	31	**35**
	ViT	42	48	46	47	**52**
	SwinTransformer	58	54	**58**	57	56
	VGG	70	70	71	63	**75**
	REGNet	55	56	55	56	**57**
R-Birds-525	*AlexNet*	63	62	62	61	**66**
	EfficientNet	10	12	15	17	**20**
	ViT	53	50	52	50	**56**
	SwinTransformer	81	80	70	81	**84**
	VGG	59	60	61	58	**66**
	REGNet	70	73	77	75	**80**

7.5 Additional Qualitative Results

In this section, we present additional qualitative results to demonstrate the superiority of our proposed metric compared to widely used metrics such as SSIM, PSNR, IS, and FID.

Figures 7.6, 7.7, and 7.8 showcase the robustness and effectiveness of our metric in capturing both photorealism and diversity of generated synthetic images. Visually complex and detailed scenes were given higher usability scores because our metric excels in evaluating image quality by considering not only pixel-wise differences but also perceptually relevant features.

Qualitative comparisons reveal instances where SSIM (Z. Wang et al., 2004) and PSNR may fall short, especially in scenarios involving highly non-photorealistic images, where our metric provides a good alternative. Furthermore, our metric proves to be more aligned with human perception when compared to IS (Salimans et al., 2016) and FID (Heusel et al., 2017). The qualitative results reinforce that our metric is not only versatile but also superior in providing a more holistic assessment of synthetic images usability.

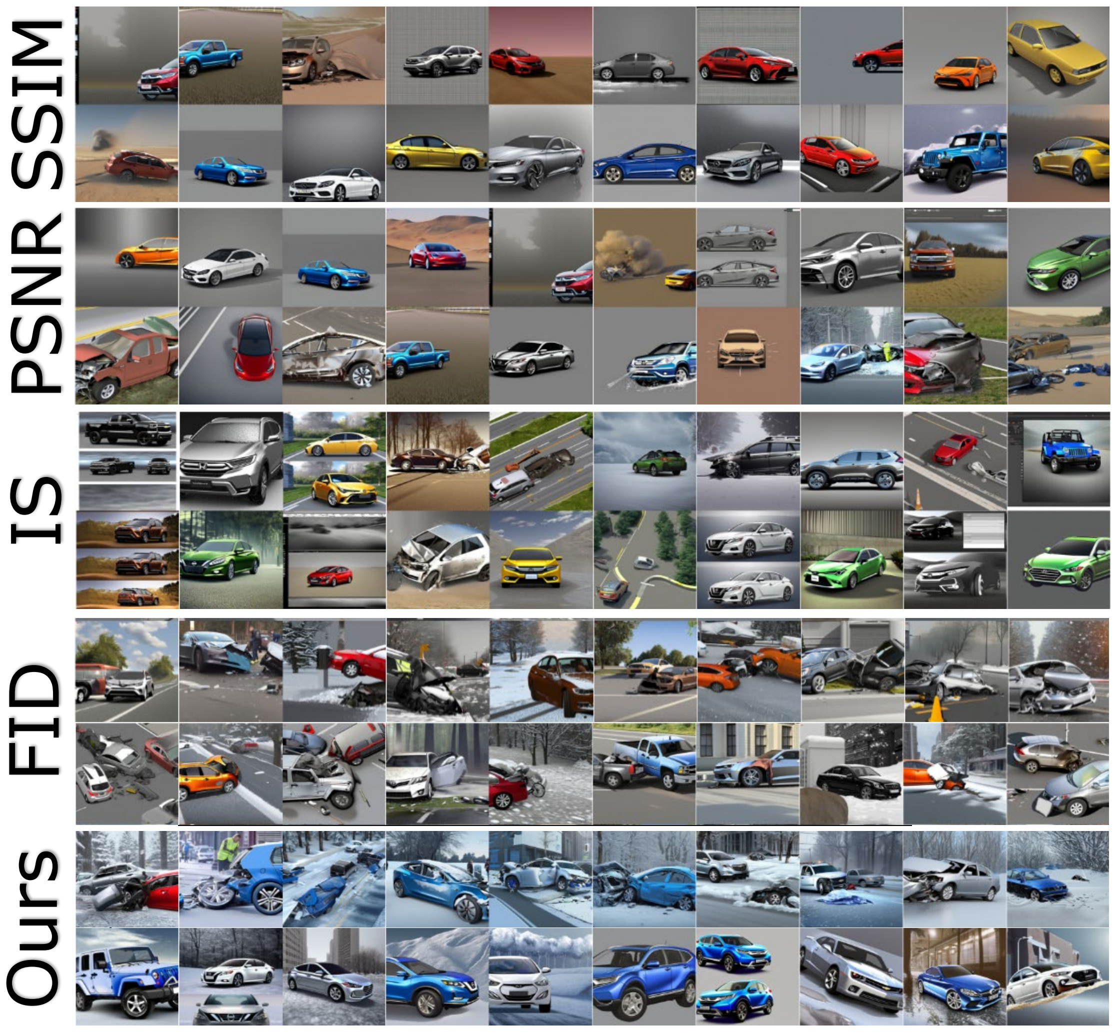


Figure 7.6: Qualitative comparison among most usable synthetic images ranked by *SSIM, PSNR, IS, FID,* and our metric on *SP-Car-2*.

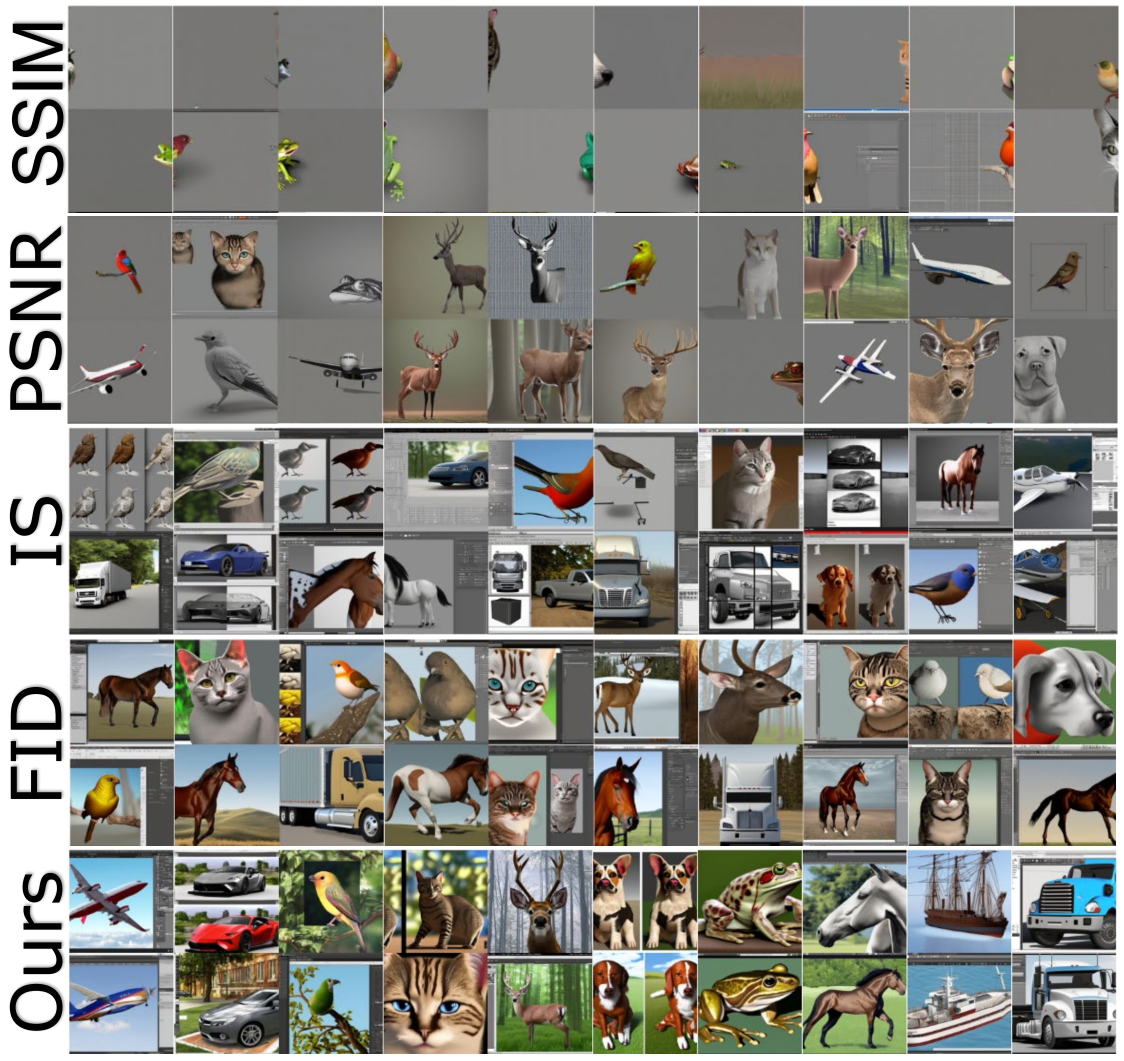


Figure 7.7: Qualitative comparison among most usable synthetic images ranked by *SSIM*, *PSNR*, *IS*, *FID*, and our metric on *SP-CIFAR-10*.

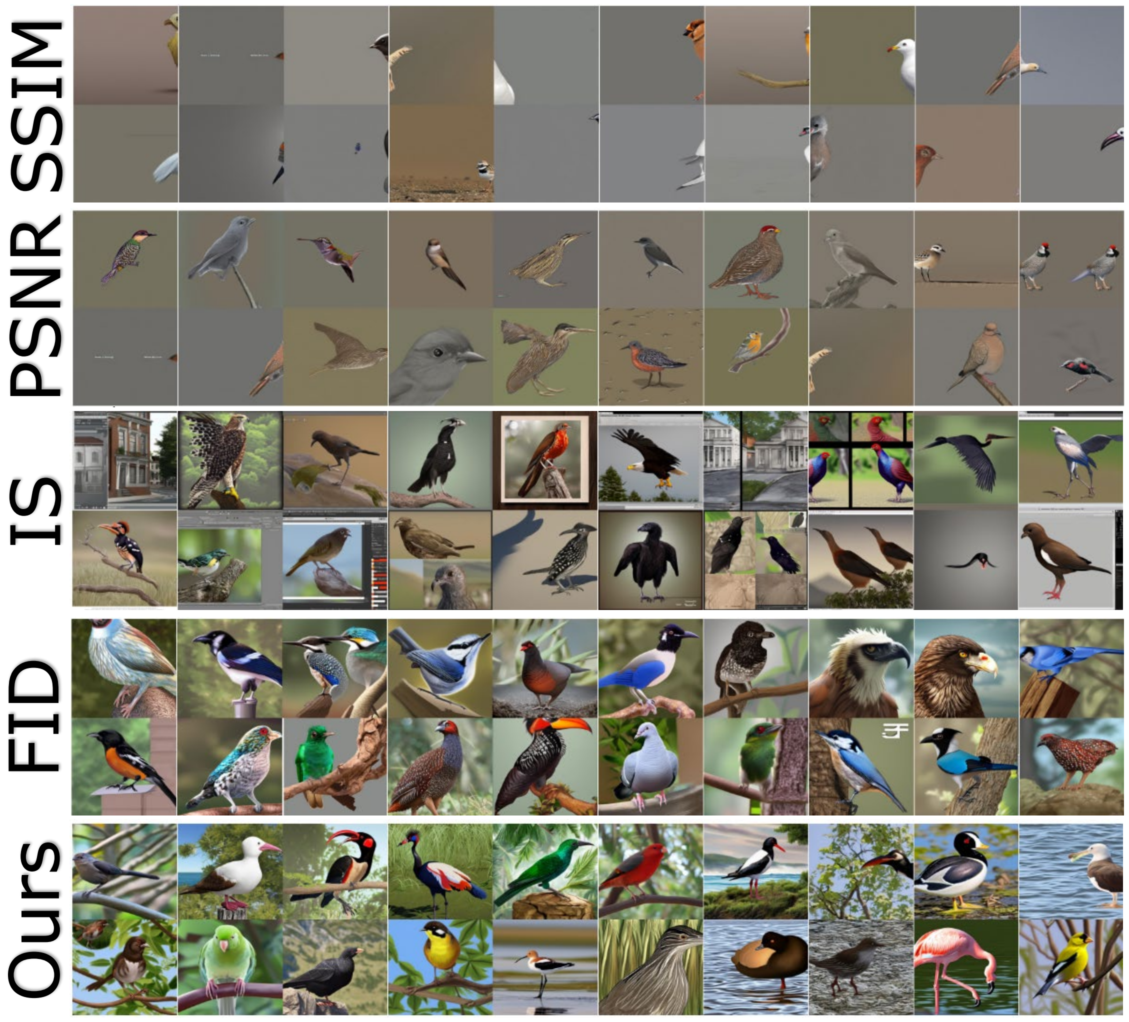


Figure 7.8: Qualitative comparison among most usable synthetic images ranked by *SSIM, PSNR, IS, FID,* and our metric on *SP-Birds-525*.

7.6 Discussion and Concluding Remarks

In this chapter, we have presented the problem of assessing the usability of synthetic images. We developed a novel synthetic image generation pipeline that leverages large language models and diffusion models to automate the generation process. Therefore, the main contribution of this chapter was the development of a novel system that offers a diverse and photorealistic collection of images suitable for computer vision models training. Additionally, the introduction of a new metric that effectively captures the strengths and limitations of using synthetic data to train models is another key contribution of this chapter.

It should be noted that our approach proves highly effective in addressing the main challenges usually associated with synthetic data utilization in supervised machine learning and more especially computer vision models. The novel usability metric we propose successfully disentangles photorealism from diversity, offering a streamlined and efficient means to evaluate synthetic data usability which is of paramount importance in the computer vision field.

Utilizing our usability metric and integration of large language models with diffusion models in our synthetic data generation pipeline yields compelling results, showcasing comparable or superior performance with a significantly reduced volume of synthetic data. This efficiency is underscored by the versatility of our modular framework, which allows for the separate application of the synthetic data generation pipeline and usability metric.

Specifically, in this chapter, we show that training on 50% less synthetic images generated and selected by our approach can achieve similar or better results compared to training on two times randomly synthetic images. Furthermore, the extensive experimentation, evaluating six diverse architectures across three real datasets, further validates the efficacy of our approach in advancing the practicality and impact of synthetic data in computer vision applications.

Chapter 8

Conclusions and Future Work

In conclusion, this thesis has made an effort to address the multifaceted challenges associated with relying only on real data for essential computer vision tasks and applications. The thesis focused on leveraging synthetic data for three fundamental computer vision tasks: semantic segmentation, video stabilization, and image classification. The development of our synthetic data generator, *Silver*, represents a key contribution, emphasizing photorealism, diversity, and scalability. Through the integration of HDRP and complex PCG algorithms, *Silver* can provide accurately and unbiased annotated large-scale training datasets for various computer vision tasks.

Furthermore, the thesis introduced novel synthetic-aware architectures tailored for domain adaptation in semantic segmentation and adverse weather video stabilization. Augmenting existing models, for semantic segmentation, with weather and time-of-day supervisors trained through multi-task learning, our proposed architecture demonstrated remarkable improvements in model robustness and accuracy, particularly under challenging environmental and illumination conditions such as rain, snow, fog, and nighttime.

Additionally, our novel approach to video stabilization in adverse weather conditions is another key contribution of this thesis. Our algorithm has effectively addressed inherent challenges in feature extraction, yielding superior performance across diverse real-world scenarios. The introduction of *VSAC105Real* dataset is another key contribution to the field too.

Moreover, our proposed novel usability metric represents another significant advancement in the evaluation of synthetic data, allowing for a detailed assessment of both photorealism and diversity. This metric not only streamlines the evaluation process but also facilitates informed decision-making regarding the suitability or usability of generated synthetic data for image classification task.

Overall, the findings presented in this thesis highlight the huge potential of synthetic data in revolutionizing the field of computer vision. By overcoming the limitations associated with traditional data collection and annotation methods, synthetic data offers a cost-effective and scalable solution for training robust computer vision models. Next, we will discuss each major contribution in more detail.

8.1 Simulator

In the scope of applying synthetic data for computer vision tasks, data diversity is of a central concern. *Silver* deploys uniform distribution to set colors, select 3D models, choose animations or to configure other scene elements. In reality this may be considered as a strong assumption. To address this, we are planning to consider the real distribution of scene elements and their associated attributes as observed in the real world. At the same time, we will make these distributions conditioned on the type of weather condition and other related aspects of the environment. For example, in a winter environment, it is more likely to observe people wearing dark color clothes and so on.

In this work, a qualitative overview of the system was shown. However, an open question was to prove the usability of the generated data for training and/or evaluation purposes which was the main scope of Chapter 7.

In future work, we plan to answer this question considering some major computer vision tasks such as semantic segmentation and depth estimation. In parallel, we aim to extend the system to other computer vision tasks and improve the photo-realism and diversity even further. We believe that utilizing *Silver* to generate photo-realistic, large-scale, and diverse training data will help computer vision models to train better and to achieve better performance on real-world data.

8.2 Semantic segmentation

In this part of the thesis, we addressed the key challenge of semantic segmentation models performance degradation in adverse weather conditions and nighttime scenarios. We experimentally demonstrated that standard models excel under typical conditions but struggle in challenging environments, and collecting annotated data for such conditions is impractical. Leveraging synthetic data to augment training sets is common, but it can negatively impact performance under normal conditions. Our proposed solution introduced a novel architecture, a modification to *DeepLabV3+*, incorporating weather and time-of-day supervisors via multi-task learning. This addition enhanced the model's adaptability to both adverse weather and nighttime conditions.

An exciting extension to our work may involve exploring the adaptability of the proposed architecture to diverse environmental factors beyond weather and time-of-day. Consider incorporating additional supervisors for factors such as varied lighting conditions, seasonal changes, or urban versus rural settings. This broader approach could enhance the model's robustness across a spectrum of real-world scenarios, making it more versatile for a wide range of applications.

As a pathway for future research, it would be beneficial to explore the potential of the proposed architecture in transfer learning across different domains. Investigate the model's performance when trained on synthetic data from one specific domain and adapted to another, evaluating its adaptability and generalization. Additionally, considering the dynamic nature of weather conditions and lighting, implementing a real-time adaptation mechanism that continuously adjusts the model during inference based on the environmental cues could be a promising avenue for further exploration. This could lead to a more dynamic, robust, and responsive semantic segmentation model that adapts on-the-fly to changing conditions.

8.3 Video stabilization

Acknowledging the limitations of current methods in challenging scenarios, we introduced a novel synthetic-aware adverse weather video stabilization algorithm. This algorithm, a departure from existing approaches, exclusively relies on synthetic data for training, overcoming prevalent issues in feature extraction. Our methodology involved a specially designed synthetic data, eliminating the need for real data and incorporating an automatic ground-truth extraction procedure. To validate the efficacy of the proposed algorithm, a new dataset, VSAC105Real, is introduced, and a comprehensive comparison against five recent video stabilization algorithms is conducted using two benchmark datasets. The results demonstrate the algorithm's superior generalization across real-world videos under diverse weather conditions, emphasizing its robustness without necessitating extensive synthetic training data.

An extension to this part of the thesis could involve a deeper exploration of the algorithm's adaptability to various camera types and recording setups. Additionally, investigating the real-time applicability of the proposed method in resource-constrained environments would contribute to the practicality of its implementation. Furthermore, an examination of the algorithm's performance in scenarios with dynamic and rapid scene changes and complex motion patterns could provide valuable insights into its versatility.

8.4 Usability metric

In this part of the thesis, we addressed the challenge of assessing the usability of synthetically generated data. Acknowledging the pivotal role of large-scale training datasets, we proposed a novel training procedure and a usability metric designed to disentangle photorealism from diversity in synthetic data. This metric serves as a simple yet effective tool to rank synthetic images based on their usability. We also proposed an innovative pipeline for synthetic data generation by integrating Large Language Models with Stable Diffusion. Quantitative results demonstrate that the proposed approach achieves comparable or superior results with less training data. The impact of photorealism on synthetic data usability is systematically assessed through an extensive set of experiments involving six different architectures and three diverse datasets.

In the future, we are planning to investigate the usability of our approach to other computer vision tasks, such as semantic segmentation, instance segmentation, and depth estimation. We aim to refine and extend our synthetic data usability framework, focusing on enhancing the diversity and realism of generated images. Further exploration will involve optimizing the integration of large language models and stable diffusion models for an even more efficient synthetic data generation pipeline. Additionally, we plan to investigate the generalization capabilities of models trained on our synthetic data across a broader spectrum of real-world scenarios including adverse weather conditions, extreme light conditions, and low-resolution images.

Continuous efforts will be directed toward expanding the application domains and evaluating the scalability of our approach to accommodate more diverse architectures and datasets. Addressing these aspects will contribute to a more comprehensive understanding of synthetic data role in advancing machine learning and further solidify its efficacy in various fundamental and key computer vision tasks and applications. Future work will involve a comprehensive examination of the potential biases embedded in our synthetic data generation pipeline and selection metric and the consequential impact on model performance and robustness.

References

Akrout, Mohamed et al. (2023). “Diffusion-based Data Augmentation for Skin Disease Classification: Impact Across Original Medical Datasets to Fully Synthetic Images”. In: *arXiv preprint arXiv:2301.04802*.

Al Khalil, Yasmina et al. (2023). “On the Usability of Synthetic Data for Improving the Robustness of Deep Learning-based Segmentation of Cardiac Magnetic Resonance Images”. In: *Medical Image Analysis* 84, p. 102688.

Ali, Hazrat, Christer Grönlund, and Zubair Shah (2023). “Leveraging GANs for Data Scarcity of COVID-19: Beyond the Hype”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 659–667.

Ali, Muhammad Kashif, Sangjoon Yu, and Tae Hyun Kim (2020). “Learning Deep Video Stabilization without Optical Flow”. In: *arXiv preprint arXiv:2011.09697*.

Alshammari, Naif, Samet Akcay, and Toby P Breckon (2020). “Competitive simplicity for multi-task learning for real-time foggy scene understanding via domain adaptation”. In: *arXiv preprint arXiv:2012.05304*.

Alvarez, Jose M et al. (2012). “Road scene segmentation from a single image”. In: *European Conference on Computer Vision*. Springer, pp. 376–389.

Bansal, Arpit et al. (2023). “Universal guidance for diffusion models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 843–852.

Barath, Daniel, Jiri Matas, and Jana Noskova (2019). “MAGSAC: marginalizing sample consensus”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10197–10205.

Barbosa, Igor Barros et al. (2018). "Looking Beyond Appearances:Synthetic Training Data for Deep CNNs in Re-identification". In: *Computer Vision and Image Understanding* 167.

Baroroh, Dawi Karomati, Chih-Hsing Chu, and Lihui Wang (2021). "Systematic literature review on augmented reality in smart manufacturing: Collaboration between human and computational intelligence". In: *Journal of Manufacturing Systems* 61, pp. 696–711.

Barratt, Shane and Rishi Sharma (2018). "A note on the inception score". In: *arXiv preprint arXiv:1801.01973*.

Boff, Kenneth R, Lloyd Kaufman, and James P Thomas (1986). *Handbook of perception and human performance.* Vol. 1. Wiley New York.

Borji, Ali (2019). "Pros and cons of gan evaluation measures". In: *Computer vision and image understanding* 179, pp. 41–65.

Bradley, Arwen et al. (2021). "Cinematic-L1 Video Stabilization with a Log-Homography Model". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1041–1049.

Brostow, Gabriel J, Julien Fauqueur, and Roberto Cipolla (2009). "Semantic object classes in video: A high-definition ground truth database". In: *Pattern Recognition Letters* 30.2, pp. 88–97.

Butler, Daniel J et al. (2012). "A Naturalistic Open Source Movie for Optical Flow Evaluation". In: *European conference on computer vision.*

Carlini, Nicolas et al. (2023). "Extracting training data from diffusion models". In: *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 5253–5270.

Chakraborty, Tanujit et al. (2024). "Ten years of generative adversarial nets (GANs): a survey of the state-of-the-art". In: *Machine Learning: Science and Technology* 5.1, p. 011001.

Chang, Woong-Gi et al. (2019). "Domain-specific batch normalization for unsupervised domain adaptation". In: *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 7354–7362.

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, et al. (2017). “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.4, pp. 834–848.

Chen, Liang-Chieh, George Papandreou, Florian Schroff, et al. (2017). “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv preprint arXiv:1706.05587*.

Chen, Liang-Chieh, Yukun Zhu, et al. (2018). “Encoder-decoder with atrous separable convolution for semantic image segmentation”. In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818.

Chen, Pengfei et al. (2019). “Understanding and utilizing deep neural networks trained with noisy labels”. In: *International Conference on Machine Learning*. PMLR, pp. 1062–1070.

Chen, Ping-Rong et al. (2020). “DSNet: An efficient CNN for road scene segmentation”. In: *APSIPA Transactions on Signal and Information Processing* 9.

Chen, Tianlong et al. (2022). “The principle of diversity: Training stronger vision transformers calls for reducing all levels of redundancy”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12020–12030.

Cheung, Ernest et al. (2016). “LCrowdV: Generating Labeled Videos for Simulation-Based Crowd Behavior Learning”. In: *European Conference on Computer Vision*.

Chiang, Feng-Kuang, Xiaojing Shang, and Lu Qiao (2022). “Augmented reality in vocational training: A systematic review of research and applications”. In: *Computers in Human Behavior* 129, p. 107125.

Choi, Jinsoo and In So Kweon (2020). “Deep iterative frame interpolation for full-frame video stabilization”. In: *ACM Transactions on Graphics (TOG)* 39.1, pp. 1–9.

Colom, Roberto et al. (2010). “Human intelligence and brain networks”. In: *Dialogues in clinical neuroscience* 12.4.

Cordts, Marius et al. (2016). “The cityscapes dataset for semantic urban scene understanding”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223.

Costa, Gabriel de Moura, Marcelo Roberto Petry, and António Paulo Moreira (2022). “Augmented Reality for Human–Robot Collaboration and Cooperation in Industrial Applications: A Systematic Literature Review”. In: *Sensors* 22.7, p. 2725.

Creswell, Antonia et al. (2018). “Generative adversarial networks: An overview”. In: *IEEE signal processing magazine* 35.1, pp. 53–65.

Croitoru, Florinel-Alin et al. (2023). “Diffusion models in vision: A survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Cry Engine (2024). `https://www.cryengine.com/`. Online; accessed: 2023-09-14.

Deng, Jia et al. (2009). “ImageNet: A Large-Scale Hierarchical Image Database”. In: *2009 IEEE conference on computer vision and pattern recognition*.

DeTone, Daniel, Tomasz Malisiewicz, and Andrew Rabinovich (2016). “Deep image homography estimation”. In: *arXiv preprint arXiv:1606.03798*.

Devlin, Jacob et al. (2018). “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805*.

Dhariwal, Prafulla and Alexander Nichol (2021). “Diffusion models beat gans on image synthesis”. In: *Advances in neural information processing systems* 34, pp. 8780–8794.

Dietrich, Cindy (2010). “Decision making: Factors that influence decision making, heuristics used, and decision outcomes”. In: *Inquiries Journal* 2.02.

Dosovitskiy, Alexey, Lucas Beyer, et al. (2020). “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929*.

Dosovitskiy, Alexey, Philipp Fischer, et al. (2015). “Flownet: Learning optical flow with convolutional networks”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 2758–2766.

Dosovitskiy, Alexey, German Ros, et al. (2017). “CARLA: An open urban driving simulator”. In: *Conference on robot learning*. PMLR, pp. 1–16.

Dundar, Aysegul et al. (2020). “Domain stylization: A fast covariance matching framework towards domain adaptation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.7, pp. 2360–2372.

Evans, Jonathan St BT (2002). “Logic and human reasoning: an assessment of the deduction paradigm.” In: *Psychological bulletin* 128.6, p. 978.

Everingham, Mark et al. (2015). “The pascal visual object classes challenge: A retrospective”. In: *International journal of computer vision* 111.1, pp. 98–136.

Ferwerda, James A (2003). “Three varieties of realism in computer graphics”. In: *Human vision and electronic imaging viii*. Vol. 5007. SPIE, pp. 290–297.

Fischler, Martin A and Robert C Bolles (1981). “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24.6, pp. 381–395.

Fu, Jun et al. (2019). “Dual attention network for scene segmentation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3146–3154.

Gaidon, Adrien et al. (2016). “Virtual Worlds as Proxy for Multi-Object Tracking Analysis”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Ge, Yuying et al. (2019). “DeepFashion2: A Versatile Benchmark for Detection, Pose Estimation, Segmentation and Re-Identification of Clothing Images”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Al-Gethami, Khalid M, Mousa T Al-Akhras, and Mohammed Alawairdhi (2021). “Empirical evaluation of noise influence on supervised machine learning algorithms using intrusion detection datasets”. In: *Security and Communication Networks* 2021, pp. 1–28.

Gong, Zhiqiang, Ping Zhong, and Weidong Hu (2019). "Diversity in Machine Learning". In: *Ieee Access* 7, pp. 64323–64350.

Gonzalez-Franco, Mar, Ofek, et al. (2020). "The Rocketbox Library and the Utility of Freely Available Rigged Avatars". In: *Frontiers in virtual reality* 1.article 561558.

Goodfellow, Ian et al. (2020). "Generative adversarial networks". In: *Communications of the ACM* 63.11, pp. 139–144.

Goyal, Manik et al. (2017). "Dataset augmentation with synthetic images improves semantic segmentation". In: *National Conference on Computer Vision, Pattern Recognition, Image Processing, and Graphics.* Springer, pp. 348–359.

Greff, Klaus et al. (2022). "Kubric: a scalable dataset generator". In.

Grundmann, Matthias, Vivek Kwatra, and Irfan Essa (2011). "Auto-directed video stabilization with robust l1 optimal camera paths". In: *CVPR 2011.* IEEE, pp. 225–232.

Guo, Mengxi et al. (2020). "High-level task-driven single image deraining: Segmentation in rainy days". In: *International Conference on Neural Information Processing.* Springer, pp. 350–362.

He, Kaiming et al. (2015). "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 37.9, pp. 1904–1916.

Heusel, Martin et al. (2017). "Gans trained by a two time-scale update rule converge to a local nash equilibrium". In: *Advances in neural information processing systems* 30.

Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33, pp. 6840–6851.

Hosseini, Hadi (2024). "The Fairness Fair: Bringing Human Perception into Collective Decision-Making". In: *Proceedings of the AAAI Conference on Artificial Intelligence.* Vol. 38. 20, pp. 22624–22631.

Hurl, Braden, Krzysztof Czarnecki, and Steven Waslander (2019). "Precise Synthetic Image and LiDAR (PreSIL) Dataset for Autonomous Vehicle Perception". In: *IEEE Intelligent Vehicles Symposium*.

Huynh-Thu, Quan and Mohammed Ghanbari (2008). "Scope of validity of PSNR in image/video quality assessment". In: *Electronics letters* 44.13, pp. 800–801.

Hyontai, SUG (2018). "Performance of machine learning algorithms and diversity in data". In: *MATEC Web of Conferences*. Vol. 210. EDP Sciences, p. 04019.

Iglesias, Guillermo, Edgar Talavera, and Alberto Díaz-Álvarez (2023). "A survey on GANs for computer vision: Recent research, analysis and taxonomy". In: *Computer Science Review* 48, p. 100553.

Ilg, E. et al. (July 2017). "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. URL: `http://lmb.informatik.uni-freiburg.de//Publications/2017/IMKDB17`.

Ilse, Maximilian et al. (2020). "Diva: Domain invariant variational autoencoders". In: *Medical Imaging with Deep Learning*. PMLR, pp. 322–348.

Ivanovs, Maksims et al. (2022). "Improving semantic segmentation of urban scenes for self-driving cars with synthetic images". In: *Sensors* 22.6, p. 2252.

Javier Hidalgo-Carrio, Daniel Gehrig and Davide Scaramuzza (2020). "Learning Monocular Dense Depth from Events". In: *IEEE International Conference on 3D Vision.(3DV)*. URL: `http://rpg.ifi.uzh.ch/docs/3DV20_Hidalgo.pdf`.

Jin, Lianchao, Fuxiao Tan, Shengming Jiang, et al. (2020). "Generative adversarial network technologies and applications in computer vision". In: *Computational intelligence and neuroscience* 2020.

Johnson-Laird, Philip N, Sangeet S Khemlani, and Geoffrey P Goodwin (2015). "Logic, probability, and human reasoning". In: *Trends in cognitive sciences* 19.4, pp. 201–214.

Junayed, Masum Shah et al. (2022). "HiMODE: A Hybrid Monocular Omnidirectional Depth Estimation Model". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5212–5221.

Kang, Minguk et al. (2023). "Scaling up gans for text-to-image synthesis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10124–10134.

Karimi, Davood et al. (2020). "Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis". In: *Medical image analysis* 65, p. 101759.

Kariyappa, Sanjay and Moinuddin K Qureshi (2019). "Improving Adversarial Robustness of Ensembles with Diversity Training". In: *arXiv preprint arXiv:1901.09981*.

Katayama, Takafumi et al. (2022). "Domain adaptation through photorealistic enhanced images for semantic segmentation". In: *Mathematical Problems in Engineering* 2022.

Kerim, Abdulrahman (2023). *Synthetic Data for Machine Learning: Revolutionize your approach to machine learning with this comprehensive conceptual guide.* Packt Publishing. ISBN: 9781803232607. URL: `https://books.google.co.uk/books?id=JpXeEAAAQBAJ`.

Kerim, Abdulrahman, Cem Aslan, et al. (2021). "NOVA: Rendering virtual worlds with humans for computer vision tasks". In: *Computer Graphics Forum*. Vol. 40. 6. Wiley Online Library, pp. 258–272.

Kerim, Abdulrahman, Ufuk Celikcan, et al. (2021). "Using Synthetic Data for Person Tracking under Adverse Weather Conditions". In: *Image and Vision Computing* 111, p. 104187.

Kerim, Abdulrahman, Felipe Chamone, et al. (2022). "Semantic Segmentation under Adverse Conditions: A Weather and Nighttime-Aware Synthetic Data-Based Approach". In: *British Machine Vision Conference (BMVC)*.

Kerim, Abdulrahman, Leandro Soriano Marcolino, et al. (2024). "On the Usability Usability of Synthetic Data for Image Classification". In: *Under Review - Computer Vision and Pattern Recognition (CVPR).*

Kerim, Abdulrahman, Washington LS Ramos, et al. (2024). "Leveraging Synthetic Data to Learn Video Stabilization Under Adverse Conditions". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV).*

Kerim, Abdulrahman, Leandro Soriano Marcolino, and Richard Jiang (2021). "Silver: Novel Rendering Engine for Data Hungry Computer Vision Models". In: *2nd International Workshop on Data Quality Assessment for Machine Learning.*

Kiefer, Benjamin, David Ott, and Andreas Zell (2022). "Leveraging synthetic data in object detection on unmanned aerial vehicles". In: *2022 26th International Conference on Pattern Recognition (ICPR).* IEEE, pp. 3564–3571.

Kodali, Naveen et al. (2017). "On convergence and stability of gans". In: *arXiv preprint arXiv:1705.07215.*

Krizhevsky, Alex, Geoffrey Hinton, et al. (2009). "Learning multiple layers of features from tiny images". In.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet Classification with Deep Convolutional Neural Networks". In: *Advances in neural information processing systems* 25.

Kumar, Varun Ravi et al. (2021). "Syndistnet: Self-supervised monocular fisheye camera distance estimation synergized with semantic segmentation for autonomous driving". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 61–71.

Kurzweil, Ray et al. (1990). *The age of intelligent machines.* Vol. 580. MIT press Cambridge.

Lam, Winnie WT and Kenneth NK Fong (2023). "The application of markerless motion capture (MMC) technology in rehabilitation programs: A systematic review and meta-analysis". In: *Virtual Reality* 27.4, pp. 3363–3378.

Lee, Heejae et al. (2023). “Game engine-driven synthetic data generation for computer vision-based safety monitoring of construction workers”. In: *Automation in Construction* 155, p. 105060.

Lee, Ken-Yi et al. (2009). “Video stabilization using robust feature trajectories”. In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE, pp. 1397–1404.

Lei, Yayun et al. (2020). “Semantic image segmentation on snow driving scenarios”. In: *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, pp. 1094–1100.

Li, Da et al. (2017). “Deeper, broader and artier domain generalization”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 5542–5550.

Li, Leheng et al. (2023). “Lift3D: Synthesize 3D Training Data by Lifting 2D GAN to 3D Generative Radiance Field”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 332–341.

Li, Shiwei et al. (2015). “Dual-feature warping-based motion model estimation”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4283–4291.

Li, Yanghao et al. (2018). “Adaptive batch normalization for practical domain adaptation”. In: *Pattern Recognition* 80, pp. 109–117.

Liang, Xuefeng, Xingyu Liu, and Longshan Yao (2022). “Review–a survey of learning from noisy labels”. In: *ECS Sensors Plus* 1.2, p. 021401.

Lin, Tsung-Yi et al. (2014). “Microsoft coco: Common objects in context”. In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, pp. 740–755.

Lisman, John (2015). “The Challenge of Understanding the Brain: Where We Stand in 2015”. In: *Neuron* 86.4.

Liu, Beyang, Stephen Gould, and Daphne Koller (2010). “Single image depth estimation from predicted semantic labels”. In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE, pp. 1253–1260.

Liu, Yu-Lun et al. (2021). “Hybrid Neural Fusion for Full-frame Video Stabilization”. In: *arXiv preprint arXiv:2102.06205*.

Liu, Shuaicheng, Yinting Wang, et al. (2012). “Video stabilization with a depth camera”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 89–95.

Liu, Shuaicheng, Lu Yuan, et al. (2013). “Bundled camera paths for video stabilization”. In: *ACM Transactions on Graphics (TOG)* 32.4, pp. 1–10.

Liu, Yilin, Fuyou Xue, and Hui Huang (2021). “UrbanScene3D: A Large Scale Urban Scene Dataset and Simulator”. In: *arXiv preprint arXiv:2107.04286*.

Liu, Yubin, CB Sivaparthipan, and Achyut Shankar (2022). “Human–computer interaction based visual feedback system for augmentative and alternative communication”. In: *International Journal of Speech Technology* 25.2, pp. 305–314.

Liu, Ze et al. (2021). “Swin transformer: Hierarchical vision transformer using shifted windows”. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022.

Lowe, David G (2004). “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2, pp. 91–110.

Lucas, Bruce D and Takeo Kanade (1981). “An iterative image registration technique with an application to stereo vision”. In: *IJCAI’81: 7th international joint conference on Artificial intelligence*. Vol. 2, pp. 674–679.

Lucic, Mario et al. (2018). “Are gans created equal? a large-scale study”. In: *Advances in neural information processing systems* 31.

Luo, Zixin et al. (2020). “ASLFeat: Learning Local Features of Accurate Shape and Localization”. In: *Computer Vision and Pattern Recognition (CVPR)*.

Ma, Chenxiang, You Zhou, and Zhiqiang Li (2020). “A new simulation environment based on Airsim, ROS, and PX4 for quadcopter aircrafts”. In: *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*. IEEE, pp. 486–490.

Ma, Xianzheng et al. (2022). “Both style and fog matter: Cumulative domain adaptation for semantic foggy scene understanding”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18922–18931.

Mackinlay, Jock et al. (1988). “Designing effective pictures: is photographic realism the only answer?” In: *ACM SIGGRAPH 88 panel proceedings*, pp. 1–48.

Mahon, Louis and Thomas Lukasiewicz (2022). “Measuring Image Complexity as a Discrete Hierarchy using MDL Clustering”. In.

Man, Keith and Javaan Chahl (2022). “A Review of Synthetic Image Data and Its Use in Computer Vision”. In: *Journal of Imaging* 8.11, p. 310.

Material, Adobe Substance (2022). `https://substance3d.adobe.com/assets`. Online; accessed: 2022-07-26.

Mayer, Nikolaus et al. (2016). “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4040–4048.

Merfeld, Katrin, Mark-Philipp Wilhelms, and Sven Henkel (2019). “Being driven autonomously–A qualitative study to elicit consumers’ overarching motivational structures”. In: *Transportation Research Part C: Emerging Technologies* 107, pp. 229–247.

Musat, Valentina et al. (2021). “Multi-weather city: Adverse weather stacking for autonomous driving”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2906–2915.

Nazar, Mobeen et al. (2021). “A systematic review of human-computer interaction and explainable artificial intelligence in healthcare with artificial intelligence techniques”. In: *IEEE Access*.

Neisser, Ulric et al. (1996). “Intelligence: knowns and unknowns.” In: *American psychologist* 51.2, p. 77.

Neuhold, Gerhard et al. (2017). “The mapillary vistas dataset for semantic understanding of street scenes”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 4990–4999.

OpenAI (2023). "ChatGPT: A Large-Scale Transformer-Based Language Model". In: Available at `https://openai.com`.

Pambrun, Jean-François and Rita Noumeir (2015). "Limitations of the SSIM quality metric in the context of diagnostic imaging". In: *2015 IEEE international conference on image processing (ICIP)*. IEEE, pp. 2960–2963.

Papo, David (2015). "How can we study reasoning in the brain?" In: *Frontiers in human neuroscience* 9.

Piano, Luca et al. (Jan. 2023). "Bent & Broken Bicycles: Leveraging Synthetic Data for Damaged Object Re-Identification". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 4881–4891.

Piosenka, Gerald (2023). *Birds 525 Species - Image Classification.* https://www.kaggle.com/datase bird-species.

Rademacher, Paul et al. (2001). "Measuring the perception of visual realism in images". In: *Rendering Techniques 2001: Proceedings of the Eurographics Workshop in London, United Kingdom, June 25–27, 2001 12.* Springer, pp. 235–247.

Radosavovic, Ilija et al. (2020). "Designing network design spaces". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10428–10436.

Rao, Qi et al. (2023). "Sim2RealVS: A New Benchmark for Video Stabilization with a Strong Baseline". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 5406–5415.

Ravuri, Suman and Oriol Vinyals (2019). "Seeing is not necessarily believing: Limitations of biggans for data augmentation". In.

Reda, Fitsum et al. (2017). *Flownet2-pytorch: Pytorch implementation of FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks.* `https://github.com/NVIDIA/flownet2-pytorch`.

Ren, Fuji and Yanwei Bao (2020). "A review on human-computer interaction and intelligent robots". In: *International Journal of Information Technology & Decision Making* 19.01, pp. 5–47.

Revaud, Jerome et al. (2019). “R2D2: Repeatable and Reliable Detector and Descriptor”. In: *NeurIPS*.

Richter, Stephan R, Zeeshan Hayder, and Vladlen Koltun (2017). “Playing for Benchmarks”. In: *Proceedings of the IEEE International Conference on Computer Vision*.

Richter, Stephan R, Vibhav Vineet, et al. (2016). “Playing for Data: Ground Truth from Computer Games”. In: *European conference on computer vision*.

Roberto de Souza, Cesar et al. (2017). “Procedural Generation of Videos to Train Deep Action Recognition Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Rocks, 4K Procedural Terrain with and Mold - Substance Material (HDRP) (2022). `https://assetstore.unity.com/`. Online; accessed: 2022-07-26.

Rombach, Robin et al. (June 2022). “High-Resolution Image Synthesis With Latent Diffusion Models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241.

Ros, German et al. (2016). “The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Rublee, Ethan et al. (2011). “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International conference on computer vision*. Ieee, pp. 2564–2571.

Russell, Stuart J and Peter Norvig (2016). *Artificial intelligence: a modern approach*. Pearson.

Sakaridis, Christos, Dengxin Dai, and Luc Van Gool (2021). “ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10765–10775.

Salimans, Tim et al. (2016). "Improved techniques for training gans". In: *Advances in neural information processing systems* 29.

Sankaranarayanan, Swami et al. (2018). "Learning from synthetic data: Addressing domain shift for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3752–3761.

Savitzky, Abraham and Marcel JE Golay (1964). "Smoothing and differentiation of data by simplified least squares procedures." In: *Analytical chemistry* 36.8, pp. 1627–1639.

Saxena, Ashutosh, Sung Chung, and Andrew Ng (2005). "Learning depth from single monocular images". In: *Advances in neural information processing systems* 18.

Saxena, Ashutosh, Min Sun, and Andrew Y Ng (2008). "Make3d: Learning 3d scene structure from a single still image". In: *IEEE transactions on pattern analysis and machine intelligence* 31.5, pp. 824–840.

Scholz, Daniel et al. (2023). "Metrics to Quantify Global Consistency in Synthetic Medical Images". In: *arXiv preprint arXiv:2308.00402*.

Schuhmann, Christoph et al. (2022). "LAION-5B: An Open Large-Scale Dataset for Training Next Generation Image-Text Models". In: *Advances in Neural Information Processing Systems* 35, pp. 25278–25294.

Shafaei, Alireza and James J Little (2016). "Real-Time Human Motion Capture with Multiple Depth Cameras". In: *2016 13th Conference on Computer and Robot Vision*.

Shafaei, Alireza, James J Little, and Mark Schmidt (2016a). "Play and Learn: Using Video Games to Train Computer Vision Models". In: *arXiv:1608.01745*.

— (2016b). "Play and Learn: Using Video Games to Train Computer Vision Models". In: *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. Ed. by Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith. BMVA Press. URL: `http://www.bmva.org/bmvc/2016/papers/paper026/index.html`.

Shah, Shital et al. (2018). "Airsim: High-fidelity visual and physical simulation for autonomous vehicles". In: *Field and Service Robotics: Results of the 11th International Conference.* Springer, pp. 621–635.

Shipard, Jordan et al. (2023). "Diversity is Definitely Needed: Improving Model-Agnostic Zero-shot Classification via Stable Diffusion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 769–778.

Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556.*

Singhal, Peeyush et al. (2023). "Domain adaptation: challenges, methods, datasets, and applications". In: *IEEE access* 11, pp. 6973–7020.

Song, Hwanjun et al. (2022). "Learning from noisy labels with deep neural networks: A survey". In: *IEEE Transactions on Neural Networks and Learning Systems.*

Sun, Tao et al. (2022). "SHIFT: A Synthetic Driving Dataset for Continuous Multi-Task Domain Adaptation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21371–21382.

Sun, Xiaoxiao and Liang Zheng (2019). "Dissecting person re-identification from the viewpoint of viewpoint". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 608–617.

Szegedy, Christian et al. (2016). "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826.

Tan, Mingxing and Quoc Le (2019). "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International conference on machine learning.* PMLR, pp. 6105–6114.

Taylor, Geoffrey R, Andrew J Chosak, and Paul C Brewer (2007). “Ovvv: Using virtual worlds to design and evaluate surveillance systems”. In: *2007 IEEE conference on computer vision and pattern recognition*. IEEE, pp. 1–8.

Teichmann, Marvin et al. (2018). “Multinet: Real-time joint semantic reasoning for autonomous driving”. In: *2018 IEEE intelligent vehicles symposium (IV)*. IEEE, pp. 1013–1020.

Torfi, Amirsina, Edward A Fox, and Chandan K Reddy (2022). “Differentially private synthetic medical data generation using convolutional GANs”. In: *Information Sciences* 586, pp. 485–500.

Tsirikoglou, Apostolia (2022). “Synthetic data for visual machine learning: A data-centric approach”. PhD thesis. Linköping University Electronic Press.

Tsirikoglou, Apostolia et al. (2017). “Procedural Modeling and Physically Based Rendering for Synthetic Data Generation in Automotive Applications”. In: *arXiv:1710.06270*.

Tzeng, Eric et al. (2017). “Adversarial discriminative domain adaptation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7167–7176.

Unity (2024). `https://unity.com/`. Online; accessed: 2024-01-14.

Unreal Engine (2024). `https://www.unrealengine.com/en-US/`. Online; accessed: 2021-01-20.

Vahdat, Arash and Jan Kautz (2020). “NVAE: A deep hierarchical variational autoencoder”. In: *Advances in neural information processing systems* 33, pp. 19667–19679.

Van der Maaten, Laurens and Geoffrey Hinton (2008). “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11.

Van Horn, Grant et al. (2015). “Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 595–604.

Volpi, Riccardo, Diane Larlus, and Grégory Rogez (2021). “Continual adaptation of visual representations via domain randomization and meta-learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4443–4453.

Wang, Guoqing et al. (2020). “Cross-domain face presentation attack detection via multi-domain disentangled representation learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6678–6687.

Wang, Miao et al. (2018). “Deep online video stabilization with multi-grid warping transformation learning”. In: *IEEE Transactions on Image Processing* 28.5, pp. 2283–2292.

Wang, Qi et al. (2021). “Pixel-wise crowd understanding via synthetic data”. In: *International Journal of Computer Vision* 129.1, pp. 225–245.

Wang, Ting-Chun, Arun Mallya, and Ming-Yu Liu (2021). “One-shot free-view neural talking-head synthesis for video conferencing”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10039–10049.

Wang, Wenhai et al. (2023). “Internimage: Exploring large-scale vision foundation models with deformable convolutions”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14408–14419.

Wang, Xiangtong et al. (2020). “Exploring the Impacts from Datasets to Monocular Depth Estimation (MDE) Models with MineNavi”. In: *arXiv preprint arXiv:2008.08454*.

Wang, Zhou et al. (2004). “Image quality assessment: from error visibility to structural similarity”. In: *IEEE transactions on image processing* 13.4, pp. 600–612.

Welch, Greg, Gary Bishop, et al. (1995). “An introduction to the Kalman filter”. In.

Wibowo, Mars Caroline, Sarwo Nugroho, and Agus Wibowo (2024). “The Use of Motion Capture Technology in 3D Animation”. In: *International Journal of Computing and Digital Systems* 15.1, pp. 975–987.

Wiseman, Yair (2022). “Autonomous vehicles”. In: *Research Anthology on Cross-Disciplinary Designs and Applications of Automation*. IGI Global, pp. 878–889.

Wrenninge, Magnus and Jonas Unger (2018). “Synscapes: A Photorealistic Synthetic Dataset for Street Scene Parsing”. In: *arXiv:1810.08705*.

Wu, Zhangkai, Longbing Cao, and Lei Qi (2024). “evae: Evolutionary variational autoencoder”. In: *IEEE Transactions on Neural Networks and Learning Systems*.

Xie, Binhui et al. (2022). “Towards Fewer Annotations: Active Learning via Region Impurity and Prediction Uncertainty for Domain Adaptive Semantic Segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8068–8078.

Xu, Jiarui et al. (2022). “Groupvit: Semantic segmentation emerges from text supervision”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18134–18144.

Xu, Qi et al. (2021). “Cdada: A curriculum domain adaptation for nighttime semantic segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2962–2971.

Xu, Yonghao et al. (2019). “Self-ensembling attention networks: Addressing domain shift for semantic segmentation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33, pp. 5581–5588.

Yang, Guorun et al. (2019). “DrivingStereo: A Large-Scale Dataset for Stereo Matching in Autonomous Driving Scenarios”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yang, Suorong et al. (2022). “Image Data Augmentation for Deep Learning: A Survey”. In: *arXiv preprint arXiv:2204.08610*.

Ye, Nianjin et al. (2021). “Motion basis learning for unsupervised deep homography estimation with subspace projection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13117–13125.

Yu, Jiyang and Ravi Ramamoorthi (2018). “Selfie video stabilization”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 551–566.

— (2020). “Learning video stabilization using optical flow”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8159–8167.

Yuan, Yuhui et al. (2019). “Segmentation transformer: Object-contextual representations for semantic segmentation”. In: *arXiv preprint arXiv:1909.11065*.

Yue, Xiangyu et al. (2019). “Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2100–2110.

Zhang, Jiahui et al. (2019). “Learning two-view correspondences and geometry using order-aware network”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5845–5854.

Zhang, Jirong et al. (2020). “Content-aware unsupervised deep homography estimation”. In: *European Conference on Computer Vision*. Springer, pp. 653–669.

Zhang, Lei et al. (2018). “Full-reference stability assessment of digital video stabilization based on riemannian metric”. In: *IEEE Transactions on Image Processing* 27.12, pp. 6051–6063.

Zhao, Hengshuang et al. (2017). “Pyramid scene parsing network”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2881–2890.

Zhou, Bolei, Hang Zhao, Xavier Puig, Sanja Fidler, et al. (2017). “Scene parsing through ade20k dataset”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 633–641.

Zhou, Bolei, Hang Zhao, Xavier Puig, Tete Xiao, et al. (2019). “Semantic understanding of scenes through the ade20k dataset”. In: *International Journal of Computer Vision* 127, pp. 302–321.

Zhou, Kaiyang et al. (2022). “Domain generalization: A survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.4, pp. 4396–4415.