Forschungsbericht 2026-08

Training Deep Learning Models on Synthetic Images

Joachim Rüter

Deutsches Zentrum für Luft- und Raumfahrt
Institut für Flugsystemtechnik
Braunschweig

DLR Deutsches Zentrum für Luft- und Raumfahrt

Forschungsbericht 2026-08

Training Deep Learning Models on Synthetic Images

Joachim Rüter

Deutsches Zentrum für Luft- und Raumfahrt
Institut für Flugsystemtechnik
Braunschweig

188 Seiten
40 Bilder
18 Tabellen
180 Literaturstellen

Deutsches Zentrum
DLR für Luft- und Raumfahrt

Herausgeber:

Deutsches Zentrum
für Luft- und Raumfahrt e. V.
Wissenschaftliche Information
Linder Höhe
D-51147 Köln

ISSN 1434-8454
ISRN DLR-FB-2026-08
Erscheinungsjahr 2026
DOI: 10.57676/xwg7-7q18

Künstliche Intelligenz, Deep-Learning, Training, Synthetische Daten, Sim-to-Real, Generalisierung, Spiele-Engine, Umgebungswahrnehmung, Objekterkennung, Semantische Segmentierung

Joachim RÜTER
DLR, Institut für Flugsystemtechnik, Braunschweig

Training von Deep Learning Modellen mit synthetischen Bildern
Technische Universität Clausthal

Eine präzise Umgebungswahrnehmung ist eine entscheidende Fähigkeit für zukünftige autonome Luftfahrzeuge, beispielsweise zur Identifizierung sicherer Notlandeplätze oder zur Erkennung verschiedener Objekte. Deep-Learning ist der führende Ansatz für viele Wahrnehmungsaufgaben, wie Objekterkennung und semantische Segmentierung in Bildern. Allerdings benötigen Deep-Learning-Modelle umfangreiche und diverse Trainingsdatensätze um eine hohe Genauigkeit zu erreichen, was in sicherheitskritischen Anwendungen wie der Luftfahrt eine Herausforderung darstellt.

Auf der anderen Seite haben Spiele-Engines große Fortschritte bei der Simulation visuell realistischer, virtueller Umgebungen gemacht und erlauben die Extraktion synthetischer Bilder. Die Verwendung solcher für das Training könnte den Bedarf an realen Daten reduzieren. Allerdings haben Deep-Learning-Modelle oft Schwierigkeiten von synthetischen auf reale Bilder zu verallgemeinern und die Faktoren, die diese sim-to-real Generalisierungsprobleme beeinflussen, sind nicht vollständig geklärt.

Vor diesem Hintergrund erforscht diese Arbeit das Potenzial synthetischer Bilder aus Spiele-Engines, Deep-Learning-Modelle zur Umgebungswahrnehmung für ausgewählte Luftfahrt-Anwendungen zu trainieren. Zusätzlich werden die Auswirkungen der Modellarchitektur auf die sim-to-real Generalisierungsfähigkeit sowie die Nutzbarkeit verschiedener Bild-Augmentierungen zur Verbesserung der Generalisierung umfassend untersucht.

Dazu wird eine umfangreiche Evaluation verschiedener Anwendungsfälle und Einflussfaktoren durchgeführt, bei der unterschiedliche Wahrnehmungsaufgaben und Datensätze betrachtet werden. Als Bewertungsmetriken dienen die Genauigkeit auf realen Bildern und der sim-to-real gap. Die Ergebnisse basieren auf über 1.700 Modell-Varianten, die auf synthetischen Bildern trainiert wurden.

Diese Arbeit zeigt, dass synthetische Bilder eine geeignete Grundlage für das Training von Deep-Learning-Modellen zur Umgebungswahrnehmung in der Luftfahrt sein können, z.B. zur Erkennung von Tankkörben bei der Luft-zu-Luft-Betankung oder zur semantischen Segmentierung aus der Perspektive eines tiefffliegenden Luftfahrzeugs. Sie zeigt weiterhin, dass Deep-Learning-Modelle sich deutlich in ihrer sim-to-real Generalisierungsfähigkeit unterscheiden, betont den starken Einfluss des Modell-Backbones und identifiziert weitere Einflussfaktoren, die praktische Erkenntnisse für die Modellauswahl liefern. Zusätzlich wird aufgezeigt, dass die Anwendung von Bild-Augmentierungen, insbesondere solcher mit Einfluss auf die Farbgebung, die Generalisierungsfähigkeit der Modelle verbessert. Dies betont den Einfluss von Farbunterschieden auf die sim-to-real Generalisierungsprobleme und präsentiert eine einfache Methode, um deren Auswirkung zu reduzieren.

Insgesamt liefert diese Arbeit neue Perspektiven, ein tieferes Verständnis des Potenzials und der Grenzen von synthetischen Daten für das Training von Deep-Learning-Modellen zur Umgebungswahrnehmung und praktische Erkenntnisse für die Modellauswahl und das Training. Sie zeigt das Potenzial von synthetischen Daten, die Notwendigkeit einer teuren und zeitaufwändigen Erfassung realer Bilddaten zu reduzieren, und trägt dazu bei Fortschritte in der Deep-Learning-basierten Umgebungswahrnehmung in zukünftige autonome Luftfahrzeuge zu überführen. Obwohl sich diese Arbeit auf die Luftfahrt konzentriert, untersucht sie auch einen Automobil-Datensatz. Diese breiten Untersuchungen ermöglichen es, die Ergebnisse auch auf andere Branchen mit begrenztem Zugang zu realen Daten, wie z.B. die See- oder Raumfahrt, zu übertragen und beschleunigen insgesamt die Nutzbarkeit von Deep-Learning-basierter Umgebungswahrnehmung in neuen Anwendungsdomänen.

Artificial Intelligence, Deep Learning, Training, Synthetic Data, Sim-to-Real, Generalization, Game Engine, Environment Perception, Object Detection, Semantic Segmentation

(Published in english)

Joachim RÜTER
German Aerospace Center (DLR), Institute of Flight Systems, Brunswick

Training Deep Learning Models on Synthetic Images

Clausthal University of Technology

Accurate environment perception is a crucial capability for future autonomous aircraft systems, e.g., to identify safe emergency landing sites or to detect various objects of interest. Deep learning has become the state-of-the-art approach for many perception tasks, such as object detection and semantic segmentation in images. However, deep learning models require large and diverse training datasets to achieve high accuracy, which is a major challenge in safety-critical domains like aviation, where data collection is often limited by regulatory, safety, ethical, and economical constraints.

On the other hand, game engines have made significant progress in simulating visually realistic virtual environments, enabling the extraction of diverse synthetic images. Using such synthetic images for training promises to reduce the need for real-world data in many domains, but deep learning models often struggle to generalize well from synthetic to real-world images, and factors influencing this sim-to-real generalization issue are not yet fully understood.

Against this background, this work explores the potential of synthetic images extracted from game engines to train deep learning models for selected aviation perception use-cases. Additionally, it comprehensively investigates the influence of deep learning model architectures on their sim-to-real generalization capability and the ability of image augmentation techniques to improve sim-to-real generalization.

To do so, an extensive evaluation of various use-cases and factors influencing the effectiveness of synthetic data is conducted, utilizing multiple perception tasks and datasets. The evaluations employ the overall real-world performance and the sim-to-real gap as the main evaluation metrics, and the results are based on over 1,700 deep learning model variations trained on synthetic images and evaluated on real-world ones.

This work demonstrates that synthetic images can be a suitable basis for training deep learning models for various aviation use-cases, such as drogue detection during air-to-air refueling and semantic segmentation from a low-altitude aerial perspective. It shows that deep learning models differ significantly in their generalization capability and highlights the strong influence of the model backbone. Further influencing factors are identified, providing practical insights for model selection. Additionally, the application of image augmentation techniques during training, particularly color augmentations, is found to significantly improve the generalization capability of deep learning models trained on synthetic images, stressing the impact of color differences on sim-to-real generalization issues, and showcasing an easy-to-use method to reduce its effect.

Overall, this work provides novel perspectives, a deeper understanding of the potential and limitations of using synthetic images for training deep learning models, and practical insights for model selection and training. It demonstrates the potential of synthetic data to reduce the need for expensive and time-consuming real-world data collection, helping to bring advances in deep learning-based perception into future autonomous aircraft. While this work focuses on aviation, the experiments also include an automotive dataset. These broad investigations further allow the transfer of the findings to other industries with limited access to real-world data, such as maritime or space, ultimately accelerating the deployment of deep learning-based perception to new domains.

Training Deep Learning Models on Synthetic Images

Doctoral Thesis
(Dissertation)

to be awarded the degree
Doctor of Engineering (Dr.-Ing.)

submitted by
Joachim Rüter
from Minden (Germany)

approved by the
Faculty of Mathematics/Computer Science
and Mechanical Engineering at
Clausthal University of Technology

Date of oral examination:
27.02.2026

Dean:
Prof. Dr.-Ing. Armin Lohrengel

Chairperson of the Board of Examiners:
Prof. Dr. Robert Bredereck

Supervisor:
Prof. Dr.-Ing. Umut Durak

Reviewers:
Prof. Dr. Sven Hartmann
Prof. Dr.-Ing. Zamira Daw

Danksagung

Diese Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Flugsystemtechnik in der Abteilung Unbemannte Luftfahrzeuge des Deutschen Zentrums für Luft- und Raumfahrt e.V. (DLR) in Braunschweig.

An erster Stelle richtet sich mein Dank an meinen Doktorvater Prof. Dr.-Ing. Umut Durak für die Betreuung dieser Arbeit, sein wertvolles Feedback und die unkomplizierten und prompten Ratschläge bei allen Fragen. Prof. Dr. Sven Hartmann und Prof. Dr.-Ing. Zamira Daw danke ich in ihrer Funktion als Gutachter und Gutachterin sowie Prof. Dr. Robert Bredereck als Vorsitzendem der Prüfungskommission für ihr Interesse an meiner Arbeit.

Allen internen und externen Reviewern meiner Publikationen sowie dieser Dissertation danke ich für ihr konstruktives Feedback und die anregenden Diskussionen, die zur Verbesserung dieser Arbeit sowie zur Weiterentwicklung meines wissenschaftlichen Denkens beigetragen haben.

Ein großer Dank gilt Johann Dauer, der mir in seiner Funktion als Abteilungsleiter den Freiraum gegeben hat, mir ein Umfeld zu schaffen, in dem ich kontinuierlich an dieser Dissertation arbeiten konnte. Sein Feedback aus einer fachlich anderen Perspektive hat mir neue Denkanstöße gegeben und meine Entwicklung als Wissenschaftler gefördert. Ebenso danke ich meinem fachlichen Betreuer Dr.-Ing. Stefan Krause für die zahlreichen Diskussionen, seine weitreichende Unterstützung und seine stets konstruktiven Rückmeldungen. Mein Dank gilt auch all meinen Kolleginnen und Kollegen für die offene Diskussionskultur sowie ihre stetige Bereitschaft zur Unterstützung. Ebenso danke ich all meinen

ehemaligen Studentinnen und Studenten, die besonders bei der Implementierung sowie dem Aufbau der Tools wichtige Beiträge zum Fortschritt dieser Arbeit geleistet haben.

Ganz besonderer Dank gilt meinen Eltern Elfi und Eckhard Rüter, die mich stets begleitet, unterstützt und gefördert haben, mir immer mit Rat und Tat zur Seite standen und mich ermutigt haben, meinen Weg zu gehen.

Ein besonders herzlicher Dank gilt auch meiner Frau Lisa Rüter für ihr Verständnis, ihre unermüdliche Unterstützung in allen Belangen, ihre liebevolle Begleitung und das wohl akribischste Review dieser Arbeit.

Meiner Familie, meinen Freundinnen und Freunden, und allen, die sich an dieser Stelle zu Unrecht nicht wiederfinden, danke ich für die Unterstützung, die mir entgegengebracht wurde.

Herzlichen Dank!

im Oktober 2025,
Joachim Rüter

Abstract

Accurate environment perception is a crucial capability for future autonomous aircraft systems, e.g., to identify safe emergency landing sites or to detect various objects of interest. Deep learning has become the state-of-the-art approach for many perception tasks, such as object detection and semantic segmentation in images. However, deep learning models require large and diverse training datasets to achieve high accuracy, which is a major challenge in safety-critical domains like aviation, where data collection is often limited by regulatory, safety, ethical, and economical constraints.

On the other hand, game engines have made significant progress in simulating visually realistic virtual environments, enabling the extraction of diverse synthetic images. Using such synthetic images for training promises to reduce the need for real-world data in many domains, but deep learning models often struggle to generalize well from synthetic to real-world images, and factors influencing this sim-to-real generalization issue are not yet fully understood.

Against this background, this work explores the potential of synthetic images extracted from game engines to train deep learning models for selected aviation perception use-cases. Additionally, it comprehensively investigates the influence of deep learning model architectures on their sim-to-real generalization capability and the ability of image augmentation techniques to improve sim-to-real generalization.

To do so, an extensive evaluation of various use-cases and factors influencing the effectiveness of synthetic data is conducted, utilizing multiple perception tasks and datasets. The evaluations employ the overall real-world performance

and the sim-to-real gap as the main evaluation metrics, and the results are based on over 1,700 deep learning model variations trained on synthetic images and evaluated on real-world ones.

This work demonstrates that synthetic images can be a suitable basis for training deep learning models for various aviation use-cases, such as drogue detection during air-to-air refueling and semantic segmentation from a low-altitude aerial perspective. It shows that deep learning models differ significantly in their generalization capability and highlights the strong influence of the model backbone. Further influencing factors are identified, providing practical insights for model selection. Additionally, the application of image augmentation techniques during training, particularly color augmentations, is found to significantly improve the generalization capability of deep learning models trained on synthetic images, stressing the impact of color differences on sim-to-real generalization issues, and showcasing an easy-to-use method to reduce its effect.

Overall, this work provides novel perspectives, a deeper understanding of the potential and limitations of using synthetic images for training deep learning models, and practical insights for model selection and training. It demonstrates the potential of synthetic data to reduce the need for expensive and time-consuming real-world data collection, helping to bring advances in deep learning-based perception into future autonomous aircraft. While this work focuses on aviation, the experiments also include an automotive dataset. These broad investigations further allow the transfer of the findings to other industries with limited access to real-world data, such as maritime or space, ultimately accelerating the deployment of deep learning-based perception to new domains.

Kurzfassung

Eine präzise Umgebungswahrnehmung ist eine entscheidende Fähigkeit für zukünftige autonome Luftfahrzeuge, beispielsweise zur Identifizierung sicherer Notlandeplätze oder zur Erkennung verschiedener Objekte. Deep-Learning ist der führende Ansatz für viele Wahrnehmungsaufgaben, wie Objekterkennung und semantische Segmentierung in Bildern. Allerdings benötigen Deep-Learning-Modelle umfangreiche und diverse Trainingsdatensätze um eine hohe Genauigkeit zu erreichen, was in sicherheitskritischen Anwendungen wie der Luftfahrt eine Herausforderung darstellt.

Auf der anderen Seite haben Spiele-Engines große Fortschritte bei der Simulation visuell realistischer, virtueller Umgebungen gemacht und erlauben die Extraktion synthetischer Bilder. Die Verwendung solcher für das Training könnte den Bedarf an realen Daten reduzieren. Allerdings haben Deep-Learning-Modelle oft Schwierigkeiten von synthetischen auf reale Bilder zu verallgemeinern und die Faktoren, die diese sim-to-real Generalisierungsprobleme beeinflussen, sind nicht vollständig geklärt.

Vor diesem Hintergrund erforscht diese Arbeit das Potenzial synthetischer Bilder aus Spiele-Engines, Deep-Learning-Modelle zur Umgebungswahrnehmung für ausgewählte Luftfahrt-Anwendungen zu trainieren. Zusätzlich werden die Auswirkungen der Modellarchitektur auf die sim-to-real Generalisierungsfähigkeit sowie die Nutzbarkeit verschiedener Bild-Augmentierungen zur Verbesserung der Generalisierung umfassend untersucht.

Dazu wird eine umfangreiche Evaluation verschiedener Anwendungsfälle und Einflussfaktoren durchgeführt, bei der unterschiedliche Wahrnehmungsauf-

gaben und Datensätze betrachtet werden. Als Bewertungsmetriken dienen die Genauigkeit auf realen Bildern und der sim-to-real gap. Die Ergebnisse basieren auf über 1.700 Modell-Varianten, die auf synthetischen Bildern trainiert wurden.

Diese Arbeit zeigt, dass synthetische Bilder eine geeignete Grundlage für das Training von Deep-Learning-Modellen zur Umgebungswahrnehmung in der Luftfahrt sein können, z.B. zur Erkennung von Tankkörben bei der Luft-zu-Luft-Betankung oder zur semantischen Segmentierung aus der Perspektive eines tiefffliegenden Luftfahrzeugs. Sie zeigt weiterhin, dass Deep-Learning-Modelle sich deutlich in ihrer sim-to-real Generalisierungsfähigkeit unterscheiden, betont den starken Einfluss des Modell-Backbones und identifiziert weitere Einflussfaktoren, die praktische Erkenntnisse für die Modellauswahl liefern. Zusätzlich wird aufgezeigt, dass die Anwendung von Bild-Augmentierungen, insbesondere solcher mit Einfluss auf die Farbgebung, die Generalisierungsfähigkeit der Modelle verbessert. Dies betont den Einfluss von Farbunterschieden auf die sim-to-real Generalisierungsprobleme und präsentiert eine einfache Methode, um deren Auswirkung zu reduzieren.

Insgesamt liefert diese Arbeit neue Perspektiven, ein tieferes Verständnis des Potenzials und der Grenzen von synthetischen Daten für das Training von Deep-Learning-Modellen zur Umgebungswahrnehmung und praktische Erkenntnisse für die Modellauswahl und das Training. Sie zeigt das Potenzial von synthetischen Daten, die Notwendigkeit einer teuren und zeitaufwändigen Erfassung realer Bilddaten zu reduzieren, und trägt dazu bei Fortschritte in der Deep-Learning-basierten Umgebungswahrnehmung in zukünftige autonome Luftfahrzeuge zu überführen. Obwohl sich diese Arbeit auf die Luftfahrt konzentriert, untersucht sie auch einen Automobil-Datensatz. Diese breiten Untersuchungen ermöglichen es, die Ergebnisse auch auf andere Branchen mit begrenztem Zugang zu realen Daten, wie z.B. die See- oder Raumfahrt, zu übertragen und beschleunigen insgesamt die Nutzbarkeit von Deep-Learning-basierter Umgebungswahrnehmung in neuen Anwendungsdomänen.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Description
AI	Artificial Intelligence.
AP	Average Precision.
CNN	Convolutional Neural Network.
CSO	Cochstedt airport.
EASA	European Union Aviation Safety Agency.
FOV	Field of view.
FPN	Feature Pyramid Network.
FPS	Frames per second.
IoU	Intersection over Union.
mAP	Mean Average Precision.
mIoU	Mean Intersection over Union.
MLN	Multi Level Neck.
MSL	Mean sea level.
NATO	North Atlantic Treaty Organization.
ODD	Operational Design Domain.
pp	Percent points.
SGD	Stochastic Gradient Descent.
SRD	Standards Related Document.
UAS	Unmanned Aircraft System.

1 Introduction

1.1 Motivation

Future autonomous aircraft systems require exceptional environment perception capabilities, particularly when operating without human oversight. To ensure safe and effective operation, various tasks rely on accurate information about the aircraft's surroundings, including planning collision-free flight paths, performing air-to-air refueling (Fig. 1.1a), interacting with other aerial objects during manned-unmanned teaming (Fig. 1.1b), and identifying safe emergency landing sites (Fig. 1.1c).

A natural approach to visual environment perception is to utilize RGB cameras, which produce images similar to human vision. To enrich these raw images with semantic information, deep learning models, i.e. multi-layer neural network architectures, have achieved state-of-the-art results in many domains. However, to achieve these results, deep learning models require a vast amount of diverse images for training. Furthermore, the images should be representative of the situations faced during deployment and should cover the entire operational envelope of the aircraft.

This need for extensive datasets poses a significant challenge in safety-critical domains and prevents the deployment of deep learning models. This particularly applies to aerospace, as collecting and labeling data from flying aircraft is expensive, time-consuming, and sometimes dangerous or even impossible. These challenges are further exacerbated in scenarios where aerial vehicles operate in close formation with manned or unmanned systems, such as aerial

(a) **Air-to-air refueling. Image credit: [1]**

(b) **Manned-unmanned teaming of helicopters. Image credit: [2]**

(c) **Emergency landing site identification. Image credit: [3]**

Figure 1.1: **Example tasks of future autonomous aircraft systems that require strong environment perception capabilities**

refueling, which can result in severe damage [4–7] or even fatalities [8] from time-to-time. Furthermore, recording data from a low-flying Unmanned Aircraft System (UAS) perspective over urban areas or humans is often restricted in the European Union regulatory framework due to safety concerns [9]. Even when data could be recorded, the manual annotation of images can be extremely time-consuming, requiring up to 60 to 90 minutes per image for semantic segmentation datasets in an automotive setting [10, 11].

To address the scarcity of real-world images, the use of synthetic images generated by increasingly realistic visual simulation environments has gained more

and more attention. Modern game engines can simulate almost any situation, extract visual sensor data, and provide ground truth annotations at relatively low costs. This approach further mitigates safety and regulatory concerns, enabling the generation of diverse data, including edge and corner cases. The European Union Aviation Safety Agency (EASA) even mentions synthetic data in their plans for certifying artificial intelligence components for aviation [12].

While training on synthetic images is a promising approach to reduce the need for real-world data, deep learning models trained solely on synthetic images regularly struggle to generalize well to real-world images, resulting in a low performance. This generalization problem often makes these models unsuitable for real-world deployment and is therefore a pressing challenge. However, the underlying causes for this phenomenon are not yet fully understood and may stem from multiple sources, including the model, data, and training process, as explained in Section 1.2. The difficulties to identify reasons and influencing factors arise from, e.g., the complexity of neural networks and the lack of comprehensive descriptions of image similarity, particularly in complex and cluttered scenes.

Against this background, this work investigates the suitability of synthetic images for training deep learning models in aviation as well as influences on the sim-to-real generalization capability of the deep learning models. The findings provide insights into the effectiveness and limitations of synthetic images for training deep learning models, contributing to the wider adoption of advances in deep learning in domains with limited access to visual data, such as the aerospace industry. Furthermore, the results will form a basis for future research on the use of synthetic visual data for deep learning model training in various domains beyond aerospace.

After an overview of the state-of-the-art in Section 1.2, the research questions considered in this dissertation are presented in Section 1.3, followed by an outline of the remaining structure of this work in Section 1.4.

1.2 Related Work

1.2.1 Synthetic Images for Training Deep Learning Models

Using synthetic images from game engines to train deep learning models has received rising attention in recent years. Researchers investigated its use for object detection [13–27], semantic segmentation [18, 28–33], classification [34], multi object tracking [14, 35] as well as viewpoint estimation [36], among others. The investigations range from the autonomous driving domain for tasks like pedestrian and vehicle detection [14, 17, 18, 28–31], safety monitoring of construction workers [16], and drone detection from static cameras [15] to UAS detection tasks for people, vehicles, and animals [21–27], aircraft detection from satellite perspective [19] and more. The growing interest and research are fueled by advancements in high-quality game engines such as the Unreal Engine [37], UNITY [38], and the one from Grand Theft Auto V [39] in combination with corresponding data extraction methods and tools like CARLA [33, 40], AirSim [41], Sim4CV [42], UnrealCV [43] and the one presented in [28]. Furthermore, published synthetic datasets such as SYNTHIA [31], Virtual KITTI [35], Synscapes [18], RarePlanes [19] and the one presented in [28] drove the research.

Because of the potential benefits of using synthetic images, its usage is even recommended by the task force on Artificial Intelligence (AI) from EASA in [12]:

> "[T]he design and testing of safety-critical machine learning models should rely on simulated/synthesized data, for which new data can be acquired at a very low cost once a system is setup."

But even when the data may look very realistic, one has to keep in mind that it is not the data the model will face when deployed. Against this background, EASA adds that "synthetic data should never be used without proper analysis and mitigation of the domain biases, no matter how realistic it looks" [12]. Recent research shows that this indication is a very important one, as using synthetic data for training is not as straightforward as it may seem. While

synthetic data may appear realistic at first glance, subtle discrepancies that may not be immediately noticeable to humans (e.g., incorrect shadows or unnatural edge sharpness) can lead to a low model performance.

Specifically, it is shown for semantic environment perception tasks that neural networks have problems generalizing to real-world images when trained only on game engine data resulting in a performance degradation on the real-world evaluation images. This problem is shown, e.g., for human detection from overhead cameras [13], drone detection [15], object detection in street scenes [18], aircraft detection from satellite perspective [19], as well as animal, vehicle and human detection from UAS perspective [21, 23, 24]. Furthermore, it is also shown for semantic segmentation in urban scenes [18, 28, 31] as well as from UAS perspective [33]. While the phenomenon is known under different names such as domain gap [44], reality gap [23], simulation-reality gap [15], simulation-to-reality gap [45], and Sim2Real gap [46], it is referred to as sim-to-real gap in this work.

With respect to the sim-to-real gap, [47] states that "synthetic datasets suffer from a specificity problem which results in models that are incapable of proper generalization. They perform very well on their own test set, however their performance suffers on any other test set". This phenomenon is actually not new but known for neural networks in general as a generalization problem of models under dataset shift [48, 49]. According to [49], dataset shift occurs when the training data is not strictly representative of the test data. Therefore, the sim-to-real gap is closely related to this broader topic. In this direction, some research even states that the sim-to-real gap is comparable to the shift between different real-world datasets [36, 50].

On the other hand, a large number of works have shown that synthetic images have high potential as the gap can be smaller than that of models trained on real-world images from a slightly different domain [14, 22, 51]. Notably, [17] shows that for vehicle detection in urban scenes, a model trained on their synthetic dataset achieves a better performance on the real-world KITTI dataset [52] than when the same model is trained on the real-world Cityscapes [10] dataset. Similarly, [22] compares a car detection model from UAS perspective trained on the VisDrone dataset [53] to a model trained solely on synthetic images. The

models are evaluated on the real-world dataset AU-AIR [54] which was recorded in a different geographical region than the VisDrone dataset. Interestingly, both models achieve similar performance, with the model trained solely on synthetic image slightly outperforming the model trained on real-world VisDrone data, despite having never seen real-world images during training.

Moreover, [31] even shows that when real-world data is very limited, models trained on synthetic images can achieve comparable results to those trained on real-world ones from the same dataset it is tested on. The authors utilize the game engine UNITY [38] to generate synthetic images for semantic segmentation in driving scenes. Notably, their experiments show that a neural network trained solely on synthetic data can reach performance levels nearly as good as a model trained on real-world data. However, it is important to consider that the real-world training dataset used in this comparison was relatively small, consisting of only 200-300 images, which may not be representative of larger, more diverse datasets.

1.2.2 Influences on the Sim-to-Real Generalization

While synthetic images have shown promising results for deep learning model training, the generalization problems from synthetic to real-world images are a prevailing issue. Although many works explored different applications and use-cases as described above, only few works investigated reasons and influencing factors on this generalization problem. On an abstract level, they could come from all three main components of a deep learning system, namely *data*, *models* and *learning* [55]. As this work considers only supervised learning, learning and training are used as synonyms. The following paragraphs present the related work on reasons and influences structured along these three components. Finding reasons and influences stemming from these components is not trivial. For example, a problem on model side is that neural networks are considered black boxes for which it is currently not fully understood how they come to their conclusions. On data side, there are no concise descriptions of overall image similarity, especially when the images get relatively complex and cluttered.

Data

When investigating synthetic data, [56] argues that the influences of the sim-to-real gap can be divided into the two components *appearance gap* and *content gap* which is further attested by [24]. The content gap is arising from differences in the distribution or diversity of the synthetic data compared to the real one. As an example, one could argue that houses in New York have different heights and appearances than those in suburbs in Germany and that the number of cars on the street is probably different. Therefore, a dataset modeling Germany may not reflect the distribution the model needs for training when it should be deployed in New York.

The term appearance gap is used to describe the gap resulting from stylistic differences between the synthetic images and the real ones. It includes pixel-level differences which could result from deviations in color-richness, texture-diversity, materials or missing capabilities of the rendering engine. It is reasonable to investigate the appearance gap as, although synthetic images look more and more realistic, differences can often still be seen with the bare eye.

However, if the extracted images looked exactly like their real counterparts, the appearance gap would be eliminated and a model should have no problem generalizing to the real data. Therefore, it is a natural step to look at the photorealism of the images extracted from game engines when using them for machine learning training. Some authors even see "strong evidence" that high realism of the synthetic data is important as "[t]here are no indications that neural networks are able to naturally abstract away domain shift" [18]. Furthermore, [24] shows that the appearance gap has influence on the model performance by minimizing the content gap as much as possible and investigating the remaining performance drop. In a similar direction, [57] tries to measure the realism of synthetic images based on some image metrics. The work shows that using different rendering parameters to optimize the synthetic image generation can improve the resulting model performance on real-world images in a very controlled environment.

Nevertheless, increasing photorealism is a complex challenge that has been

a long-standing focus of research and development in the gaming industry. Therefore, achieving significant advancements seems infeasible for industry users and practitioners in the near future. Furthermore, improving graphics quality usually comes with high computational demand and high amounts of manual labor for graphic artists. As [21] shows that the graphics settings only have a small influence on the performance drop, it is also unclear whether synthetic images really have to look photorealistic and have to mirror all aspects of the real-world or whether the models pay attention to different aspects than humans.

In this direction, there are multiple works investigating different image property metrics to measure the differences between the synthetic and real-world images as well as influences on the sim-to-real gap. In [58], the authors consider grouped image properties such as Color, Brightness/Luminance/Contrast, and Distortion/Blur/Noise and show that real and synthetic images have differences on these levels. For the KITTI dataset [59] and its synthetic counterpart VKITTI [35], they show that even one metric for noise is sufficient to differentiate the datasets as the synthetic images contain almost no noise. They achieve similar differentiation results for their real-world and synthetic images from UAS perspective and conclude that the general coloration and lighting conditions as well as a lack of noise and fine structures in the simulation have the greatest influence on the sim-to-real gap.

Going deeper into the investigation of the influence of noise and proper sensor modeling, [51] models different sensor effects including Chromatic Aberration, Blur, Exposure, Sensor Noise and Color Shift. Using a Faster R-CNN object detection model and the real-world KITTI dataset as well as the synthetic VKITTI and a synthetic GTA-V dataset, they show that each effect as well as the combination of all improves the performance compared to synthetic images without these sensor effects. In [60], the authors model sensor lens artifacts for the synthetic training data and use the Wasserstein distance to optimize sensor modeling parameters between the synthetic and the real-world images. Doing so, the work improves the mean Intersection over Union (mIoU) for a semantic segmentation network trained on the synthetic images. The benefits of modeling camera vignetting are shown in [61]. Modeling it on the VKITTI dataset, a car detector improves on the real-world KITTI dataset by more than

5% compared to the original synthetic data. In [13], performance improvements when matching the distortion of the real-world and the simulation camera are shown. Generally, [62] states that image augmentation can improve the performance of an object detector on real-world images but no details on the augmentations used or further evaluations are presented.

When looking at more subjective features, [21] states that some object classes look different in synthetic and real-world datasets and that the appearance gap may be reduced by editing these objects to look more realistic. In detail, there is research suggesting that the shape of certain objects in game engines looks realistic but the texture does not, preventing the model from finding some of the learned features in the real-world images. Experimentally, it is shown in [30] that the shape of *foreground classes* like persons and cars is realistic but the texture is not. To do so, the authors train a binary neural network classifier to differentiate whether the silhouette of an object comes from a real-world image or from an image extracted from a game engine. On the other hand, they state that *background classes* without well-defined shapes, such as the sky, look comparably realistic in synthetic images.

A factor on which many researchers agree is the necessity of high diversity and variety in the synthetic training data. For example, [34] finds that the variation of synthetic data is important as maximum variation leads to maximum performance. This is an important finding as some research states that the diversity of synthetic data is generally lower than that of real-world data. In this direction, [17] states that the training value of a synthetic image is smaller than that of a real-world one as a neural network needs more synthetic images to get a comparable performance. The authors argue that this is because of the lower variation of the synthetic images. The authors of [23] argue in a similar direction for their use-case of vehicle detection as they state that the variance of their 80 3D car models used in their simulation is obviously lower than that of the 2,700 vehicle instances in the real-world test dataset UAVDT [63]. Similarly, [14] states that for the task of pedestrian detection, the diversity of the synthetic data is far more important than the amount of synthetic data. In their experiments, they tried to maximize diversity by generating over 9,519 unique pedestrians by randomizing clothes, backpacks, bags, masks, hair and beard styles. Furthermore, they randomized the weather conditions from clear,

extra sunny, cloudy, overcast, rainy, thunder, smog, foggy, and blizzard and recorded the conditions at night as well as day, resulting in 768 generated sequences. In the same line of reasoning, [47] states that the diversity of the data is more important than photorealism.

Some approaches aim to close the appearance gap using image stylization or (unpaired) image-to-image translation like CycleGAN [64]. These approaches employ neural networks that try learning to transform the synthetic images to look more like the real-world images while persevering their content. By doing so, generated annotations can still be used for training but the sim-to-real gap may be reduced. In the past years, some variations of this general idea have been developed [65]. As this family of approaches needs real-world data during training which is not assumed to exist in this work, they will not be investigated further.

Model

While closely related to the training process and the data, it is reasonable to also look at the model architecture itself when investigating influences on the sim-to-real generalization problems of deep learning models. It is known that some models have better detection performance than others [66,67]. Therefore, there may also be differences in the model's ability to generalize from synthetic to real-world images. A notable example is illustrated by a study on person detection in artwork, where a YOLO model demonstrated a smaller decline in performance when trained on real-world data and applied to artwork, compared to other models [68]. This observation suggests that some architectures may be more robust to domain shifts, motivating the further investigation of the generalization capability of different models across simulated and real-world environments.

There are not many works that explicitly look at differences in model generalization capability. However, implicitly, there are some works training various models and achieving different detection results between them but they do not further analyze these differences, nor investigate influences [14, 21, 31]. For

instance, [21] implicitly shows that a YOLO model for cattle detection seems to be better at generalizing from synthetic training to real-world evaluation images compared to the other models investigated. However, there is no interpretation nor systematic study.

Investigating the influence of the model architecture on the sim-to-real generalization has the potential to lead to interesting results. For example, recent studies revealed that vision transformers and Convolutional Neural Networks (CNNs) learn different features [69]. Furthermore, vision transformers demonstrated to be better at generalizing [70] and to be more robust to domain shifts [71]. CNNs tend to rely more on texture for decision-making [70–72], whereas vision transformers are less texture-biased but rely more on shape information [70, 71]. Combined with the assumption made by the literature presented above, that the shape of certain objects in game engines looks realistic but the texture does not, vision transformers appear to hold promise for superior generalization capabilities from synthetic to real-world images. Although this hypothesis has not been extensively studied, preliminary findings from a recent study [62] provide supporting evidence. The study demonstrates that vision transformers are able to outperform CNNs on real-world images when trained on the same synthetic data, suggesting that vision transformers may be better suited for bridging the gap between simulated and real-world environments.

Training

The training process of the deep learning model has a significant influence on its final performance. Therefore, different training approaches have been presented with the aim to improve the sim-to-real generalization as presented below. However, as most of these approaches require labeled or unlabeled real-world data during training, which are not available in the setting considered in this work, they are only briefly described.

Many works consider the setting in which a combination of synthetic images and labeled real-world training images is available. In this scenario, different

strategies for integrating and balancing the synthetic and the real-world images were investigated.

In this setting, one approach is to train on a dataset that is a mixture of the synthetic and the real-world images. Some works show that training on this mixture improves performance compared to training only on the real-world images for object detection [15, 23, 24], semantic segmentation [31, 33] and viewpoint estimation [36]. While they show the benefits of mixing real-world and synthetic images, [23] further demonstrates that the mixing ratio also has an influence as using too many synthetic images deteriorates the performance. They achieve the best results with a proportion of 20% synthetic data.

In the same setting, other works consider pre-training the models on synthetic images followed by fine-tuning on real-world images and again show an improved performance compared to training solely on real-world images for object detection [15, 18, 21, 25], semantic segmentation [18, 29, 32, 33] and multi-object tracking [35]. However, this does not need to hold for every object class [18] and the performance improvement seems to get smaller when the size of the real-world fine-tuning dataset increases [21].

When comparing mixed training to fine-tuning, [47] states that fine-tuning models pre-trained on synthetic data leads to better results than mixed training for an object detection use-case. Some works even try to combine both approaches by pre-training the model on a mixed dataset followed by fine-tuning on real-world images and also achieve improved results compared to training solely on real-world images [28].

Besides performance improvements, the required amount of real-world training images can be reduced when used in combination with synthetic ones [15, 19, 28, 47]. For example, in the context of aircraft detection from satellite images, [19] shows that an object detection model loses only little performance when using just 10% of the available real-world images but adding synthetic images during training. For drone detection, [15] states that the inclusion of just 1% of the real-world images into the synthetic training datasets results in a performance comparable to exclusive real-world training data. For semantic segmentation in street scenes, [28] achieves comparable performance with syn-

thetic data and only one-third of the real-world images compared to exclusive real-world data training.

When no labeled real-world images but only unlabeled ones are available during training, the given problem setting is generally known as Domain Adaptation [73, 74]. This is a research field independent of synthetic data but more general for different source and target data. It considers the problem of having the same task on the source and the target data but having a slightly different distribution between the datasets. There are several approaches in research to face this problem as shown in survey papers [75–77]. For example, it is possible to incentivize the model during training to learn feature representations that are independent of the source and target domain. As these approaches need real-world data during training, which is not assumed to be available in this work, and are often not specific to synthetic data, they will not be presented and investigated further.

When no real-world data is available during training, domain randomization [78] is a prominent approach which is specifically designed for using synthetic data for training. While it is a natural approach to make the simulation look more realistic as explained above, in domain randomization, a large amount of variation is introduced in the simulation for example by changing the textures of the objects and adding distractors. The resulting scenes may on purpose not look realistic anymore but give the deep learning model a wide range of data to learn form. When the model is applied to real-world target data, the real-world data may be seen as just another sample of the large set of variations faced during training.

This approach is also closely related to the observation in the literature that the texture of some objects in simulation seems to not look as realistic as their shape. By swapping the textures of the objects repeatedly, the model cannot rely on the texture to detect them but has to learn other features, e.g., their form. Domain randomization leads to promising real-world performance for some domains and therefore supports the claim of realistic form but unrealistic textures of foreground objects in game engines as presented above.

While one of the first works applies domain randomization to the task of object

localization for robot grasping of objects on a table [78], it was also applied to various object detection tasks. For example, [79] considers the task of detecting cars in the KITTI dataset [52] and shows that training a Faster R-CNN model on domain randomized data achieves better detection performance than a model trained on the synthetic VKITTI dataset [35], which was built to closely resemble KITTI. However, the detection performance is still much worse than when the model is trained on the real-world target data of the KITTI dataset directly. When pre-trained on the domain randomized data and fine-tuned on real-world KITTI target data, the model even outperforms a model trained on the target data alone as well as a model pre-trained on VKITTI data instead of the domain randomized data. There are also more advanced forms of domain randomization and deeper investigations in recent literature [80–82].

As current aviation certification efforts for deep learning focus on the concept of Operational Design Domain (ODD) and on ensuring that the training and evaluation data have the same distribution [83], the author expects that intentionally waiving photorealism will currently lead to certification hurdles. However, randomizing different parameters in realistic ranges seems to be a promising approach to improve the synthetic images and models trained on them, as presented in the data section above.

1.3 Research Questions

Building on the motivation presented in Section 1.1 and the review of related work in Section 1.2, this section outlines the specific research questions considered in this work. They all contribute to the overlaying goal of investigating the suitability of synthetic images for training deep learning models in aviation as well as influences on the sim-to-real generalization capability of the deep learning models.

As discussed in Section 1.2, numerous factors potentially influence the generalization capability from synthetic to real-world images of deep learning models. Given the multitude of potential influences, this study focuses on a subset of key factors.

> **Research Question 1: How well do selected deep learning models trained on synthetic images perform in terms of accuracy-related metrics on real-world images in specific aviation use-cases compared to a baseline model trained on real-world images?**

The first research question investigates the suitability of synthetic images for training deep learning models in aviation. This question is divided into two sub-questions, each addressing a distinct aviation use-case with specific characteristics that has received limited attention in the literature. By investigating these under-explored use-cases, this work contributes new insights into the applicability of synthetic images for model training.

> **Research Sub-Question 1.1: How well does a deep learning drogue detection model trained on synthetic images perform in terms of mAP on real-world images compared to a baseline model trained on real-world images?**

The first research sub-question focuses on the use-case of drogue detection during air-to-air refueling, a scenario characterized by high altitude and limited

Figure 1.2: **Exemplary comparison of an image from the autonomous driving domain (left, [52]) and an image of air-to-air refueling (right, [84])**

visual complexity. Unlike many of the use-cases explored in the literature, this scenario presents an interesting combination of conditions that may result in a better generalization from synthetic to real-world images. Specifically, the high altitude and the limited variability of participating objects reduce the occurrence of shadows, reflections, distractions, and occlusions, resulting in visually simpler images. Fig. 1.2 shows an exemplary comparison of an image from the autonomous driving domain and an image of air-to-air refueling, highlighting the differences in visual complexity. Furthermore, the limited number of objects in sight, typically one or two drogues and one tanker aircraft, as well as the absence of complex ground textures, such as grass, leaves, and tarmac, contribute to a more controlled and reproducible environment.

This seemingly simpler setting makes air-to-air refueling a suitable starting point for investigating the generalizability from synthetic to real-world images and it allows for better isolation of influential factors. The situation may even be simple enough for neural networks to generalize well from synthetic to real-world images. Although not yet considered in the literature, it is a natural scientific practice to resort to a simpler and more controlled setting in which as many uncontrollable factors as possible can be eliminated. In addition to these benefits, the use-case is also chosen because it is a representative example of a simple perception task in aviation. By exploring this use-case, this work aims to determine whether it is possible to train a deep learning object detection model on synthetic images that can perform effectively on real-world images.

To investigate this question, a synthetic image generation toolchain is developed using a publicly available game engine. It is utilized to generate and extract synthetic images of air-to-air refueling. The images are used to train a deep learning object detection model which is then evaluated on real-world images.

Research Sub-Question 1.2: How well does a deep learning semantic segmentation model trained on stylistically replicated synthetic images from low-altitude UAS perspective perform in terms of mIoU on real-world images compared to a baseline model trained on real-world images?

The second research sub-question focuses on the use-case of semantic segmentation from low-altitude UAS perspective. This scenario is inherently more complex due to the presence of multiple objects, varying object sizes, and diverse perspectives. However, it is hypothesized that the claimed influence of differences in texture on the sim-to-real generalization (cf. Section 1.2), may have a reduced impact because of the flight altitude and the resulting lower resolution of the objects and the background.

While [33] shows a large performance gap for a semantic segmentation model trained on their synthetic dataset from UAS perspective, the synthetic and the real-world datasets exhibited considerable visual differences. This may have influenced and skewed the results. Therefore, in this work, a real-world dataset is stylistically replicated in a simulation environment and synthetic images are generated and used for training. In this context, stylistically replicated describes that the synthetic dataset shows a visually similar scenery but does not clone the positions and appearance of every object, as this would not be feasible. This stylistic replication allows to explore the sim-to-real generalization in a more controlled setting, where the synthetic data is more closely aligned to the real-world data. Similar to Research Sub-Question 1.1, the final evaluation of the deep learning models is performed on the real-world counterparts.

Research Question 2: How much do selected deep learning model architectures differ in accuracy-related metrics on real-world images when trained on synthetic images?

The second research question explores the impact of the deep learning model architecture on the sim-to-real generalization capability of models trained on synthetic images.

The influence of the deep learning model architecture on the generalization capability from synthetic to real-world images has received limited attention in the literature. However, it may have a significant influence on the generalization capability of the final model. First, the architecture itself is an integral part of a deep learning system as described in Section 1.2. Second, it has been shown that deep learning models have different strengths and weaknesses. For example, some models are better at generalizing to art-work then others (cf. Section 1.2). Third, recent advancements in model architectures have led to an improvement in overall accuracy. However, it remains unclear whether models also differentiate with respect to their generalization capabilities from synthetic to real-world images and whether overall accuracy improvements translate to enhanced generalization.

Against this background, this research provides a novel perspective by investigating the generalization capability from synthetic to real-world images of different deep learning models. It examines whether the generalization of these models aligns with their overall performance or whether certain architectures or backbones are better at generalizing from synthetic images to real-world ones.

To ensure the generalizability of the findings, this research considers multiple deep learning architectures, backbones, datasets, and perception tasks. On the one hand, it considers drogue detection during air-to-air refueling from Research Sub-Question 1.1. On the other hand, it considers two semantic segmentation use-cases. First, the semantic segmentation use-case from a low-altitude UAS from Research Sub-Question 1.2. Second, a semantic segmentation use-case from the autonomous driving domain. As deep learning models, 27 semantic

segmentation models on each of the two datasets as well as 12 object detection models on the corresponding dataset are considered. Because of the promising characteristics of transformer backbones that may lead to a good generalization capability as explained in Section 1.2, the investigation also includes various transformer backbones.

By exploring the relationship between model architecture and generalization capability across these diverse scenarios, this study aims to provide novel insights into the factors that influence the sim-to-real generalization of deep learning models.

Research Question 3: What influence do selected image augmentation techniques during training of deep learning models on synthetic images have on accuracy-related metrics on real-world images?

The third research question explores the potential of image augmentation techniques to enhance the generalization capability of deep learning models from synthetic to real-world images.

While basic image augmentations, such as adding blur or adjusting contrast, have shown to improve performance on real-world datasets [85, 86] and robustness to real-world distribution shifts [87], their effectiveness in reducing the impact of the shift from synthetic to real-world images remains unclear. Although initial studies have demonstrated that image augmentations modeling different sensor effects can improve performance on real-world images (cf. Section 1.2), a comprehensive investigation of the impact of various augmentations is lacking.

Against this background, this study systematically investigates the influence of various pixel-level augmentations on the sim-to-real generalization capability of deep learning models. By examining which image augmentations are most beneficial, this research also aims to provide insights into the shortcomings of synthetic images and the differences to real-world images that are relevant for deep learning models. Using image augmentations has the advantage that

current state-of-the-art visual simulation environments can be used and that only image post-processing techniques have to be applied during the training of the model. However, such insights might also benefit the future development of improved visual simulation environments.

Specifically, this work considers 25 basic pixel-level image augmentations, such as adding blur or noise, adjusting the contrast of images, and changing image colors. Again, to ensure the generalizability of the results, this work considers multiple deep learning models, datasets and perception tasks.

Compared to previous works, this study investigates a broader range of sensor effects and augmentations, as well as more deep learning models and datasets covering various domains and perception tasks. Furthermore, it presents a systematic evaluation of the effect of each augmentation in isolation and of some combinations, providing a detailed assessment of which augmentations improve real-world performance and which do not. This broad investigation aims to increase the generalizability of the results to other domains and to provide actionable insights for practical applications.

1.4 Structure of this Work

In the following Chapter 2, the basics of this thesis are presented. First, the evaluation metrics for assessing the accuracy of object detection and semantic segmentation tasks are explained in Section 2.1. Subsequently, methods for quantifying the sim-to-real generalization of deep learning models are introduced in Section 2.2, including the proxy metric sim-to-real gap, which is useful for the investigation of architecture-based influences on the generalization capability. Section 2.3 provides a concise overview of the deep learning models examined in this work. The chapter concludes in Section 2.4 with a brief presentation of the image augmentation techniques evaluated in the context of Research Question 3.

Chapter 3 presents the image datasets corresponding to the three use-cases investigated in this research, which represent diverse applications with unique characteristics and challenges. For each use-case, both real-world and synthetic datasets are introduced. A particular focus is laid on the two synthetic datasets created within the scope of this work. First, the datasets for investigating the use-case of drogue detection during air-to-air refueling is presented in Section 3.1. Since the synthetic dataset is created within the scope of this work, the section also contains some background on the air-to-air refueling procedures which were used to derive parameters for data generation to ensure an unbiased synthetic dataset. A first presentation of the created dataset was published in [88]. Second, the datasets for semantic segmentation from a low-altitude UAS perspective is introduced in Section 3.2, which contains complex aerial imagery. The synthetic dataset is again created within the scope of this work. A first presentation of it was published in [89]. Third, the datasets for the semantic segmentation use-case in the autonomous driving domain is presented in Section 3.3, which contains different perspectives and diverse lighting conditions.

The investigation of the main Research Question 1 is conducted last to allow the influences explored in Research Question 2 and Research Question 3 to be considered in the final evaluation. Therefore, Chapter 4 explores Research Question

2 on the influence of the deep learning model architecture on its sim-to-real generalization. Following a presentation of the methodology and experimental setup in Sections 4.1 and 4.2, the results are presented in Section 4.3. They are divided into general influences as well as specific influences of the backbone, the number of trainable weights, the optimizer, and the architecture. A summary of the findings is provided in Section 4.4. A first presentation of the results for the object detection models was published in [90] and for the semantic segmentation models in [91].

Chapter 5 investigates Research Question 3 on the influence of image augmentation techniques during training on the sim-to-real generalization of deep learning models. After presenting the methodology and experimental setup in Sections 5.1 and 5.2, the results are presented in Section 5.3. First, the influence of image augmentation techniques is explored in isolation. Afterwards, selected combinations are evaluated. A summary of the results is given in Section 5.4. A first presentation of the results was published in [92].

Building on the findings from the previous chapters, Chapter 6 explores the main Research Question 1 on how well deep learning models trained on synthetic images perform compared to models trained on real-world images in selected aviation use-cases. Specifically, Research Sub-Question 1.1 on the use-case of drogue detection is investigated in Section 6.1, and Research Sub-Question 1.2 on the use-case of semantic segmentation from a low-altitude UAS perspective is investigated in Section 6.2.

The thesis closes with Chapter 7 giving a conclusion of the main findings on the research questions and a presentation of potential future work.

2 Basics

2.1 Evaluation Metrics

On a general level, deep learning models for environment perception can be assessed using two key factors: accuracy and speed. In practical applications, these factors often require a trade-off. For instance, warehouse systems with real-time requirements may prioritize speed to maintain high throughput and revenue. In such cases, some errors may be acceptable. On the other hand, some applications in medical image analysis may allow for longer processing times but require higher accuracy and fewer errors.

Speed is influenced by multiple factors, including the model itself, hardware, and implementation. The same model can perform faster on more powerful hardware or with a more efficient implementation. However, as this work considers sim-to-real generalization, speed is not the primary focus. Instead, it concentrates on evaluating accuracy. There are various metrics that are commonly used to assess it. The following section will introduce the accuracy metrics employed in this work.

2.1.1 Object Detection

To evaluate the accuracy of object detection models, this work utilizes the mean Average Precision (mAP) metric. Its definition is derived in the following section.

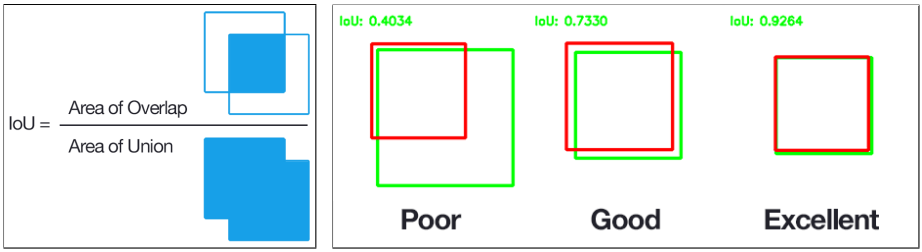
First, it requires the definition of Intersection over Union (IoU), precision, and recall.

Let $C = \{c_1, \ldots, c_K\}$ be the K classes that the object detection model is trained on and let $G = \{g_1, \ldots, g_M\}$ be the M ground truth objects in the dataset. Each ground truth object g_m can be described as (c_{g_m}, b_{g_m}), where $c_{g_m} \in C$ represents the class and $b_{g_m} \in \mathbb{R}^4$ represents the bounding box of the object. The subscript g_m is used to symbolize the corresponding ground truth object. A bounding box b is a vector with four components that can be represented in various formats, such as those used in Microsoft COCO [93], YOLO [68, 94, 95], Pascal VOC [96–98], and others [99]. Since these formats can be converted into each other, the exact format can be neglected here. For each format, there is a function *area* that calculates the area of a bounding box. Let further $D = \{d_1, \ldots, d_N\}$ be the N detections of the object detection model on the dataset. Each detection d_n can be described as $(c_{d_n}, b_{d_n}, \alpha_{d_n})$, where $c_{d_n} \in C$ represents the predicted class, $b_{d_n} \in \mathbb{R}^4$ represents the predicted bounding box, and $\alpha_{d_n} \in [0, 1]$ represents the confidence of the model in the detection. The subscript d_n is used to symbolize the corresponding detection. Typically, only detections with a confidence greater than or equal to a confidence threshold $\gamma \in [0, 1]$ are considered.

The primary task of an object detection model is to place an accurate bounding box around all objects of interest, i.e., around all ground truth bounding boxes, and assign them to their correct classes [100]. An accurately predicted bounding box is one that has a high overlap with a ground truth bounding box of the same class and has a similar size, i.e., it is not too large, nor too small. To quantify the accuracy of the predicted bounding box, the IoU is used. The IoU, also known as the Jaccard index or Jaccard similarity coefficient [100], is calculated for two bounding boxes b_m and b_n as

$$\mathrm{IoU}(b_m, b_n) := \frac{\mathrm{area}(b_m \cap b_n)}{\mathrm{area}(b_m \cup b_n)}, \tag{2.1}$$

where $\mathrm{area}(b_m \cap b_n)$ represents the area of the intersection and $\mathrm{area}(b_m \cup b_n)$ the area of the union of the bounding boxes. Fig. 2.1a illustrates the calculation visually. From the definition, it follows that $\mathrm{IoU}(b_m, b_n) \in [0, 1]$, where $\mathrm{IoU}(b_m, b_n) = 1$ if both bounding boxes overlap exactly, and $\mathrm{IoU}(b_m, b_n) = 0$ if



(a) **Visualization of IoU calculation. Source: [101]** (b) **Visualization of different IoU values for two overlapping bounding boxes. Source: [102]**

Figure 2.1: **Visualization of Intersection over Union (IoU)**

the boxes do not overlap. The closer the IoU is to 1, the better the localization of the detection. For illustration, Fig. 2.1b shows several bounding box pairs along with their corresponding IoU values.

To specify the minimum localization accuracy required for a predicted bounding box to be considered a correct detection, an IoU threshold $\tau \in [0, 1]$ is defined. A detection $d_n \in D$ is then considered to be a correct detection for a ground truth object $g_m \in G$ if and only if the following conditions are met:

$$c_{d_n} = c_{g_m} \quad \text{(Correct class)} \tag{2.2}$$

$$\text{IoU}(b_{d_n}, b_{g_m}) \geq \tau \quad \text{(Accurate bounding box)} \tag{2.3}$$

If a detection meets the conditions for correct detection for multiple ground truth objects, it is assigned to the object with the highest IoU. In case of a tie, the detection can be assigned arbitrarily.

Using the IoU threshold τ, the set of detections D can be partitioned into subsets TP containing all corrected detections (true positives) and FP containing all incorrect detections (false positives). When multiple bounding boxes are predicted for the same ground truth object, it is common to only consider the detection with the highest confidence as a true positive and to classify the others as false positives [97]. It can be assumed without loss of generality that the detections are sorted in descending order of confidence, i.e., $\alpha_{d_i} \geq \alpha_{d_j}$ for

$1 \leq i < j \leq N$. Therefore, to classify a detection $d_n \in D$ as a correct detection (true positive), it must further meet the following condition:

$$\forall d_j \in D \setminus \{d_n\} : d_j \text{ meets (2.2) and (2.3)} \Rightarrow j > n \quad \text{(Highest confidence)} \tag{2.4}$$

Overall, a detection can be classified as a false positive due to a detection of a non-existent ground truth object, an incorrectly placed bounding box (IoU less than τ), an incorrect class prediction, or a duplicate detection of an already detected object. Summarized, the sets *TP* and *FP* are defined as

$$TP := \{d_n \in D \mid \exists g_m \in G : \text{conditions (2.2), (2.3), and (2.4) are met}\}, \tag{2.5}$$

$$FP := \{d_n \in D \mid d_n \notin TP\}. \tag{2.6}$$

In addition to *TP* and *FP*, the set *FN* contains all missed detections (false negatives) and is defined as

$$FN := \{g_m \in G \mid \nexists d_n \in D : \text{conditions (2.2), (2.3), and (2.4) are met}\}. \tag{2.7}$$

It should be noted that, there are different opinions in the literature on whether the IoU in condition (2.3) should be *greater than* [96–98] or *greater than or equal to* [93, 99] the IoU threshold τ, for a detection to be considered a true positive. In this work, a detection is classified as a true positive if the IoU in condition (2.3) is greater than or equal to the IoU threshold τ, following the widely adopted evaluation protocol of Microsoft COCO [93].

Precision and Recall For a given IoU threshold τ, the precision P and recall R of an object detection model are defined as

$$P := \frac{|TP|}{|TP| + |FP|} = \frac{|TP|}{|D|}, \tag{2.8}$$

$$R := \frac{|TP|}{|TP| + |FN|} = \frac{|TP|}{|G|}, \tag{2.9}$$

where $|.|$ denotes the cardinality of a set [100]. Note that both $P = P(\tau, \gamma)$ and $R = R(\tau, \gamma)$ depend on the IoU threshold τ and the confidence threshold γ.

By definition, $P \in [0, 1]$ and $R \in [0, 1]$, with $P = 1$ indicating that all detections are correct and $R = 1$ indicating that all ground truth objects are detected. An optimal detector would achieve both $P = 1$ and $R = 1$, but in practice, this is rarely the case. Instead, a trade-off between precision and recall has to be made, which is influenced by the confidence threshold γ. Specifically, decreasing γ can lead to a higher recall by considering more detections, but often at the cost of precision. Conversely, increasing γ can increase precision by only considering detections with a high confidence, but often at the cost of recall [99].

The importance of precision compared to recall depends on the application. Precision is crucial when false positives have severe consequences. For instance, in military aviation applications, such as target identification, a false positive could have devastating consequences, and precision is critical to avoid wrongful engagements. On the other hand, recall is crucial when false negatives have serious consequences. For example, in search and rescue operations, a false negative could mean loss of life, and prioritizing recall ensures all potential victims are identified.

TP, FP, and consequently P and R can also be calculated class-wise by considering only detections $d_n \in D$ with $c_{d_n} = c$ and ground truths $g_m \in G$ with $c_{g_m} = c$ for a class $c \in C$. These are denoted as TP_c, FP_c, P_c, and R_c.

Average Precision Another common metric to evaluate and compare object detection models is Average Precision (AP). For a given object detection model and a given IoU threshold τ, precision P_c and recall R_c can be calculated for each class $c \in C$ at various confidence threshold values γ. This allows to draw a Precision-Recall Curve. The AP, dependent on class $c \in C$ and IoU threshold τ, is then calculated as the area under the Precision-Recall Curve as

$$\mathrm{AP}_c^\tau = \int_0^1 P_c(r)dr. \tag{2.10}$$

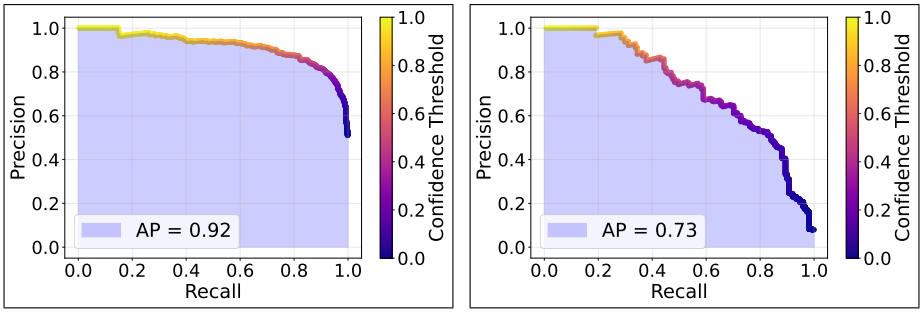


Figure 2.2: **Example Precision-Recall Curves with Average Precision**

A high AP indicates that when the confidence threshold γ decreases, precision remains high while recall increases, meaning that both precision P and recall R achieve high values simultaneously. A low AP indicates that precision declines rapidly when the confidence threshold γ decreases. Example Precision-Recall Curves are shown in Fig. 2.2.

Note that $TP(\gamma)$ and $FP(\gamma)$ can be viewed as decreasing functions of γ, since an increasing confidence threshold γ leads to fewer detections. As a result, $R = R(\gamma)$ is a monotonically decreasing function. On the other hand, no statement can be made about $P = P(\gamma)$, as both the numerator and denominator are decreasing functions. This typically results in a zig-zag pattern of the Precision-Recall Curve in real-world applications. To address this, the curve is often pre-processed for practical calculations [99].

Mean Average Precision Building on the AP, the mean Average Precision (mAP) is calculated by taking its average over all classes $c \in C$ as

$$\text{mAP}^{\tau} = \frac{1}{|C|} \sum_{c \in C} \text{AP}_c^{\tau}. \tag{2.11}$$

The metric is again dependent on the chosen IoU threshold τ. Commonly used IoU thresholds for calculating mAP^{τ} are $\tau = 0.5$ and $\tau = 0.75$, resulting in $\text{mAP}^{0.5}$ and $\text{mAP}^{0.75}$, respectively [93, 96].

Another widely used but more challenging metric is $\text{mAP}^{[.5:.05:.95]}$, which is calculated as the average of the mAP^{τ} over multiple IoU thresholds $\tau \in T := \{0.5, 0.55, \ldots, 0.9, 0.95\}$ as

$$\text{mAP}^{[.5:.05:.95]} = \frac{1}{|T|} \sum_{\tau \in T} \text{mAP}^{\tau}. \tag{2.12}$$

This metric rewards detectors with better localization capabilities and originates from the COCO evaluation [93]. Note that in the COCO evaluation, the metric $\text{mAP}^{[.5:.05:.95]}$ is denoted as AP. However, this work strictly adheres to the definitions outlined above. As this work mainly uses $\text{mAP}^{[.5:.05:.95]}$ for the evaluation of object detection models, in the following it is abbreviated as mAP unless otherwise stated. Furthermore, in accordance with common literature, mAP is expressed as a percentage between 0% and 100% instead of between 0 and 1.

2.1.2 Semantic Segmentation

To evaluate the accuracy of semantic segmentation models, this work utilizes the mean Intersection over Union (mIoU) metric. Its definition is derived in the following section. First, it requires the definition of the pixel-wise IoU of two semantic segmentation annotation masks. It is then generalized for a whole dataset.

The goal of semantic segmentation is to assign each pixel of an image to a specific class. Let $I \subset \mathbb{R}^{h \times w \times 3}$ be a dataset containing M RGB images of size $h \times w$ and $C = \{c_1, \ldots, c_K\}$ be the K classes that the semantic segmentation model is trained on. For each image $i \in I$, the ground truth annotation mask is a 2D-array $g^{(i)} \in C^{h \times w}$ where each entry contains the class of the corresponding pixel. Similarly, a predicted annotation mask is a 2D-array $p^{(i)} \in C^{h \times w}$ where each entry contains the class that the model predicted for the corresponding pixel.

Pixel-wise Intersection Over Union The per-image pixel-wise IoU of a predicted and a ground truth annotation mask for an image i measures the overlap between the annotations masks for a class $c \in C$. It is defined as

$$\widehat{\text{IoU}}_c(p^{(i)}, g^{(i)}) = \frac{TP_c^{(i)}}{TP_c^{(i)} + FP_c^{(i)} + FN_c^{(i)}}, \tag{2.13}$$

where $TP_c^{(i)}$ is the number of true positive, $FP_c^{(i)}$ of false positive, and $FN_c^{(i)}$ of false negative pixels for class c. Specifically, true positives are pixels for which the model correctly predicts class c. False positives are pixels for which the model falsely predicts class c. False negatives are pixels for which the model misses a detection of class c. The per-image pixel-wise IoU ranges between zero and one, with higher values indicating better semantic segmentation accuracy.

To evaluate the accuracy of a semantic segmentation model for a class $c \in C$ on the whole dataset, the pixel-wise IoU is calculated for the whole dataset as

$$\text{IoU}_c(I) = \frac{\sum_{i \in I} TP_c^{(i)}}{\sum_{i \in I} TP_c^{(i)} + \sum_{i \in I} FP_c^{(i)} + \sum_{i \in I} FN_c^{(i)}}. \tag{2.14}$$

For each class, it measures the overlap between the ground truth and the predicted annotation masks.

Mean Intersection over Union To comprehend the semantic segmentation accuracy of a model on a dataset into a single number, the mIoU is used. It averages the pixel-wise IoU_c of the dataset over all classes, providing an overall assessment of the accuracy of the model. For a dataset I, it is calculated as

$$\text{mIoU}(I) = \frac{1}{|C|} \sum_{c \in C} \text{IoU}_c(I). \tag{2.15}$$

Summarized, it measures the average overlap between the predicted and the ground truth annotation masks over all classes. A higher mIoU value indicates better semantic segmentation performance, with a maximum value of 1 representing perfect segmentation for all classes.

2.2 Quantification of Sim-to-Real Generalization

This section outlines the evaluation metrics used to assess the sim-to-real generalization capabilities of deep learning models in this work.

The primary metric to measure the generalization capability is the absolute performance of the model on real-world images when trained on synthetic images. This measure is crucial for practical applications as it directly quantifies the final performance on the real-world dataset. To do so, established metrics exist. For object detection, the mAP as presented in Section 2.1.1 is used and for semantic segmentation, the mIoU as presented in Section 2.1.2.

However, for a model with a poor performance on real-world images, this metric does not take into account whether the model did not learn any useful features at all, i.e., also has poor performance on synthetic images, or whether the model is just unable to generalize to real-world images. To evaluate this, a second metric to quantify the generalization to real-world images relative to the performance on synthetic images is used. This metric, called sim-to-real gap, is a useful proxy for investigating different models and model-based influences on the generalization capability.

The sim-to-real gap is defined as follows. Let R be a real-world dataset and S be a synthetic dataset. Both may further be split into disjoint training ($R_{\text{train}} \subset R$, $S_{\text{train}} \subset S$) and testing ($R_{\text{test}} \subset R$, $S_{\text{test}} \subset S$) datasets. Let m_S denote a model trained on S_{train} and $e(m, X) \rightarrow [0, 100]$ be an evaluation function for a model m on a dataset X. The sim-to-real gap g is then defined as

$$g := e(m_S, S_{\text{test}}) - e(m_S, R_{\text{test}}). \tag{2.16}$$

It measures the absolute performance difference of the model between the synthetic and real-world test datasets in percent points (pp). Using this evaluation metric, a comprehensive understanding of the performance gap of deep learning models can be gained to identify potential areas for improvement.

In the literature, the notion of sim-to-real gap is sometimes used with a slightly different meaning [15, 21, 28, 31]. Instead of measuring the performance of the same model on the synthetic compared to the real-world test dataset, it measures how well a model trained on synthetic images performs on real-world images compared to a model trained directly on real-world images. This can be formalized as

$$\tilde{g} := e(m_R, R_{\text{test}}) - e(m_S, R_{\text{test}}) \tag{2.17}$$

for a model m_R trained on R_{train}.

This definition implicitly considers the complexity of the real-world images by incorporating the performance of a model trained directly on them. Therefore, when a model trained on real-world data also has poor performance on the real-world images, the calculated gap might be smaller. However, this definition has several drawbacks. First, as it requires to also train a model on real-world images, more real-world images are needed. However, these are usually not available in the given problem setting. Second, even when some real-world images are available for training, R_{train} is usually much smaller because of the general lack of real-world images leading to potential bias or overfitting of m_R. Alternatively, only a fraction of the available synthetic images could be used which might also skew the results. Third, the model might have different optimal training hyperparameters on the synthetic and the real-world datasets. Therefore, differing optimal hyperparameters might be another factor influencing the results. Additionally, this makes it more difficult to isolate the influence of training hyperparameters on the sim-to-real gap. Last, the need to train and evaluate two deep learning models increases the computational cost and time required to calculate the sim-to-real gap significantly.

On the other hand, Eq. (2.16) does not require any model training on real-world images. Furthermore, the sim-to-real gap calculation is based on a single trained model, allowing for a more straightforward and isolated investigation of model-based characteristics and influences. This also removes the influence of differing hyperparameters and allows to investigate their influence on the gap. The downside that the complexity of the real-world images is not taken into account has only little relevance for this work, as the characteristics of interest of the different models should be observable when comparing the

Table 2.1: **Advantages and disadvantages of both sim-to-real gap definitions**

	Advantages	**Disadvantages**
g	• No real-world training data needed. • Sim-to-real gap calculation is based on the same trained model. • Allows to investigate influence of hyperparameters.	• Complexity of the real-world images is not taken into account. • Model with low performance on both datasets has a small gap.
$\tilde{g}$	• Complexity of the real-world data is taken into account.	• Differing training dataset sizes of R_{train} and S_{train} might influence results. • More real-world data needed. • Potentially influenced by suboptimal hyperparameters on one of the datasets.

trend of multiple models on the same dataset. Although it is unclear how well a model trained on real-world images might perform, differences in performance between the synthetic and real-world test datasets are quantified using this formulation. A second potential shortcoming, that the metric might favor models with a low performance on both the synthetic and the real-world test datasets, can be easily mitigated when considered in combination with the absolute performance of the model, as is done in this work. All advantages and disadvantages are summarized in Table 2.1.

2.3 Deep Learning Models

In this work, differences in sim-to-real generalization of various deep learning models are investigated. This chapter presents a comprehensive overview of the examined deep learning models.

The selection of the models is based on multiple criteria. Most importantly, the models should be influential and widely recognized, as shown by a high number of citations. Specifically, all models evaluated have more than 1,000 citations according to Google Scholar[1] as of May 12, 2025, with many of them having many more citations of up to 267,300. Furthermore, source code should be available to reduce potential errors and differences in implementation, and corresponding pre-trained weights should be published to reduce computational load during training.

To ensure consistency in training and evaluation, this study utilizes two widely-used libraries. For object detection, Torchvision [103] is used and for semantic segmentation, MMSegmentation [104] is used. These libraries provide standardized implementations of the investigated models, minimizing potential errors and differences in implementation.

Because of the selection criteria, and because of the duration of a PhD, this work might not include all the most recent state-of-the-art models. However, this is justified since the primary objective of this research is to examine whether there are differences in sim-to-real generalization across various models and to explore general trends, rather than to identify the best possible model for a given use-case.

On a general level, the considered deep learning models consist of two primary components. First, a backbone, which extracts high-dimensional feature representations from raw input images. This backbone is typically a CNN such as a ResNet [105] or a transformer-based model such as a Swin transformer [106]. It serves as a feature extractor, capturing spatial and semantic features for down-

[1] https://scholar.google.com

stream tasks. Second, the meta-architecture, which defines how the extracted features are processed for a specific task, such as object detection or semantic segmentation. The meta-architectures include, e.g., Faster R-CNN [107] for object detection and UPerNet [108] for semantic segmentation. They define how the feature representations of the backbone are refined, aggregated, and used to generate final predictions.

The backbones examined in this study include Visual Geometry Group models (VGG) [109], Residual Neural Networks (ResNets) [105], Wide ResNets [110], ResNeXt [111], Split-Attention Networks (ResNeSt) [112], ConvNeXt [113], MobileNetV2 [114], MobileNetV3 [115], Data-efficient Image Transformers (DeiT) [116], Shifted Windows Transformer (Swin) [106], Twins-PCPVT [117], and Twins-SVT [117].

The object detection architectures investigated include Faster Region Convolutional Neural Networks (Faster R-CNN) [107], Single Shot MultiBox Detectors (SSD) [118], RetinaNets [119], and Fully Convolutional One-Stage Object Detectors (FCOS) [120].

The semantic segmentation architectures examined include Unified Perceptual Parsing Networks (UPerNet) [108], Pyramid Scene Parsing Networks (PSPNet) [121], Fully Convolutional Networks (FCN) [122], DeepLabV3 [123], and DeepLabV3+ [124].

A comprehensive list of all considered model combinations is given in Chapter 4. The following sections provide brief descriptions of the used backbones and architectures.

2.3.1 Backbones

VGG [109] is a family of CNNs introduced in 2014. The networks follow a simple basic architecture consisting of convolutional layers with, by the time, small filters of size 3×3. This allowed to increase the depth of the model compared to previous state-of-the-art models. Spatial pooling is done by five

max-pooling layers and the hidden layers further use the ReLU [125] function. The final classification is performed by three fully-connected layers followed by a soft-max layer. All presented models follow this design pattern and only differ in their depth, i.e., the number of weighted layers. The introduced models include VGG-13, -16, and -19, consisting of 10, 13, and 16 convolutional layers respectively, in addition to the three fully-connected layers.

ResNet [105] is a family of CNNs that were developed based on the observation that the accuracy of networks degrades with increasing depth once a certain threshold is passed. This observation was counter-intuitive, as deeper networks are a superset of shallower ones and therefore should have at least the same accuracy. It was shown that this degradation is not because of overfitting but rather that the training error increased as well. To mitigate this problem and to ease the training of deeper neural networks, the residual learning framework was presented. In this framework, the residual building block is introduced consisting of multiple convolutional layers and a shortcut connection which performs the identity mapping over multiple layers. The shortcut connection is applied to skip the convolutional layers in the block by adding the input of the block to its output. ResNets consist of multiple of these blocks.

The authors introduced multiple variants of these networks including ResNet-34, -50, -101, and -152. The appended number denotes the number of trainable layers including one fully-connected layer responsible for final classification. The architectural design of ResNet-34 is relatively simple consisting mainly of 3×3-layers similar to VGG models. The skip connections are used over every two convolutional layers that build a residual block. Networks deeper or equal to ResNet-50 employ a bottleneck design in the residual blocks reducing the depth dimension of the feature maps internally for computational reasons. In these cases, the residual blocks contain three convolutional layers with filter sizes of 1×1, 3×3, and 1×1. In this form, the 1×1-filters are used to decrease and consecutively increase the depth dimension of the feature maps. It is important to note that ResNets have a lower complexity in terms of needed floating-point operations compared to VGGs although they are much deeper [105].

Wide ResNet [110] is the result of an architecture study on ResNets. The study was based on the observation that the accuracy of ResNets improves

with depth but that the number of needed layers increases exponentially for each fraction of improvement. The inventors of Wide ResNets argue that the shortcut connections of ResNets are the enabler for deep ResNets but also lead to problems in the training process. Because of the shortcut connection, the gradient is not forced to go through the residual block and therefore, a block might not learn anything during training. As a result, there may be only a few blocks that learn useful representations and many blocks that provide just very little information to the overall goal.

Against this background, the authors conducted a detailed experimental study on the architecture of ResNet blocks and presented an adapted architecture. The key innovation was to increase the width of the network instead of only increasing the depth dimension. In this context, width describes the depth of the feature maps. Furthermore, Wide ResNets do not use the bottleneck architecture of deeper ResNets as it was used to decrease width but employ a changed order of batch normalization, activation, and convolution in the residual blocks as presented in [126]. Overall, Wide ResNets employ a simple architecture of repeated residual blocks consisting only of 3×3-filters while keeping the depth of the feature maps constant inside a block.

Wide ResNets can be denoted by WRN-n-k where n describes the number of convolutional layers and k is a factor introduced to denote how wide a network is compared to the original ResNet. As an example, WRN-50-2 describes a Wide ResNet with 50 convolutional layers where the depth of the feature maps is twice as large as in the original ResNet architecture.

ResNeXt [111] also builds on ResNet but splits the input of each residual block into a specified number of branches, also called groups. This number is called cardinality C and can be seen as a new dimension of the architecture in addition to width and depth. In each of the C branches, the input is projected into a lower-dimensional embedding with width w, i.e., number of channels, using 1×1-filters. Afterwards, each branch performs transformations using 3×3-filters followed by a transformation back to the original number of channels using 1×1-filters. The feature maps of each branch are then aggregated by element-wise addition. Furthermore, each block has a shortcut connection as presented in ResNets. Similar to VGGs, ResNets and Wide ResNets, the overall

ResNeXt model consists of a stack of multiple of these blocks.

A ResNeXt model can be denoted by ResNeXt-N (Cxwd) where N is the number of trainable layers, C is the cardinality and w is the width of each branch as described above. Exemplary, a ResNeXt-50 (32x4d) simply replaces all ResNet blocks with ResNeXt blocks. Each of these ResNeXt blocks has 32 branches where the 3×3-convolutions are applied on an embedding with 4 channels. A ResNeXt-50 (32x4d) has a comparable number of parameters and floating-point operations as a regular ResNet-50 [111].

ResNeSt [112] further builds on ResNet and ResNeXt. It replaces the traditional residual blocks of ResNet with a novel Split-Attention module, which serves as a drop-in replacement for the original residual block. At its core, ResNeSt combines the multi-branch network layout guided by the cardinality parameter from ResNeXt with a channel-wise attention mechanism inspired by the SK-Net block [127]. To do so, each branch of the ResNeXt block is further split, guided by the newly introduced numerical hyperparameter radix that denotes the number of splits in each branch. After performing a series of transformations on the splits individually, the splits are fused again with learnable weight parameters resulting in channel-wise attention. This enables the model to select more informative features while suppressing less relevant ones. Afterwards, the branches are aggregated again, similar to the ResNeXt block.

MobileNetV2 [114] is built for use in mobile applications and extends the previous version MobileNetV1 [128]. MobileNetV1 uses depthwise separable convolutions to replace traditional convolutional layers to reduce computations. Depthwise separable convolutions are defined by two separate layers: A depthwise convolution for spatial filtering, followed by a 1×1 point-wise convolution. MobileNetV2 then introduces an inverted residual structure and linear bottlenecks. In its building block, the depth-dimension of the input is first expanded using 1×1-filters followed by depthwise separable convolutions whose 1×1-filters project the depth-dimension back to a smaller size. This is an *inverted* residual structure as previous residual blocks reduced the depth dimension internally, instead of increasing it.

MobileNetV3 [115] further builds on MobileNetV2 [114] as well as Mnas-

Net [129]. MnasNet is an extension of MobileNetV2 that introduces lightweight attention modules based on the squeeze-and-excitation structure [130] into the building block. MobileNetV3 then builds on it by using MnasNet as the initial model and applying network architecture search methods to find an improved configuration. The authors combine hardware-aware network architecture search (NAS) similar to [129] for the search of global structures and NetAdapt [131] for fine-tuning of individual layers. Additionally, they introduce some manual architecture changes which include the redesign of computationally expensive layers and the introduction of a non-linearity that is faster to compute. MobileNetV3 is published in two variants, MobileNetV3-Small and -Large, designed for mobile use-cases with low and high computational resources, respectively.

ViT [132] applies the transformer architecture, originally designed for natural language processing, to the visual domain. Instead of using convolutional layers, ViT processes images by dividing them into fixed-size patches, which are then linearly embedded into a sequence of tokens. These tokens are fed into a transformer encoder, enabling the model to capture long-range dependencies and global context using attention mechanisms. The ViT architecture also introduces key modifications to adapt transformers for vision tasks. These include the addition of positional embeddings to retain spatial information to compensate for the transformers lack of an inherent inductive bias that is naturally found in CNNs. ViTs come in different variants, such as ViT-S, ViT-B, and ViT-L, corresponding to increasing model size and complexity.

DeiT [116] uses the same transformer-based architecture as ViT but tries to mitigate its disadvantage of requiring huge amounts of training data to achieve good performance. To do so, DeiT introduces some novel training strategies and optimizations. When pre-trained weights for a DeiT model are used in this work, they are based on these optimizations. However, to not let the training procedure influence the model architecture comparison results, the newly introduced training procedures are not used during the training on synthetic images performed in this research but all models are trained using the same ones. As the model architecture is the same, DeiT also comes in different variants, such as DeiT-S, DeiT-B, and DeiT-L. For their use in downstream tasks like object detection or semantic segmentation, DeiT is often expanded with

a Multi Level Neck (MLN) that improves multi-scale representation learning by aggregating information from multiple layers instead of using only the final transformer output.

Swin [106] builds upon ViT by addressing key limitations related to multi-scale representation and computational efficiency. While ViT produces feature maps at a single scale, coming from its origins in natural language processing where words are the basic elements, images inherently contain objects of varying scales. To account for this, Swin introduces a hierarchical architecture, starting with small image patches that are progressively merged in deeper layers, allowing the model to build multi-scale feature representations.

The second improvement is with regards to computational efficiency. ViT computes global self-attention, which scales quadratically with image size, making it impractical for high-resolution images. Swin mitigates this by restricting self-attention to a fixed number of local non-overlapping windows, resulting in linear computation complexity with image size. To maintain global context, consecutive transformer blocks apply attention to shifted windows, allowing for cross-window communication while preserving efficiency. The Swin model is also introduced in multiple sizes, including Swin-T, Swin-S, and Swin-B.

Twins-PCPVT [117] is a hierarchical vision transformer that revisits the design of spatial attention to address computational efficiency and deployment challenges. While the Swin Transformer effectively reduces computational complexity using shifted windows, it can introduce uneven window sizes, which can cause difficulties during deployment. To mitigate this, the authors of Twins-PCPVT propose an architecture that builds on Pyramid Vision Transformer (PVT) [133] and Conditional Position Encoding Vision Transformer (CPVT) [134] to address these challenges while maintaining a simple and effective global attention mechanism.

PVT introduces a multi-stage pyramid structure to improve performance on dense prediction tasks such as object detection and semantic segmentation. Instead of computing self-attention globally across all tokens as in standard self-attention, PVT computes it only on a subsampled version of the input tokens which reduces the computational demands. The authors of Twins-PCPVT find

performance problems arising from the utilized absolute position encodings and replace them with conditional positional encodings from [134], resulting in an improved accuracy.

Unlike the Swin Transformer, which combines local and global attention, Twins-PCPVT exclusively employs global attention. The Twins-PCPVT family includes models of different sizes, such as Twins-PCPVT-S, Twins-PCPVT-B, and Twins-PCPVT-L.

Twins-SVT [117] is a hierarchical vision transformer introduced in the same publication as Twins-PCPVT. Unlike Twins-PCPVT, which only uses global attention, Twins-SVT introduces a hybrid attention mechanism that combines local and global attention. To do so, it proposes a novel spatial attention mechanism called spatially separable self-attention, which is inspired by separable depthwise convolutions and improves computational efficiency compared to PVT. It consists of locally-grouped self-attention (LSA) and global subsampled attention (GSA).

LSA captures fine-grained and short-distance information by dividing the 2D feature maps into non-overlapping sub-windows, restricting self-attention computation to within these windows. To allow communication between different sub-windows for global attention, GSA uses a single representative token for each sub-window to summarize its most important information. These representatives interact with each other across windows, allowing for global information exchange while keeping computational complexity low. The Twins-SVT family includes models of different sizes, such as Twins-SVT-S and Twins-SVT-B.

ConvNeXt [113] is a CNN designed through a systematic architecture study with the goal to modernize ResNets towards the design principles of vision transformers. The authors explored a ResNet-50 backbone and investigated the influence of incorporating several existing components. The key design choices can be summarized as macro design, ResNeXt adoption, inverted bottleneck, larger filter size, and layer-wise micro design.

On macro level, the authors change the stage compute ratio and substitute the stem cell with a simpler patchify layer. They further adopt the ResNeXt

design of multiple branches and increase network width. Furthermore, they use inverted bottlenecks inspired by MobileNetV2 [114] and larger filter sizes of 7×7 in depthwise convolution layers. On a micro level, they introduce various design changes such as moving from a ReLU [125] activation function to GELU [135], reducing the overall number of activation functions in each block, and using Layer Normalization [136] instead of Batch Normalization [137].

The ConvNeXt family consists of models of different sizes, including ConvNeXt-S, -B, and -L, which are designed to match the complexity of transformers Swin-S, -B, and -L, respectively.

2.3.2 Object Detection Architectures

Faster R-CNN [107] is an object detection architecture that performs object detection in two steps. In the first step, it generates region proposals of potential objects. In the second step, these proposals are classified to object classes. Faster R-CNN directly builds on the previous version Fast R-CNN [138] which uses the selective search algorithm [139] to generate region proposals, a CNN to extract feature maps from the image, ROI-Pooling to project the region proposals onto the feature maps, as well as fully-connected layers to predict the class of the object and to adjust the final bounding box. Faster R-CNN improves the speed of the object detection by replacing the computationally expensive selective search algorithm for region proposal generation with a Region Proposal Network (RPN). The RPN is constructed to work on the feature maps extracted from the CNN and is therefore merged directly into the network structure. It takes the feature maps of the last layer and uses fully-convolutional layers to output region proposals and objectness scores at each location of the feature map. The region proposals are predicted relative to predefined anchor boxes that serve as references boxes. In Faster R-CNN, usually nine anchor boxes are used which are composed from three different aspect ratios at three different scales. The objectness score measures whether there is an object in that region proposal or not. It is used to reduce the number of region proposals to be classified by just considering the top-ranked proposals. Similarly to Fast R-CNN, the region proposals are then projected back onto the feature maps

using ROI-Pooling and a pre-defined number of features are extracted to be used in the fully-connected layers.

SSD [118] is a one-stage object detection architecture that uses a single neural network to detect objects and does not rely on explicit region proposal generation. It performs predictions on multiple feature maps with different resolutions coming out of the CNN backbone. The utilized feature maps get progressively smaller throughout the CNN, where the early, higher-resolution ones are mainly used to detect small objects, while the later, lower-resolution feature maps are used to detect larger objects. Similar to the anchor boxes from Faster R-CNN, SSD utilizes default boxes with different aspect ratios and scales. At each position of the considered feature maps and for each default box, the model predicts a score for the presence of an object of each category as well as adjustments relative to the reference box. During training, most of the default boxes do not contain an object, i.e., are negative examples, which is the result of the dense sampling method using the reference boxes at each feature map position. This results in a significant class-imbalance compared to the positive examples that contain an object. To reduce the negative effects of the class-imbalance on the stability of the training, SSD employs a so-called hard negative mining strategy. Instead of using all negative examples to calculate the training loss, only the negative examples with the highest contribution to the overall loss are considered, so that the ratio of positive and negative examples matches a pre-defined value.

RetinaNet [119] is a one-stage object detection architecture that mainly addresses the challenge of foreground-background class imbalance faced by one-stage detectors because of the dense sampling of object locations. The authors investigate reasons why one-stage detectors trail the accuracy of two-stage detectors and identify an extreme foreground-background class imbalance as the main cause. They argue that most bounding boxes contain easy negative examples that do not carry useful information for the learning process but that a large number of these can comprise a majority of the loss and therefore dominate the gradient during training. RetinaNet proposes to mitigate this problem by modifying the standard cross-entropy loss to down-weight the contribution of easy, already well-classified examples. To do so, they introduce the Focal Loss which contains a modulating factor to the cross-entropy loss. In

contrast to hard negative mining, where only a small set of anchors is used and many easy examples are completely discarded during training, the calculation of Focal Loss uses all anchors in each sampled image. RetinaNet also uses anchor boxes similar to [107] as well as two fully-convolutional subnetworks for the prediction of class scores and for regressing bounding box offsets to the anchor boxes.

FCOS [120] is a fully convolutional one-stage object detection architecture. In contrast to the architectures described above, it solves object detection in a per-pixel prediction fashion similar to semantic segmentation. It does not utilize anchor boxes or region proposals and therefore avoids corresponding computations and hyperparameters. Instead, at every location in the utilized feature maps, FCOS directly regresses the target bounding box by predicting a class as well as a four-dimensional vector describing the distance from the location to the four sides of a corresponding bounding box. To reduce the number of ambiguous locations, i.e., locations which could be assigned to multiple ground truth bounding boxes, it uses multi-level predictions. Therefore, multiple feature maps are utilized to make predictions and each feature map is responsible to detect objects of specific sizes. In case there are still ambiguous locations, the bounding box with the smallest area is chosen as the regression target. As the authors observed a problem with low-quality predictions coming from locations far away from the center on an object, they further introduce a center-ness branch which predicts the normalized distance from a feature map location to the center of the object. During inference, the center-ness value is multiplied with the corresponding classification score and therefore used to down-weight bounding boxes that are created at locations far away from the center of an object. As a result, these low-quality bounding boxes are more likely to be filtered out using non-max suppression.

Feature Pyramid Network (FPN) [140] is not an object detection architecture itself but an architectural enhancement for the integrated backbones designed to address the challenge of detecting objects at various scales. To address the challenge, it builds a feature pyramid consisting of multiple levels at different spatial resolutions using feature maps from the CNN. The predictions of the object detector are then made independently on each level of the pyramid. The idea of FPNs is to combine low-resolution, semantically strong features from

deep feature maps with high-resolution, semantically weak features to create high-level semantics on each level. As a result, the construction of the FPN involves a bottom-up pathway, a top-down pathway, and lateral connections. The bottom-up pathway is the regular feedforward computation of the CNN. In addition, a top-down pathway is created that upsamples the lower-resolution but semantically stronger feature maps successively. These upsampled feature maps are enhanced with feature maps from the bottom-up pathway via lateral connections that combine both maps by element-wise addition. The lateral connections are introduced to improve the localization accuracy as the feature maps of the bottom-up pathway are subsampled fewer times. To reduce aliasing effects of the upsampling, the feature maps are transformed using 3×3-convolutions before being used for predictions. As the FPN exploits the CNNs inherent multi-scale, pyramidal structure, it introduces only marginal extra cost. Furthermore, as the concept is independent of the backbone, it can be integrated into most object detection architectures.

2.3.3 Semantic Segmentation Architectures

FCN [122] is a semantic segmentation architecture that leverages a fully convolutional design to enable efficient and accurate pixel-wise predictions. The architecture consists of an encoder-decoder structure, where the encoder is the backbone that extracts features from the input image. For decoding, the fully-connected layers of the backbone are replaced with convolutional layers to perform in-network upsampling of the features, allowing the model to produce pixel-wise predictions. These upsampling convolutional layers are sometimes also called deconvolution layers. A further key innovation of FCN is the definition of a skip architecture, which combines coarse semantic information from deeper layers and fine appearance information from shallow layers. This design enables the model to capture both local and global contextual information and to produce accurate and detailed segmentations.

PSPNet [121] builds on FCN and aims to improve its semantic segmentation performance by better incorporating global context information. To achieve this, PSPNet introduces a Pyramid Pooling module that captures global scene

information at different scales and varying sub-regions, and acts as a global contextual prior that enables the model to better understand complex scenes. The module takes the last feature map of the backbone and applies global pooling under different pyramid scales. This process involves applying global pooling on the whole feature map as well as on different sub-regions corresponding to each pyramid level. The resulting feature maps are upsampled and concatenated to the original feature map, creating a representation that combines local and global context information. The enhanced feature maps are then fed into a convolutional layer to produce the final per-pixel predictions, allowing it to capture local patterns and global relationships within the input image.

DeepLabV3 [123] represents the third major iteration of the DeepLab architecture for semantic segmentation, improving upon the limitations observed in earlier versions. The first version [141] introduces the use of atrous convolutions, also known as dilated convolutions. They allow the network to have a larger receptive field, without increasing the number of parameters or reducing the resolution of feature maps, by introducing spaces, also called holes, between the kernel elements. To upsample the feature maps to the original size for final classification, the model uses bilinear interpolation at negligible computation costs compared to deconvolutions. Furthermore, DeepLab uses a fully-connected Conditional Random Field (CRF) [142] to improve object localization and to capture fine object boundary details. DeepLabV2 [143] extends the development by introducing the Atrous Spatial Pyramid Pooling (ASPP) module, which applies multiple parallel atrous convolutions with different dilation rates, i.e., different amount of space between the kernel elements, to capture objects and image context at multiple scales using various fields of view. DeepLabV3 then revisits and improves the ASPP module further by adding image-level features using global average pooling, which enrich the feature representation with global context, and adding batch normalization within ASPP. Furthermore, DeepLabV3 removes the need for CRF post-processing while improving performance compared to previous versions.

DeepLabV3+ [124] extends DeepLabV3 by introducing an encoder-decoder structure, incorporating a simple decoder module to refine segmentation results, particularly along object boundaries. This decoder upsamples the high-level

features of the backbone and combines them with low-level features from earlier layers, refining edge information and enhancing segmentation accuracy. DeepLabV3+ further integrates depthwise separable convolutions into the ASPP and decoder modules to reduce computation complexity while maintaining performance.

UPerNet [108] is a semantic segmentation architecture that builds on FPN and PSPNet. It extends the backbone with an FPN structure as presented above, enabling the model to aggregate features from different stages of the backbone. To increase the receptive field and to enhance the global context of high-level features, a Pyramid Pooling module, inspired from PSPNet, is applied to the last feature map of the backbone, before feeding it into the top-down branch of the FPN. Using the FPN structure, the enriched high-level features are progressively upsampled and merged with features from earlier layers via lateral connections to enhance spatial details. The merged features from the different pyramid levels are then fused to form a high-resolution representation that is used to output the final per-pixel predictions.

2.4 Image Augmentations

Image augmentation techniques are a set of methods used to modify certain image characteristics and to increase the overall diversity of image training datasets. Basic image augmentations, such as adding blur or adjusting contrast, have shown to improve performance of deep learning models on real-world datasets [85, 86] and robustness to real-world distribution shifts [87].

In this work, the influence of 25 pixel-level image augmentation techniques on the sim-to-real generalization of deep learning models is investigated. Compared to spatial-level augmentations, which alter the input image spatially through transformations such as image flipping while leaving its visual appearance largely unchanged, pixel-level augmentations modify the visual characteristics of images, e.g., by adding blur or altering colors. Because of that, they are more relevant to the considered research question.

This section gives a brief overview of the investigated augmentations. For all augmentations, their implementation from the Albumentation library [85] is utilized. For more details about a specific augmentation, the reader is referred to the online documentation of the library [144, 145].

Following 25 pixel-level augmentations are investigated in this work. An illustration of their effects is given in Fig. 2.3 at the end of this section.

AdvancedBlur [144]: Applies a Generalized Gaussian Blur using randomized parameters to create a diverse range of blur effects. This augmentation also allows for anisotropic blurring, i.e., independent control over horizontal and vertical blur strengths, as well as kernel rotation. Additionally, the augmentation injects multiplicative noise into the kernel, resulting in more realistic and varied blur effects.

Blur [144]: Applies a uniform box blur by utilizing a square kernel. The blur kernel size is randomly chosen, resulting in varying levels of blur intensity. By averaging all pixels within the kernel area, it can reduce noise but can also

compromise image detail. This method is computationally efficient, but may produce less natural results compared to Gaussian blur.

Contrast Limited Adaptive Histogram Equalization (CLAHE) [145]: Enhances the contrast of the input image by applying a form of histogram equalization. Unlike traditional histogram equalization, CLAHE divides the image into small regions and applies equalization locally, resulting in a more balanced contrast enhancement.

ColorJitter [145]: Randomly perturbs the color properties of an image, specifically altering its brightness, contrast, saturation, and hue. The order in which these transformations are applied is randomly determined for each image, introducing variability in the resulting color appearance. The magnitude of the changes is controlled by ranges that are applied as multiplicative factors for brightness, contrast, and saturation, and as an additive factor for hue.

Defocus [144]: Applies a defocus blur to simulate an out-of-focus camera effect. This is achieved by using a combination of disc kernels, which mimic the shape of an aperture of a camera, and Gaussian blur, which softens the edges of the disc kernel to create a more realistic effect.

Downscale [145]: Decreases image quality by simulating the effect of low-resolution images. This is achieved by reducing the image resolution via downscaling and then restoring its original size via upscaling, resulting in a loss of detail. The downscaling factor is randomly selected from a specified range, introducing variability in the degradation process.

Emboss [145]: Applies an embossing effect, which gives the impression of raised or recessed areas. By emphasizing differences in adjacent pixel values, embossing highlights the edges and contours of an image and creates a sense of depth.

Equalize [145]: Adjusts the contrast of the input image by applying histogram equalizing, a technique that redistributes pixel intensity values to create a more uniform distribution. The equalization is done for each color channel separately.

FancyPCA [145]: Applies Principal Component Analysis (PCA) to the color channels of the input image and then adjusts the image by adding multiples of the principal components, resulting in altered colors and intensity. The magnitude of the adjustment is controlled by a random variable with a Gaussian distribution.

GaussNoise [145]: Introduces Gaussian noise into the input image, simulating a type of sensor noise that can occur, e.g., during digital image acquisition. The noise is sampled for each color channel independently.

GaussianBlur [144]: Applies a Gaussian blur to the input image, reducing image noise and detail while creating a smoothing effect. The blur is achieved using a Gaussian filter with a randomly sized kernel, allowing for a range of blur effects from subtle to strong. By normalizing the kernel, the augmentation preserves the image luminance.

GlassBlur [144]: Simulates the visual distortion caused by looking through textured glass, creating a frosted glass-like effect. It does so by locally shuffling pixels in the image.

HueSaturationValue [145]: Adjusts the hue, saturation, and value of an image by manipulating its HSV channels. The augmentation adjusts each channel independently, allowing for a wide range of color and brightness modifications.

ISONoise [145]: Simulates the effects of high ISO settings in digital photography by introducing camera sensor noise. It models two primary components of ISO noise. First, color noise, which causes random shifts in color hue. Second, luminance noise, which results in random variations in pixel intensity.

MedianBlur [144]: Applies a median filter to the input image, effectively removing salt-and-pepper noise while preserving edges. It is especially useful for noise reduction and more robust to outliers than, e.g., Gaussian blur.

MotionBlur [144]: Simulates blur that can result from movements such as camera shake or object motion during image capture. The effect is generated

by applying a directional kernel to the input image.

RGBShift [145]: Applies a constant uniform shift to the values of each channel in the input RGB image. The shift is independent for each channel and randomly selected from a specific range for each image. By doing so, the augmentation simulates variations in color tone and intensity.

RandomBrightnessContrast [145]: Randomly adjusts the brightness and contrast of an image, simulating a wide range of lighting and contrast variations. The augmentation first applies a contrast adjustment, followed by a brightness adjustment.

RandomGamma [145]: Applies a random gamma correction. By doing so, it can modify the brightness of an image while preserving relative differences between lighter and darker areas in the image.

RandomToneCurve [145]: Manipulates the tone curve of the input image, altering the relationship between bright and dark areas using a non-linear random S-curve. By doing so, it modifies the histogram of the image and can adjust the contrast and brightness. As the same curve is applied to all channels, the color balance remains unaltered.

RingingOvershoot [145]: Simulates ringing or overshoot artifacts near sharp transitions that can result from digital image processing such as sharpening or edge enhancements.

Sharpen [145]: Applies a kernel-based interpolation method to sharpen an image. Specifically, the augmentation employs the Laplacian operator to enhance image details but may also produce artifacts.

ToSepia [145]: Applies a sepia effect transforming the image into having a warm and brownish tone. The augmentation uses a specific, pre-defined transformation matrix that is applied to the color channels of the image.

UnsharpMask [145]: Sharpens the input image using the Unsharp Masking technique, which increases edge contrast and results in a perceived increase of

sharpness. The augmentation achieves this by applying Gaussian blur to the image, creating a mask derived from the blurred version and combining it with the original image to highlight edges and fine details.

ZoomBlur [144]: Simulates a blur effect that can be caused when a camera rapidly zooms in or out, or moves towards or away from an object. This results in a blur pattern that radiates outward from a central point in the image.

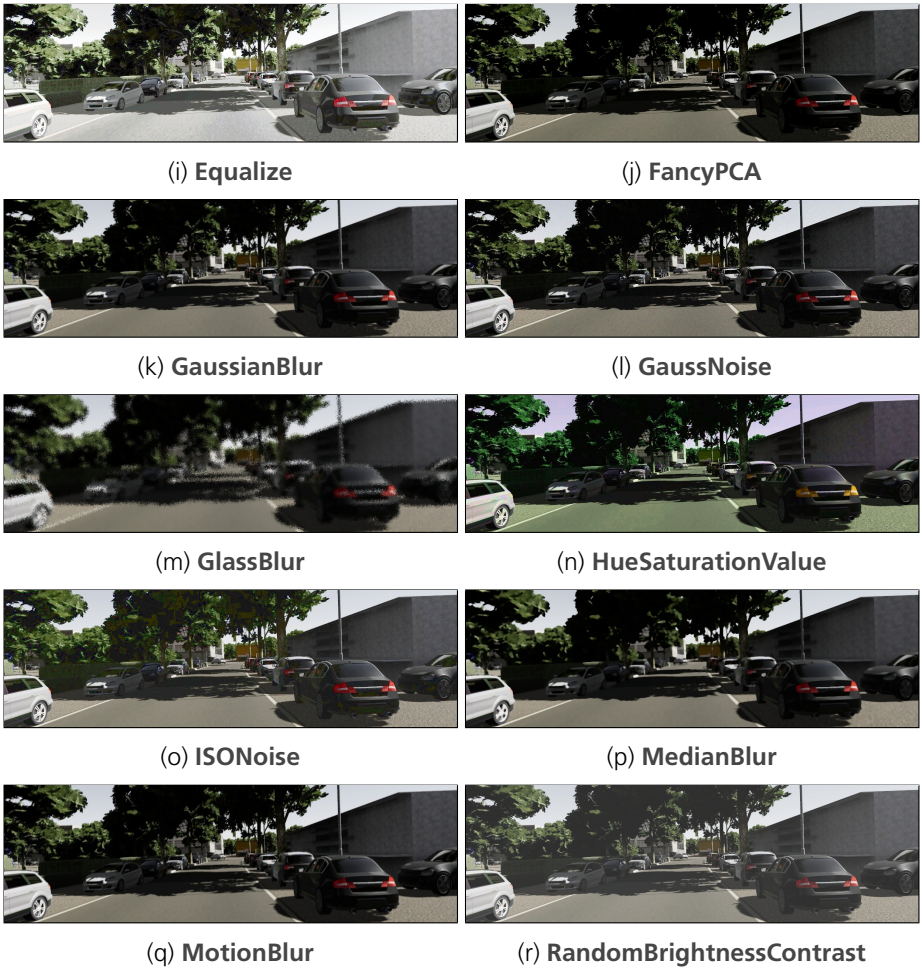
(a) **No Augmentations** (b) **AdvancedBlur**

(c) **Blur** (d) **CLAHE**

(e) **ColorJitter** (f) **Defocus**

(g) **Downscale** (h) **Emboss**

Figure 2.3: **Illustration of the effects of the considered image augmentation techniques. Image without augmentations from [146]**



(i) **Equalize** (j) **FancyPCA**

(k) **GaussianBlur** (l) **GaussNoise**

(m) **GlassBlur** (n) **HueSaturationValue**

(o) **ISONoise** (p) **MmedianBlur**

(q) **MotionBlur** (r) **RandomBrightnessContrast**

Figure 2.3: **(Cont.) Illustration of the effects of the considered image augmentation techniques. Image without augmentations from [146]**

(s) **RandomGamma** (t) **RandomToneCurve**

(u) **RGBShift** (v) **RingingOvershoot**

(w) **Sharpen** (x) **ToSepia**

(y) **UnsharpMask** (z) **ZoomBlur**

Figure 2.3: **(Cont.) Illustration of the effects of the considered image augmentation techniques. Image without augmentations from [146]**

3 Datasets

To ensure the generalizability and robustness of the findings, this study utilizes three distinct use-cases with corresponding datasets to address the research questions. By conducting experiments and investigations across these diverse datasets, it aims to provide a comprehensive understanding of influences on the sim-to-real generalization of deep learning models.

The three use-cases represent a wide range of applications, each with unique characteristics and challenges. First, the task of drogue detection during air-to-air refueling is considered, which presents a visually simple yet critical aviation use-case. Second, a semantic segmentation use-case from a low-altitude UAS perspective is explored, which involves complex aerial imagery. Third, a semantic segmentation use-case from the autonomous driving domain is examined, which provides different perspectives and diverse lighting conditions.

For each use-case, both real-world and synthetic datasets are required. This chapter introduces these datasets. A particular focus is paid to the two synthetic datasets created within the scope of this work.

3.1 Air-to-Air Refueling

The first investigated use-case focuses on drogue detection during drogue-relative navigation in air-to-air refueling, a visually simple yet critical application that serves as a baseline for further analysis. The simplicity of this use-case

enables the generation of synthetic images that closely resemble real-world scenarios, thereby minimizing the content gap. To ensure unbiased data, i.e. data that is not skewed towards specific conditions potentially present in the real-world images such as the distribution of drogue positions or flight altitude, the synthetic images are created in accordance with standardized air-to-air refueling procedures instead of directly mirroring the real-world images.

To achieve this, Section 3.1.1 describes the necessary air-to-air refueling background and outlines the relevant refueling procedures. The focus is laid on the aspects relevant for visual data generation as well as the aircraft and procedures depicted in the available real-world data presented in Section 3.1.2. Building on this background, Section 3.1.3 outlines the derived parameters for data generation and presents the toolchain developed to generate the synthetic data.

A first description of the data and the generation process was published in [88].

3.1.1 Background

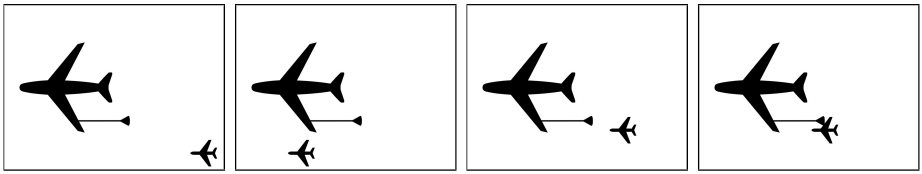
Air-to-air refueling is the process of transferring fuel from one aircraft to another during flight. It allows aircraft to extend their range and carry heavier payloads. Especially in military context, air-to-air refueling is important for various missions, as it enables smaller aircraft such as fighter jets to perform tasks that would otherwise be beyond their operational range or capabilities. Some entities even consider air-to-air refueling to be one of the essential capabilities of airpower [147].

In air-to-air refueling, the two primary methods employed by North Atlantic Treaty Organization (NATO) member states today are the Flying-Boom method, also known as Boom-Receptacle, and the Probe-and-Drogue method [148]. The Flying-Boom system utilizes a rigid tube extending from the tanker aircraft, which is guided into a receptacle on the receiver aircraft. The boom is controlled by a dedicated operator on board of the tanker aircraft. Once connected, fuel

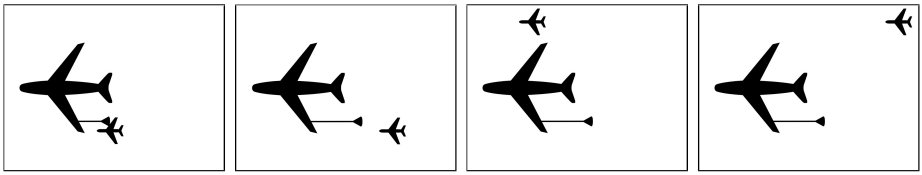
Figure 3.1: **Illustration of the air-to-air refueling methods Flying-Boom (left, [149]) and Probe-and-Drogue (right, [1])**

is transferred from the tanker aircraft to the receiver aircraft through the boom. In contrast, in the Probe-and-Drogue system, the tanker aircraft tows a flexible hose without manual control. It is stabilized by a drogue which contains a funnel that allows fuel to flow into the receiver aircraft when a connection is established. To initiate this connection, the receiver aircraft navigates the probe of its aircraft into the drogue, triggering the mechanism to enable fuel transfer. Fig. 3.1 shows examples of both refueling methods.

This work considers air-to-air refueling with the Probe-and-Drogue method. The method offers several advantages over the Flying-Boom system but has the significant drawback of imposing a high workload on the pilot of the receiver aircraft. While the close formation flight makes the flight task dangerous, the pilots often additionally operate under stress, as they typically only have a limited time window to complete the refueling process. Given the benefits of the system, which are however tied to a challenging task for the pilot, research in recent years has increasingly focused on developing methods to support the pilot during this maneuver or to even automate the process. For instance, various automation and control concepts for Probe-and-Drogue refueling have been explored in the literature [150].



(a) **Receiver aircraft joins tanker** (b) **Receiver aircraft takes Echelon Left position** (c) **Receiver aircraft takes Astern Left position** (d) **Receiver aircraft maneuvers probe into drogue**



(e) **Receiver aircraft pushes back part of the hose** (f) **Receiver aircraft takes Astern Left position** (g) **Receiver aircraft takes Echelon Right position** (h) **Receiver aircraft leaves formation**

Figure 3.2: **Air-to-air refueling procedure of the considered use-case. Based on [148]. First published in [88]**

Air-to-Air Refueling Procedure using the Probe-and-Drogue Method The standardized procedures for air-to-air refueling within NATO are documented in NATO ATP-3.3.4.2 [148]. The ATP further allows NATO member states to supplement it with Standards Related Documents (SRDs) that add specific national procedures as needed. This section describes the general air-to-air refueling procedure based on NATO ATP 3.3.4.2, focusing on the use-case relevant to this work: refueling of a fixed-wing aircraft using the Probe-and-Drogue method with air-to-air refueling pods. For further explanations and possibly omitted details that are not needed within the scope of this work, the reader is referred to [148]. A graphical representation of the procedure described below is provided in Figure 3.2.

After rendezvous, i.e., once the tanker aircraft and the receiver aircraft have

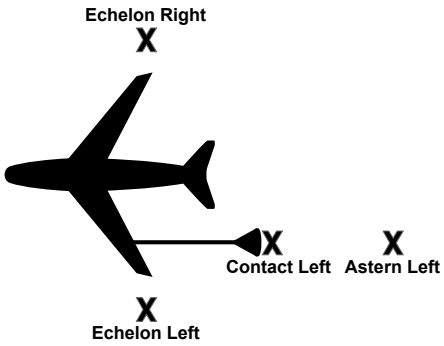


Figure 3.3: **Positions during air-to-air refueling, including designations. Figure not to scale. Based on [151]. First published in [88]**

established visual contact, the task of the receiver aircraft is to take and hold various positions relative to the tanker aircraft or the refueling drogue. These positions and their designations are illustrated in Fig. 3.3. The first task of the receiver aircraft is to establish a close formation flight (Fig. 3.2a). This is typically achieved by taking the Echelon Left position (Fig. 3.2b), unless otherwise directed by the tanker aircraft. The exact position depends on the tanker aircraft and should be described in the relevant SRDs. From Echelon Left, the receiver aircraft is directed to the Astern position behind a free refueling drogue (Fig. 3.2c). This position is approximately 5-20 feet behind the drogue with a closure rate of zero. Astern is also the position where the receiver aircraft transitions from tanker-relative to drogue-relative navigation. From the Astern position, the receiver is cleared by the tanker aircraft to establish contact with the refueling drogue. To establish contact, the receiver aircraft maneuvers its probe into the drogue. The capture speed at contact must be within a certain range. If it is too low, the coupling will not engage. If it is too high, damage may occur [150]. When contact is established, the probe engages with the coupling latch, creating a fuel-tight connection (Fig. 3.2d). The receiver aircraft then moves closer to the tanker aircraft, pushing a section of the refueling hose back into the refueling pod. When a sufficient amount of the hose has been rewound, the valve opens and fuel is pumped to the receiver aircraft. Typically, the required retraction length of the hose is approximately 2 meters,

but can be defined in the SRDs. This position is referred to as the Contact position (Fig. 3.2e) and the receiver aircraft has to hold this position during the refueling process. When refueling is complete, the receiver aircraft is cleared to disconnect. It then reduces speed to slowly fall back and to stabilize in the Astern position (Fig. 3.2f). Afterwards, the receiver aircraft is directed to the Echelon Right position (Fig. 3.2g). Similar to the Echelon Left position, the exact position depends on the tanker aircraft and should be described in relevant SRDs. Once the receiver aircraft has taken the position, other aircraft may be refueled and the receiver aircraft can break away from the formation flight (Fig. 3.2h).

Relevant for the use-case considered in this work, i.e. drogue-relative navigation, is the sequence from Astern to Contact.

3.1.2 Real-World Data

The *Wehrtechnische Dienststelle für Luftfahrzeuge und Luftfahrtgerät der Bundeswehr* (WTD 61) recorded and provided real-world videos of air-to-air refueling sequences for research purposes. The videos show an Airbus A400M with an air-to-air refueling pod under its left wing and a trailing refueling drogue. A Panavia 200 (PA-200) Tornado aircraft performs multiple refueling runs with this drogue. The videos were recorded by a camera mounted on the probe of the Tornado, capturing images with a resolution of 720x567 pixels and a field of view of approximately 53°x41.7°. They were recorded at 25 frames per second (FPS).

A total of four videos featuring multiple refueling runs are available. The videos start with the deployment of the probe from the aircraft body, end with its retraction, and show one or multiple refueling runs in-between. The runs do not always start from the Echelon Left position but sometimes a new contact attempt is initiated directly from the Astern position after a disconnect.

The use-case considered in this work focuses on detecting the refueling drogue from the beginning of drogue-relative navigation to contact, i.e., from Astern to

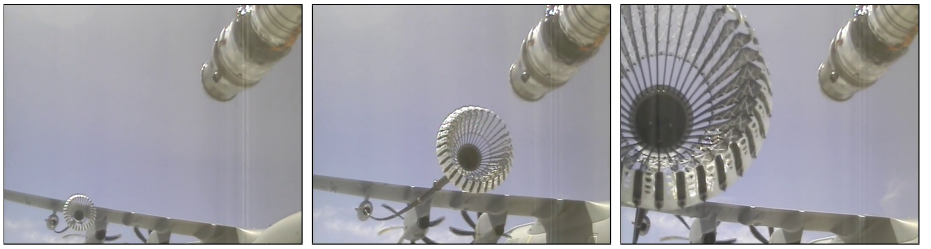


Figure 3.4: **Examples of the available real-world images recorded from a camera mounted on the receiver aircraft during air-to-air refueling. Source: [84]. First published in [88]**

Contact. However, approaches starting from Echelon Left will also be included in the real-world evaluation dataset. As the refueling drogue is not visible at the beginning of these approaches, they allow the evaluation of the detection model output when no drogue is in the field of view of the camera.

Sometimes the drogue is missed during an approach and the aircraft falls back slightly to attempt another coupling. Since the drogue should be detected until successful coupling is achieved, missing the drogue in the approach is not relevant for the detection task. Therefore, this work defines the end of a refueling approach to be when contact is established between the probe and the drogue. A subsequent approach begins when the receiver aircraft returns to the Echelon Left or Astern position and starts to move towards the refueling drogue.

To extract the refueling approaches from the full videos, the program Losslesscut [152] is used. It allows to losslessly cut the videos into multiple smaller segments without requiring re-encoding, thereby preserving video quality. However, cutting is only possible at so-called key-frames which occur every few seconds. Due to the relatively low relative velocity, this constraint does not lead to the loss of relevant frames. Instead, it allows to keep the images as close to the originals as possible. Subsequently, the tool FFmpeg [153] is used to extract and save individual frames from the videos. Fig. 3.4 shows some exemplary extracted frames.

The images from five randomly selected refueling approaches are manually annotated, resulting in a total of 5,487 annotated real-world images. Images from the shortest refueling approach, containing 518 images, are used for training and validation. They are randomly split into a subset of 80% (414 images) for training, while the remaining 20% (104 images) are designated for validation. When no real-world training images are needed, as is the case for most of this work except for Section 6.1, the complete approach is used as the validation dataset. Taking the shortest approach aligns with the assumption that usually only limited real-world data is available in aviation use-cases, if existent at all. The remaining 4,969 images from the other refueling approaches are combined in an evaluation dataset.

3.1.3 Synthetic Data

This section presents the developed toolchain and the data generation process to create the synthetic air-to-air refueling images that should closely resemble the real-world images. The utilized tools and 3D models are outlined, followed by a description of how these components are integrated to enable automated data generation, and a presentation of the parameters used for the final dataset creation.

Components of the Toolchain for Synthetic Data Generation

A range of tools and components are employed to generate and extract synthetic images of diverse air-to-air refueling scenarios from a game engine.

Unreal Engine The foundation for the generation of the synthetic images is the game engine Unreal Engine 4.27 [37]. This engine is utilized to create a virtual world and visualize the air-to-air refueling situation. Developed by Epic Games, the Unreal Engine is a game engine that has been continuously

improved since its first release in 1998 [154]. The engine provides a range of features, including for designing virtual worlds, rendering of scenes and animations, and creating interactive environments, as well as physics systems. As a result, it has become a widely-used platform for video game development [155, 156] and is also employed in other fields, such as automotive [157] and architecture [158]. The long history of the Unreal Engine and an active community have created a vast ecosystem, including forums and support resources, as well as the Unreal Engine Marketplace (now known as Fab [159, 160]), which offers a variety of digital assets for integration into Unreal Engine projects. It also offers 3D models of various aircraft. Furthermore, Unreal Engine contains a cloud system, called Volumetric Clouds [161], which is employed for the generation of clouds in the virtual world.

Cesium Ion To create a virtual world in which the aircraft can be placed, data from Cesium Ion [162] is loaded into the Unreal Engine using the Cesium for Unreal plugin [163, 164]. It provides a 3D visualization of the Earth for the Unreal Engine, onto which aerial images can be projected as the texture. Specifically, Cesium World Terrain and Bing Map Aerial Imagery from Cesium Ion are loaded into the Unreal Engine, which contain a detailed terrain model compiled from various sources, textured with Bing Map aerial images [165, 166]. Furthermore, the virtual world is automatically populated with buildings, represented by 3D polygons, from OpenStreetMap using the OpenStreetMap Buildings option from Cesium. Finally, the virtual world is enhanced with a geographically accurate sun and sky, as well as an optical representation of the atmosphere, using Cesium SunSky [167].

AirSim AirSim [41] is a plugin for the Unreal Engine 4 that enables the simulation of drone flights and autonomous driving. Developed by Microsoft and afterwards released as open-source software, it provides programming interfaces for placing objects and extracting visual sensor data. For the latter, various virtual sensors can be attached to objects, including RGB cameras, LiDARs, and distance sensors. Additionally, AirSim allows to extract segmentation images, which link each pixel to their respective object class name. These can be utilized

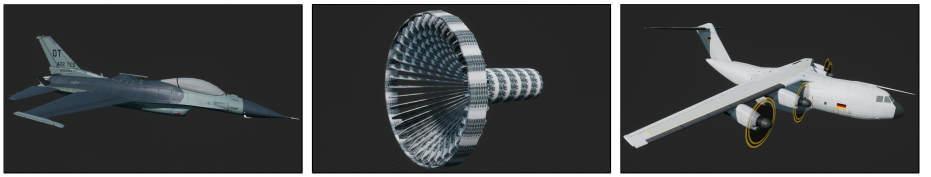


Figure 3.5: **Imported and utilized 3D models of the receiver aircraft, refueling drogue, and tanker aircraft (from left to right). First published in [88]**

to create bounding boxes for the objects of interest. Overall, AirSim is employed to set the poses of the involved objects and to extract RGB and segmentation images.

3D Models To visualize the air-to-air refueling scenario in the virtual world, various 3D models are imported into the Unreal Engine. As the tanker aircraft, a DLR-internal model of an aircraft similar to an Airbus A400M is used. As the receiver aircraft, a modified General Dynamics F-16 model from [168] is employed. The real-world images were captured from a Panavia PA-200 Tornado but a corresponding 3D model was not available. It is assumed that the different receiver aircraft model does not significantly impact the accuracy of the drogue detection model because the probe is a separate component whose position relative to the drogue is not fixed in the images and therefore should not be considered by the model to detect the drogue. Furthermore, only a small portion of the refueling probe is visible in the images. As the refueling drogue, a DLR-internal model is used. The hose is not simulated in detail, but interpolated linearly between the aircraft and the refueling drogue. A more detailed model of the hose could be created and integrated in the future but was deemed to be infeasible in the scope of this work. It is assumed that the hose does not have a significant impact on the accuracy of the drogue detection model because its position relative to the drogue is not fixed in the images, it never occludes the drogue, and sometimes is not even visible in the images. Therefore, it should not be considered by the detection model to detect the drogue. The imported and utilized 3D models are shown in Fig. 3.5.

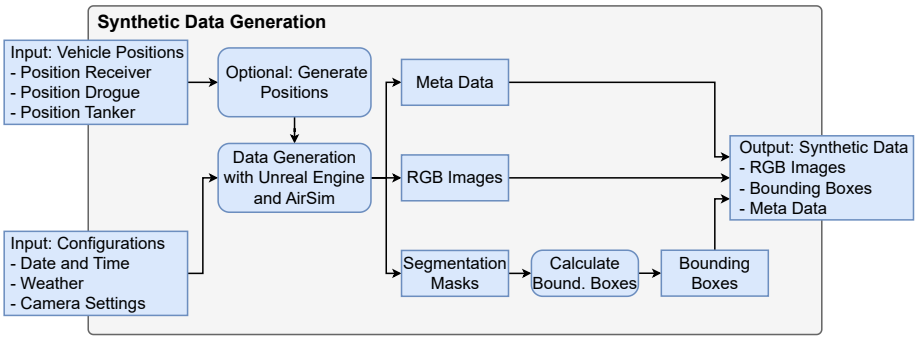


Figure 3.6: **Toolchain for generating and extracting synthetic images and bounding boxes of air-to-air refueling using the Unreal Engine. Adapted from [88]**

Structure of the Toolchain for Synthetic Data Generation

Using the presented components, a toolchain is developed for generating and extracting synthetic images, including bounding boxes for relevant objects. The setup and workflow are illustrated in Fig. 3.6.

The virtual world is rendered in the Unreal Engine containing the 3D models and Cesium components. Using a setup file, the date, time, and weather conditions can be defined by the user, enabling the simulation of various lighting conditions, such as day, dusk, and night, as well as varying cloud intensities. Additionally, camera parameters, including resolution and field of view (FOV), can be set. To position the objects involved in air-to-air refueling, an interface is developed to receive, process, and automatically place them in the Unreal Engine using AirSim. This allows the visualization of real-world flight paths or of those simulated using flight mechanics models, and the extraction of sensor data, such as RGB images. In addition to sensor data, various metadata is extracted. It includes the positions of the objects and environmental conditions.

Furthermore, AirSim is used to extract not only RGB images from the virtual

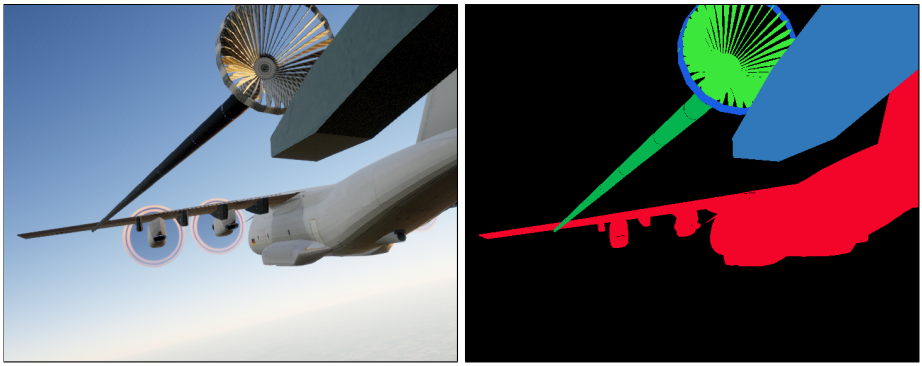


Figure 3.7: **Example RGB image (left) and corresponding semantic segmentation annotation mask (right) from a virtual camera attached to the receiver aircraft. Images are generated and extracted using the presented toolchain. First published in [88]**

camera but also corresponding semantic segmentation annotation masks. In the segmentation masks, each object is colored according to its associated class. In the air-to-air refueling use-case, the classes tanker aircraft, receiver aircraft, refueling drogue body, refueling drogue perimeter, refueling hose, and background are distinguished. Fig. 3.7 shows a generated RGB image with its segmentation mask. Since object detection models require bounding boxes instead of segmentation masks during training, they are derived from the segmentation masks. This is done by placing a bounding box around the pixels of the drogue classes.

A list of all adjustable parameters is given in Table 3.1. The position of the receiver aircraft in the virtual world is set absolutely, while the positions of the refueling drogue and tanker aircraft are set relatively. An illustration of the positioning parameters in the toolchain and the corresponding coordinate system are shown in Fig. 3.8.

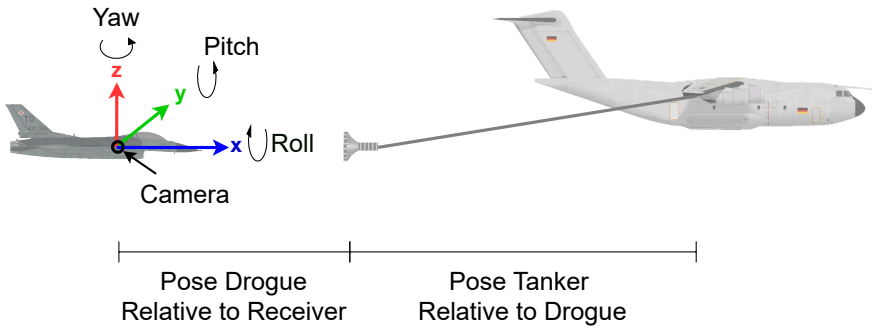


Figure 3.8: **Illustration of the coordinate system and the positioning parameters for the aerial objects in the data generation toolchain. First published in [88]**

Automation of the Toolchain for Synthetic Data Generation

The developed toolchain enables the visualization of the position and orientation of multiple aircraft, including the extraction of sensor data and bounding boxes. For training deep learning object detection models, a diverse training dataset is required, ideally covering all situations faced during deployment. To achieve this, the interface for entering position data is extended to also accept parameter value ranges. If ranges are specified, random combinations are generated, visualized in the Unreal Engine, and sensor data is extracted. For example, this allows specifying the permitted range of distances between the tanker and the receiver aircraft, leading to the creation and extraction of sensor data from a set of random positions in this range. Overall, this enables the automated generation of diverse datasets covering a specified flight envelope.

In this context, an additional option is implemented to allow the position of the drogue to be randomly located within the camera frustum rather than requiring explicit specification. This ensures that the extracted images can contain the drogue at all possible positions within the camera frustum. Furthermore, drogues are also randomly placed outside the camera frustum to obtain images

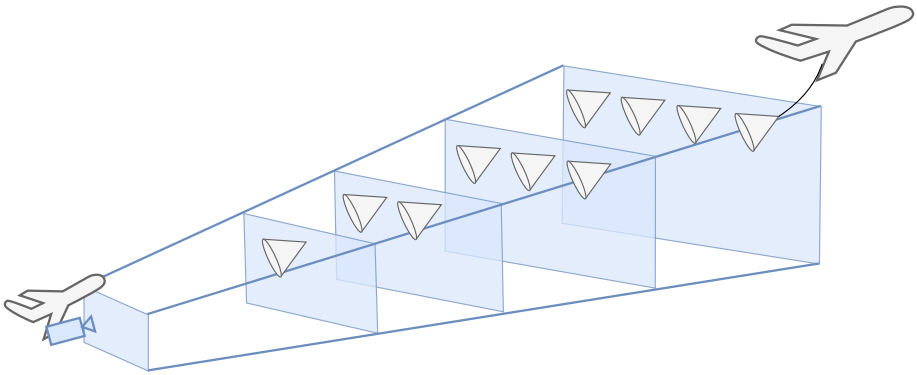


Figure 3.9: **Illustration of the random positioning of the refueling drogue within the camera frustum**

without it, as this may also occur during deployment. Fig. 3.9 exemplarily visualizes this camera frustum including possible drogue positions.

Generation of the Synthetic Images

To generate synthetic images for training deep learning drogue detection models, the presented toolchain is utilized with the parameters listed in Table 3.1. These parameters can be interpreted as the ODD defining the operational range for which the detection model is trained and ultimately is intended to function in. They are derived from the NATO air-to-air refueling procedures presented above and can be seen as a system-level requirement when aiming to develop a system for deployment. This broad range of parameters is selected so that the synthetic data covers the expected conditions of air-to-air refueling scenarios and does not exactly mirror the real-world data. Instead, it forms a general basis in which the conditions depicted in the real-world data should be included.

As 3D models, the tanker, receiver and drogue model as presented above are used. The virtual camera mounted on the receiver is positioned and configured to mimic the real-world camera. The images are generated at a resolution of

Table 3.1: **Parameters used for synthetic data generation**

Parameter	Value / Range
Receiver aircraft 3D model	DLR modified F-16 model
Tanker aircraft 3D model	DLR model similar to A400M
Drogue 3D model	DLR model of drogue
Camera resolution	720x567 pixels
Camera FOV	53° vertical, 41.7° horizontal
Camera exposure	Auto exposure
Receiver altitude	5,000 - 25,000 ft above MSL
Receiver lon and lat	100 km radius around CSO airport
Receiver pitch	-20° - 20°
Receiver roll	-20° - 20°
Receiver yaw	0° - 359° (every geographic direction)
Drogue x-offset to probe tip	0 - 24 ft
Drogue y-offset to probe tip	Random positions in camera frustum
Drogue z-offset to probe tip	Random positions in camera frustum
Drogue pitch-offset to probe tip	-10° - 10°
Drogue roll-offset to probe tip	None
Drogue yaw-offset to probe tip	-10° - 10°
Tanker x-offset to drogue	15 - 27 m
Tanker y-offset to drogue	-10.6 - 10.6 m
Tanker z-offset to drogue	3 - 5 m
Tanker pitch-offset to drogue	-10° - 10°
Tanker roll-offset to drogue	-10° - 10°
Tanker yaw-offset to drogue	-10° - 10°
Date	1st Jan., 1st May, 1st Sept. 2023
Time of day	10 am, 1 pm, 4 pm
Weather - clouds	3 different intensities

720x567 pixels, with a vertical FOV of 53° and a horizontal FOV of 41.7°, using auto-exposure. The refueling altitudes are modeled after the PA-200 Tornado of the real-world dataset, which is able to refuel with the A400M between 5,000 and 25,000 ft above mean sea level (MSL) [169]. To create diverse scenarios, the geolocation of the receiver is randomly set within a 100 km radius of Cochstedt airport (CSO, Lon: 51.853444°, Lat: 11.39997°). Since the maximum pitch and roll angles during refueling are not specified in the relevant standard documents, they are set to +/-20° to introduce variation. The yaw value is randomized between 0-359° to simulate flight in any direction. The distance between the drogue and probe tip in x-direction ranges from 0-20 ft, as per [148], and is expanded by 20% to 0-24 ft in this work to add further variance and buffer. The position of the drogue on the y- and z-axes is randomly chosen within the camera frustum, ensuring it can appear in every possible location in the image. Therefore, the distance in y- and z-direction depends on the camera frustum and is not explicitly set. As the drogue is largely symmetric, roll simulation is neglected. Pitch and yaw values are set to +/-10° to introduce variation. The hose length, as per [148], ranges from 15 to 27 meters, which is used to set the distance from the drogue to the tanker wing. Following [169], the drogue is positioned 3-5 meters below the tanker's air-to-air refueling pod and up to 10.6 meters offset horizontally. Cross-relationships are ignored, which further increases variation in the training data. The roll, pitch, and yaw values of the tanker are set to a maximum of +/-10°. To create variations in lighting, different dates and times of day are selected. Furthermore, to introduce some variation in the weather, different cloud intensities are generated. With the given tools, these parameters can not be selected randomly in the Unreal Engine. Instead, for each combination, a separate environment had to be created and stored. Because of the resulting combinatorial complexity, three different dates and three different times of day are selected. To maximize variation, the dates are evenly spread over a year and times are chosen to ensure enough lighting so that the drogue is visible. Furthermore, three different cloud intensities according to Unreal Engine options (none, medium, and maximum) are used. Overall, this results in 27 separate Unreal Engine environments.

Using the presented toolchain and parameters, synthetic images are rendered and extracted for training deep learning drogue detection models. For the model comparison experiments in Chapter 4, 15,579 synthetic images are gen-

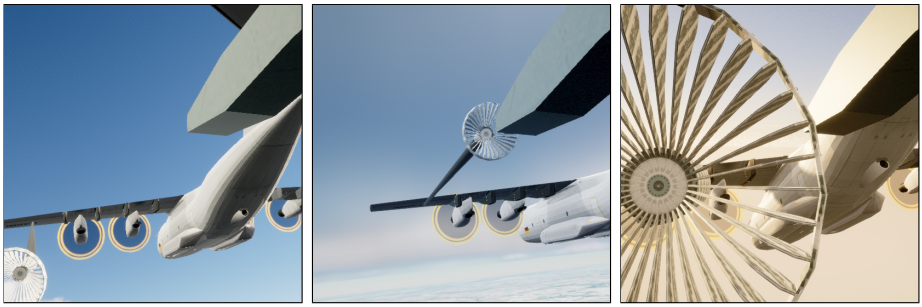


Figure 3.10: **Example synthetic images generated using the developed toolchain and presented parameters. First published in [88]**

erated. Of these, approximately two-thirds are used as the training dataset and the remaining one-third is used as the evaluation dataset. The augmentation comparison experiments in Chapter 5 use a slightly smaller training dataset of 8,000 synthetic images to reduce computational load and improve training speed. Fig. 3.10 shows some of the generated RGB images.

Because of the generation process of the synthetic images, the synthetic and real-world datasets depict the same scenario as can be seen in Figs. 3.4 and 3.10. Although there are no exact matching pairs between the synthetic and real-world images, they are visually comparable. Visual differences are subtle and can mainly be observed with regards to the rotors of the tanker aircraft and the probe of the receiver aircraft. Following the argumentation given in the description of the 3D models, the differences of the probe are accepted as the detection of the refueling drogue can be considered largely independent of this component. Similarly, the detection of the drogue can also be considered largely independent of the rotors as their positions relative to the drogue are not fixed in the images, they never occlude the drogue, and sometimes are not even visible in the images. Therefore, they should not be considered by the detection model to detect the drogue. Furthermore, some differences in photometric characteristics can be observed. While the real-world images exhibit a relatively consistent visual appearance, the synthetic images show a broader spectrum of variation in appearance, resulting in a greater overall diversity. However, this

does not need to deteriorate the detection performance but in contrast may be beneficial for the generalization capability of the detection model.

The generation of 15,579 synthetic images required approximately 11.75 hours on a workstation featuring an Intel® Xeon® W-2265@3.5 GHz processor, 128 GB of RAM, and an NVIDIA® RTX™A5000 graphics card. The rendering and extraction of the RGB images and the semantic segmentation masks required approximately 3.75 hours of the total time. The remaining 8 hours were needed to generate the bounding boxes from the semantic segmentation images. There is potential for reducing the time required for image and bounding box generation in the future. For example, the hardware utilization was not maximized during the process and the source code used to generate bounding boxes may be optimized to improve its runtime performance.

As some of the investigated object detection models expect squared images, all real-world and synthetic images are center cropped to 544x544 pixels which is the largest possible multiple of 32. The input image size being a multiple of 32 is beneficial, e.g., for feature alignment in models with five downsampling operations by a factor of two, such as VGGs [109], ResNets [105], and MobileNetV3-Large [115]. To further reduce computational load during training and evaluation, the images are subsequently downscaled to 320x320 pixels. While this downscaling might also reduce overall detection accuracy, it does not influence the investigation of the research questions as all models are trained and evaluated on the same rescaled images.

3.2 Ruralscapes

The first semantic segmentation use-case is from low-altitude UAS perceptive and involves complex aerial imagery. While the real-world dataset is publicly available, the utilized synthetic dataset was created in a master thesis under the supervision of this work and is therefore presented with greater detail. The dataset and a first description of it were published in [89].

3.2.1 Real-World Data

As the real-world dataset for semantic segmentation from low-altitude UAS perspective, the Ruralscapes dataset [170] is utilized. This dataset comprises high-resolution images of 3840x2160 pixels captured by a tilted camera on a UAS at various altitudes over a rural area in Europe characterized by mountainous terrain. The variation in flight altitude and camera tilt results in objects being depicted across a wide range of scales and perspectives. The dataset consists of 20 videos recorded at 50 FPS, with video sequence lengths ranging from 11 seconds to 2 minutes and 45 seconds, totaling 17 minutes of flight recordings. Fig. 3.11 shows some exemplary images from the dataset.

Semantic segmentation annotations are provided for every 50th frame, resulting in labels for 1 FPS and a total of 1,127 annotated images. The annotated classes are forest, land, hill, sky, residential area, church, road, water, person, car, haystack, and fence. Due to the rural setting and the flight altitude, the dominant classes are residential area, land, and forest, while smaller objects such as haystack, car, and person account for a comparably small percentage. The labeling process was carried out by 21 volunteers, with an average time of 45 minutes per image [170].

To enhance the relevance of the annotated classes for environment perception from a low-altitude UAS and to improve comparability with other public datasets, in this work, the annotated categories are consolidated into the six

Figure 3.11: **Example images from the Ruralscapes dataset [170]**

classes building, car, greenery, person, road, and background. The original road, car, and person classes remain unchanged. The residential area and church annotations are merged to form the building class. The fence, water, and sky classes are combined into the background category, while forest, haystack, and land are merged to create the greenery class. Additionally, the hill class is incorporated into the greenery class, as the areas annotated as hills predominantly feature trees and grass. A detailed summary of the class merging is presented in Table 3.2.

The original paper partitions the annotated video sequences into 13 videos for training and 7 for evaluation [170]. The authors ensured that both subsets are representative of the various flying scenarios and that the scenes are equally distributed between the training and evaluation sets but no identical scenes appear in both subsets. This work uses the same training and evaluation split. Furthermore, a subset of 20% of the training dataset is allocated as the validation dataset, leaving the remaining 80% for training on real-world images in Section 6.2. The resulting real-world training, validation, and evaluation

Table 3.2: **Mapping of the semantic segmentation annotation classes from Ruralscapes to the classes considered in this work**

This Work	Ruralscapes
Building	Church, Residential
Car	Car
Greenery	Forest, Haystack, Hill, Land
Person	Person
Road	Road
Background	Fence, Sky, Water

datasets contain 652, 164, and 311 images, respectively.

3.2.2 Synthetic Data

As no corresponding synthetic dataset from low-altitude UAS perspective existed, a dataset was created in a master thesis under the supervision of this work. The dataset and a first description were published in [89].

The goal was to closely resemble the real-world images to close the content gap to a reasonable extent. However, as it was infeasible to create an exact representation, the approach was to stylistically replicate the flight area in the Unreal Engine [37] and extract images from similar UAS perspectives. Consequently, while the synthetic dataset does not exactly mirror the real-world images and, e.g., building positions may differ, it is designed to be subjectively comparable in terms of visual appearance. As a result, the data generation can be interpreted as following an ODD implicitly given by the real-world dataset.

To create the synthetic dataset, the Unreal Engine 4.27 [37] is utilized again. The steps involved in creating the virtual world are illustrated in Fig. 3.12. The virtual world is built upon an existing environment featuring a mountainous landscape (Fig. 3.12a), which is modified to resemble the real-world dataset. First, a foundation comparable to the Ruralscapes region is created by removing unnecessary objects, leaving only the mountainous terrain with grass and some

(a) **Selection of an existing virtual environment as a basis**

(b) **Basic adjustments and insertion of a road network with cars and people**

(c) **Addition of houses along the road and forests in the background**

(d) **Final adjustments to characteristics such as colors and brightness**

Figure 3.12: **Illustration of the main steps to create the virtual world for the synthetic image dataset corresponding to Ruralscapes. Images first published in [89]**

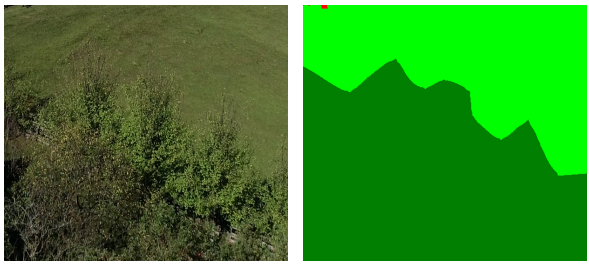
trees. The ground inside the valley is flattened to accommodate the village and the mountains are reduced in size, smoothed, and modified to remove sharp edges and cliffs to better resemble the real-world hills. A small hill with a meadow and a stream is also added to match the real-world setting. Second, a simple road network is added using build-in Unreal Engine tools. Similar to the Ruralscapes dataset, it consists of a main road with lane markings and smaller streets without markings or sidewalks. Along the streets, cars, lanterns, fences, and people are placed (Fig. 3.12b). Third, houses are placed along the streets. To increase variation, these houses have a wide range of shapes, as well as roof and facade textures (Fig. 3.12c). Finally, adjustments are made to improve the

Figure 3.13: **Example images from the generated synthetic dataset**

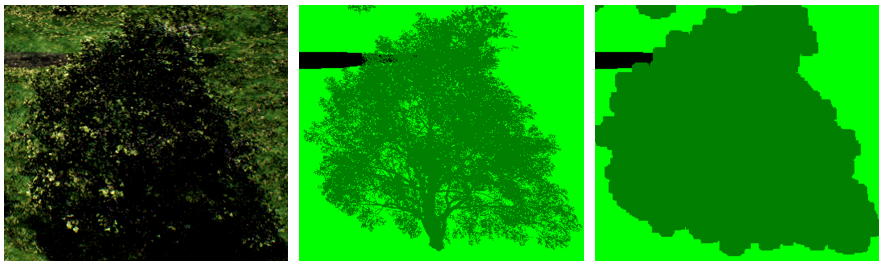
similarity to the real-world dataset, including color adjustments to the greenery and brightness adjustments using the Unreal Engine lighting system. A part of the final virtual world is shown in Fig. 3.12d.

To extract images and annotations from the virtual world, the AirSim plugin [41] is utilized again. Overall, 1,569 synthetic images are extracted for training and 449 for evaluation. Example images are presented in Fig. 3.13. Although there are no exact matching pairs between the synthetic and real-world images, the synthetic images appear visually comparable. However, some differences in photometric characteristics may be observed. While the real-world images exhibit a relatively consistent visual appearance, the synthetic images display more variation. This increased variation does not need to deteriorate the segmentation performance but may even be beneficial for the generalization capability of the semantic segmentation models.

The time required to generate the synthetic dataset was not measured exactly but was much shorter than the time needed to create the synthetic drogue



(a) **Example real-world RGB image (left) with annotations (right) from the Ruralscapes dataset [170]**



(b) **Example RGB image from the virtual world (left) with extracted annotations (middle) and annotations after applying max pooling (right) to improve similarity to the real-world annotations**

Figure 3.14: **Comparison of the level of detail of the annotations for the tree class. Images first published in [89]**

detection dataset, as fewer images were generated and no bounding boxes extraction was necessary, as the semantic segmentation annotation masks can be directly used to train the semantic segmentation models.

When comparing the automatically extracted annotations from the Unreal Engine to the manual annotations of the real-world dataset, a difference in level of detail is observed. This is especially noticeable for the annotations of the trees. As illustrated in Fig. 3.14, the automatic annotations of the synthetic images are pixel-precise, whereas the manual annotations of the real-world dataset are coarser and contain simplifications that seem to be intuitive for

human annotators. In the synthetic annotations, every branch and leaf of the trees are annotated and the background visible through the trees is assigned to its corresponding class. Such a high level of detail would be impractical for human annotators and is unlikely to provide any benefits for the considered use-case. In contrast, the real-world dataset assigns all pixels within the silhouette of the tree to the tree class. As it is expected that this difference in annotation will create incorrect incentives for the semantic segmentation model, and ultimately lead to a worse overall accuracy on the real-world images, the level of detail of the synthetic tree annotations is artificially reduced to better align with the real-world annotations. This is achieved by applying a max pooling filter to the tree annotations, overcasting the areas between the branches and resulting in silhouette-like annotations for the trees.

While the real-world images have a resolution of 3840x2160 pixels, the synthetic images are generated with a resolution of 1920x1080 pixels. To prevent different image sizes from influencing the results, and to reduce computational load during deep learning model training and evaluation, all images are downscaled to 960x540 pixels using linear interpolation. The annotations are downscaled using nearest-neighbor interpolation to preserve the discrete nature of the class labels and to prevent color mixing between different classes.

3.3 KITTI

The second semantic segmentation use-case is from the autonomous driving domain. It is characterized by different perspectives and diverse lighting conditions.

3.3.1 Real-World Data

The real-world dataset for the autonomous driving use-case is derived from the KITTI dataset [59], with annotations for semantic segmentation provided by [171]. The KITTI dataset contains images captured from a camera mounted on top of a car driving in and around Karlsruhe, Germany. It features multiple recording sequences from a diverse range of scenes, including freeway, rural, and inner-city environments, all recorded during daytime in September and October 2011. The raw images have been cropped to exclude the engine hood and sky. While the KITTI dataset is available with annotations for, e.g., object detection, tracking, and optical flow estimation, it lacks annotations for semantic segmentation.

To address this, [172] contributed semantic segmentation annotations for a subset of 146 images. These annotations were later improved in [171]. The 146 images are utilized as the real-world validation dataset in this work. Furthermore, [171] provided annotations for 299 additional images from disjoint recording sequences, serving as the real-world evaluation dataset in this work. The disjoint recording sequences ensure that the validation and evaluation datasets do not share any images which could skew the results.

The resolutions of the available images vary slightly, ranging from 1226 to 1242 pixels in width and from 370 to 376 pixels in height. To ensure consistency, all images are center cropped to the uniform size of 1226x370 pixels, which is the biggest common resolution among the dataset. Example images are presented in Fig. 3.15.

Figure 3.15: **Example images from the KITTI dataset [171]**

3.3.2 Synthetic Data

As the synthetic dataset for the autonomous driving use-case, the Virtual KITTI 2.0 (VKITTI2) dataset [146] is used. This dataset is a successor of the Virtual KITTI dataset [35], containing the same recording sequences but using a newer version of the Unity game engine, leading to more photorealistic images by exploiting improved lighting and post-processing features.

The VKITTI2 dataset is constructed to mimic the original KITTI dataset using a so-called real-to-virtual cloning method. It clones fives scenes from the original dataset which are disjoint from the real-world ones used for evaluation. They display diverse scenery such as a crowded urban area, busy intersections, a road in the forest, and highway driving. Overall, the dataset contains 2,115 images. Example images are presented in Fig. 3.16.

While the VKITTI2 dataset is created using the real-to-virtual cloning method, there are no exact matching pairs between the synthetic and the annotated real-world images available. All images show street scenes and are comparable between the datasets but the real-world images contain considerably more images from urban neighborhoods. Furthermore, the synthetic images, at least partly, exhibit a simpler visual appearance compared to the real-world images, with less color diversity but more uniformly colored areas as exemplified by the grass areas in Fig. 3.16. The synthetic images also do not contain pedestrians

Figure 3.16: **Example images from the VKITTI2 dataset [146]**

and cyclists. Overall, although similar, this synthetic and real-world dataset pair seems to differ the most out of the three considered ones.

As the synthetic evaluation dataset, scene number 2 is utilized because it covers a wide variety of potential situations and contains the most diversity. It shows a road in an urban area as well as a busy intersection and contains 223 images. The remaining 1,892 synthetic images from the other scenes are used for training.

Although the VKITTI2 dataset is rebuilt after KITTI, some deviations in class annotations are present. For instance, the synthetic dataset lacks the pedestrian and cyclist classes. When a model is trained on this dataset, it will be unable to detect these classes. Therefore, such class differences should be resolved to enable a fair evaluation. To address this, some classes are merged in this work, as shown in Table 3.3. The final dataset consists of the classes Building, Ground, Pole/Sign, Sky, Vegetation, Vehicle, and Other. The classes Truck, Van, and Car are merged into a Vehicle class, while Trees are merged into the Vegetation class. Additionally, Traffic Sign, Traffic Light, Sign, and Pole are merged into a Pole/Sign class, and Road, Sidewalk, and Terrain are merged into a Ground class. The classes Ignore, Misc, Fence, and Guard Rail are all merged into Other, and since VKITTI2 does not contain Pedestrian and Cyclist classes, these KITTI classes are also merged into Other.

Table 3.3: **Merging of the semantic segmentation annotation classes from KITTI and VKITTI2 to resolve annotation differences**

Merged	KITTI	VKITTI2
Building	Building	Building
Ground	Road, Sidewalk	Road, Terrain
Pole/Sign	Pole, Sign	Pole, Traffic Light, Traffic Sign
Sky	Sky	Sky
Vegetation	Vegetation	Tree, Vegetation
Vehicle	Car	Car, Truck, Van
Other	Cyclist, Fence, Ignore, Pedestrian	Guard Rail, Misc

For consistency with the real-world images, all synthetic images are center cropped to 1226x370 pixels.

4 Deep Learning Model Influences on the Sim-to-Real Generalization

4.1 Methodology

This section presents the methodology for investigating Research Question 2, which explores the influence of the deep learning model itself on the sim-to-real generalization. The research question addresses a significant gap in the literature, as the impact of the model architecture on the generalization has received only limited attention. To enhance the generalizability of the results and to provide actionable insights for practical applications, multiple deep learning models for different perception tasks are investigated on various datasets.

As presented in Section 2.3, on an abstract level, the considered deep learning models consist of a backbone for feature extraction and a meta-architecture defining how the extracted features are processed to produce the final output for the specific task. The influence of both components is investigated. To improve generalization of the results, object detection and semantic segmentation architectures are considered. To investigate the influence of different backbones, various object detection and semantic segmentation architectures are combined with different backbones, trained and evaluated. To compare the influence of different architectures, various backbones are combined with

different object detection and semantic segmentation architectures, trained and evaluated. The detailed experimental setup is presented in Section 4.2.

The backbones of the investigated object detection models come with initial weights pre-trained on the ImageNet dataset [173] and the semantic segmentation models on the ADE20K dataset [174]. Using pre-trained weights is common practice across both academia and industry to improve training stability and to reduce training time. The pre-training datasets are widely used, generic real-world datasets that serve as a basic initialization but do not mirror the specific distributions targeted during evaluation. Importantly, the weights of the final models are trained exclusively on synthetic images. This setup reflects practical scenarios in which pre-trained models but no task-specific real-world data is available and therefore enhances the practical relevance of the findings.

To ensure that the model variations with the best real-world performance are selected for further investigations, real-world validation datasets are used. As real-world use-cases are ultimately interested in the best real-world performance of a model, this approach has more practical relevance than using synthetic validation datasets. However, it is important to note that the real-world validation datasets do not influence the training itself and that the weights of the final models are trained exclusively using synthetic images.

Two metrics are employed to investigate the sim-to-real generalization and potential differences between the deep learning models. First, all trained models are evaluated on their corresponding real-world evaluation datasets using mAP for object detection (cf. Section 2.1.1) and mIoU for semantic segmentation (cf. Section 2.1.2). These measures are important for practical applications as they directly quantify the final performance on a real-world dataset. Additionally, the sim-to-real gap, as presented in Section 2.2, is used to quantify the performance on real-world images in reference to the performance on synthetic images. This allows for a more in-depth investigation of the sim-to-real generalization of models with lower overall performance. Therefore, all models are also evaluated on synthetic evaluation datasets. The results are presented in Section 4.3. First presentations of the results were published in [90] for object detection and in [91] for semantic segmentation.

Table 4.1: **List of all investigated object detection models and their number of trainable weights**

Architecture	Backbone	Weights
Faster R-CNN	VGG-13	39M
Faster R-CNN	VGG-16	44M
Faster R-CNN	VGG-19	49M
Faster R-CNN	MobileNetV3-Large FPN	19M
Faster R-CNN	ResNet-34 FPN	38M
Faster R-CNN	ResNet-50 FPN	41M
Faster R-CNN	ResNet-101 FPN	60M
Faster R-CNN	WRN-50-2 FPN	85M
Faster R-CNN	ResNeXt-50 (32x4d) FPN	41M
SSD	ResNet-50	25M
RetinaNet	ResNet-50 FPN	32M
FCOS	ResNet-50 FPN	32M

4.2 Experimental Setup

4.2.1 Object Detection

To compare the influence of different backbones on the sim-to-real generalization, a Faster R-CNN object detection architecture combined with various backbones is trained and evaluated. To compare the influence of different architectures, a ResNet-50 backbone is integrated, trained, and evaluated with various object detection architectures. All trained object detection models and their number of trainable weights are presented in Table 4.1. SSD with ResNet-50 is trained without FPN because SSD uses its own multi-level prediction architecture as explained in Section 2.3. All models are used with their implementation from the Torchvision library [103].

Each of the considered models is trained with a range of different optimizers and hyperparameters to minimize potential bias of the results because of unsuitable training settings. Overall, each model is trained twelve times using

the following optimizers and hyperparameters:

- Adam [175] with a learning rate of 0.01;
- Adam with a learning rate of 0.001;
- Adam with a learning rate of 0.0001;
- Adam with a learning rate of 0.00001;
- Stochastic Gradient Descent (SGD) with a learning rate of 0.01;
- SGD with a learning rate of 0.001;
- SGD with a learning rate of 0.0001;
- SGD with a learning rate of 0.00001;
- SGD with a learning rate of 0.01, momentum of 0.9, and weight decay of 0.0005;
- SGD with a learning rate of 0.001, momentum of 0.9, and weight decay of 0.0005;
- SGD with a learning rate of 0.0001, momentum of 0.9, and weight decay of 0.0005;
- SGD with a learning rate of 0.00001, momentum of 0.9, and weight decay of 0.0005.

As a result, each of the 12 models is trained 12 times, resulting in 144 trained variations.

The object detection models are trained on the drogue detection dataset with the splits presented in Section 3.1. Each model is trained for a maximum of 200 epochs with a training batch size of 16. To reduce computation when the model has reached a local optimum and does not improve anymore, early stopping with a patience of 20 is used. The model performing best on the validation dataset is used for further evaluation.

Because of the low variability of the images in the air-to-air refueling situation, some standard image augmentations are applied during training. The utilized augmentation techniques are horizontal flipping, randomly changing brightness

and contrast of the image, applying Gaussian and camera sensor noise, and blurring the image using a randomly sized kernel. The augmentations are applied using the Albumentations library [85]. An isolated investigation of the influence of image augmentations on the sim-to-real generalization is given in Chapter 5.

4.2.2 Semantic Segmentation

To compare the influence of different backbones, an UPerNet semantic segmentation architecture combined with various backbones is trained and evaluated. To compare the influence of different architectures, different backbones are integrated, trained, and evaluated with different semantic segmentation architectures. All trained semantic segmentation models and their number of trainable weights are given in Table 4.2. All models are used with their implementation from the MMSegmentation library [104].

Each of the considered models is trained with a range of different optimizers and hyperparameters to minimize potential bias of the results because of unsuitable training settings. Overall, each model is trained seven times using the following optimizers and hyperparameters:

- Adam [175] with a learning rate of 0.001;
- AdamW [176] with a learning rate of 0.001 and weight decay of 0.01;
- Nadam [177] with a learning rate of 0.002, momentum of 0.9, and momentum decay of 0.004;
- SGD with a learning rate 0.01, momentum of 0.9 and weight decay of 0.0005;
- SGD with a learning rate of 0.01;
- SGD with a learning rate of 0.001;
- SGD with a learning rate of 0.0001.

Table 4.2: **List of all investigated semantic segmentation models and their number of trainable weights**

Architecture	Backbone	Weights
UPerNet	ResNet-50	66M
UPerNet	ResNet-101	85M
UPerNet	ConvNeXt-T	60M
UPerNet	ConvNeXt-S	82M
UPerNet	ConvNeXt-B	122M
UPerNet	Swin-T	60M
UPerNet	Swin-S	81M
UPerNet	Swin-B	121M
UPerNet	Twins-PCPVT-S	54M
UPerNet	Twins-PCPVT-B	74M
UPerNet	Twins-PCPVT-L	91M
UPerNet	Twins-SVT-S	54M
UPerNet	Twins-SVT-B	88M
UPerNet	DeiT-S MLN	58M
UPerNet	DeiT-B MLN	145M
PSPNet	ResNet-50	49M
PSPNet	ResNeSt-101	72M
PSPNet	MobileNetV2	14M
FCN	ResNet-50	49M
FCN	ResNeSt-101	72M
FCN	MobileNetV2	10M
DeepLabV3	ResNet-50	68M
DeepLabV3	ResNeSt-101	91M
DeepLabV3	MobileNetV2	19M
DeepLabV3+	ResNet-50	44M
DeepLabV3+	ResNeSt-101	66M
DeepLabV3+	MobileNetV2	15M

As a result, each of the 27 models is trained 7 times on each of the 2 datasets, resulting in 378 trained variations.

The semantic segmentation models are trained separately on each of the semantic segmentation datasets and their respective splits presented in Chapter 3. Each model is trained for a maximum of 200 epochs with a training batch size of 2. To reduce computational load when the model reaches a local optimum and does not improve anymore, early stopping with a patience of 10 is applied. The model performing best on the validation dataset is used for further evaluation on the evaluation dataset.

4.3 Results

4.3.1 General Influence

Fig. 4.1 shows the performance of all trained object detection models on the drogue detection dataset and Fig. 4.2 shows the performance of all trained semantic segmentation models on the KITTI and the Ruralscapes datasets. The results demonstrate significant differences in sim-to-real generalization among the trained models and variations. The general trend of the experiments is similar across all three datasets: The better a model performs on the synthetic images, the better it performs on the real-world images, as shown by the positive slopes of the linear regression lines. However, the increase in performance on the real-world images is lower than the increase on the synthetic images, i.e., the sim-to-real gap also tends to increase when performance on synthetic images increases. In the figures, the sim-to-real gap as described in Section 2.2 is symbolized by the distance in y-axis direction of a datapoint to the identity line.

Furthermore, the results highlight that the sim-to-real gap may also be negative, i.e., the performance on the real-world images is better than on the synthetic images even though the model was trained on synthetic images. While this

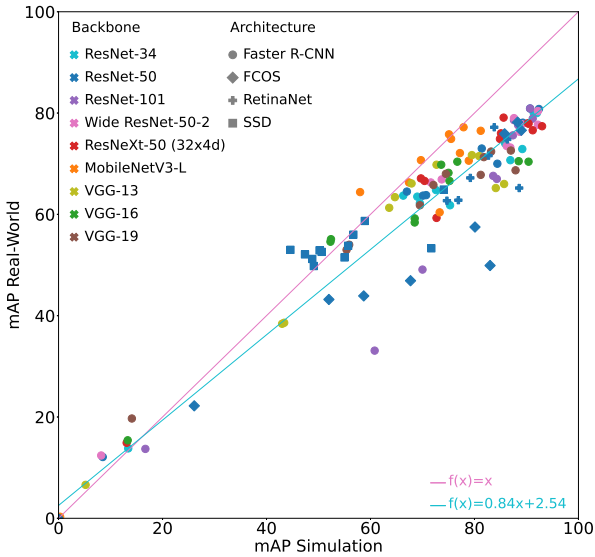
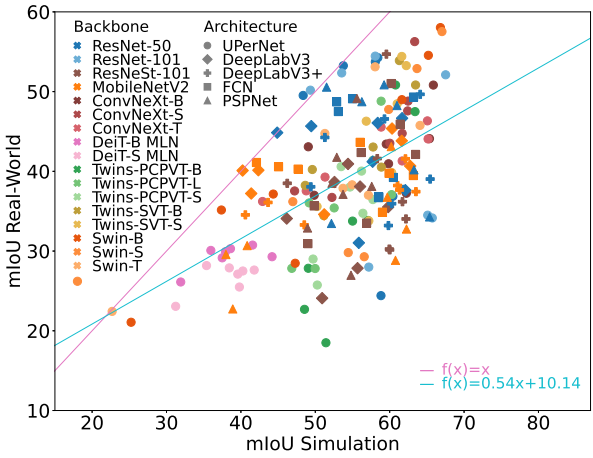


Figure 4.1: **Performance of all trained object detection models on the evaluation datasets. Pink: Identity line. Turquoise: Linear regression through all datapoints. First published in [90]**

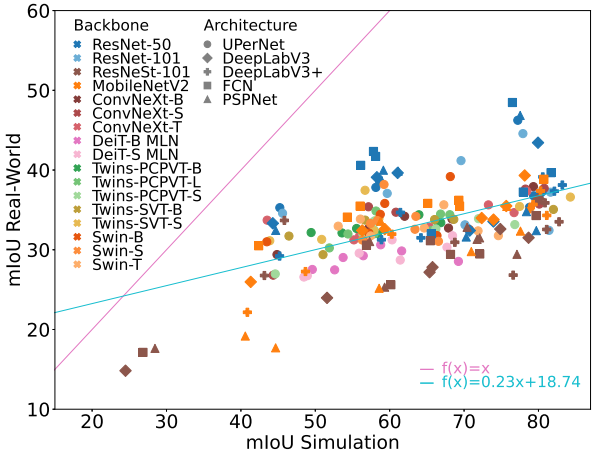
mostly occurs for models with a relatively low real-world performance, which would be irrelevant for practical applications, some models show a negative gap with a relatively high real-world performance. This will further be analyzed and discussed in Section 4.3.3.

Interestingly, model variations with a negative gap occur less often on the two semantic segmentation datasets than on the simple object detection dataset. This is probably rooted in the complexity of the considered datasets. In general, it can be observed that the sim-to-real gap is not only model but also dataset dependent. Therefore, the absolute value of the gap depends on the considered datasets.

When comparing the performance and the generalization of the models on the three datasets, the experiments show that the models have the best overall real-



(a) **KITTI dataset**



(b) **Ruralscapes dataset**

Figure 4.2: **Performance of all trained semantic segmentation models on the evaluation datasets. Pink: Identity line. Turquoise: Linear regression through all datapoints. First published in [91]**

world performance and the smallest sim-to-real gap on the drogue detection dataset. It is followed by the KITTI dataset that shows a smaller linear regression slope as symbolized by the turquoise line and a slightly larger general gap. The datapoints are relatively centered and the sim-to-real gap of the best models is also comparably small. On the Ruralscapes datasets, the sim-to-real gap is, in general, larger and the datapoints are more distributed across the x-axis. Also, the slope of the linear regression through all datapoints, as shown by the turquoise line, is much smaller for the Ruralscapes datasets. There are large deviations in mIoU on the synthetic evaluation dataset but comparably little deviations in mIoU on the real-world one.

For the specific use-cases considered in this work, the larger sim-to-real gap for the Ruralscapes dataset is probably caused by the fact that the synthetic training and evaluation datasets are disjoint but drawn from the same distribution. Because of that, the performance on the synthetic evaluation dataset is high, resulting in a high sim-to-real gap. For the KITTI dataset, however, the synthetic evaluation dataset is from a scene that is not in the training dataset and looks considerably different. As a result, the performance on the synthetic evaluation dataset is lower and, consequently, so is the sim-to-real gap. For the drogue detection dataset, there are no exact image pairs but the data generation process was designed to result in images that are visually similar to the real-world images and cover the whole range of possible drogue position. Furthermore, because of the visually simple situation, there are less general appearance and content differences possible, resulting in a low gap.

The best overall object detection model on the drogue detection dataset in terms of real-world mAP is a Faster R-CNN with either a ResNet-50 or a ResNet-101 backbone, achieving an mAP of 90.7% on the synthetic and an mAP of 80.9% on the real-world evaluation dataset. Because of their same numerical performance, both models are displayed exactly on top of each other in Fig. 4.1. Furthermore, there is a visible separation between the worst performing models and the rest. These low performing models seem to have failed to converge during training or do not produce meaningful results, indicating a failure to learn effective features under the specific hyperparameter configurations. As they are not relevant for practical applications and would typically be filtered out in real-world training scenarios, these models with an mAP of less than

30% on the synthetic evaluation dataset are not displayed in the following figures to improve their visual clarity and focus on the parts relevant for the considered research question.

On the Ruralscapes dataset, the best real-world model also uses a ResNet-50 backbone, integrated into an UPerNet architecture. It achieves an mIoU of 77.2% on the synthetic and an mIoU of 46.2% on the real-world dataset. The model with the best real-world performance on the KITTI dataset is an UPerNet with a Swin-B backbone, achieving an mIoU of 66.8% on the synthetic and an mIoU of 58% on the real-world dataset.

4.3.2 Influence of the Backbone

Object Detection

To investigate the influence of the backbone on the sim-to-real generalization of object detection models, a Faster R-CNN architecture is trained with various backbones. The performance of these models is presented in Fig. 4.3, which demonstrates that the backbone significantly impacts the generalization capability of the models.

In general, the models with a ResNet backbone are able to achieve the best performance on real-world images. In contrast, the models with a VGG backbone have a significantly lower performance. This is especially true for the models with a high performance on the synthetic images. Since their performance on the synthetic images does not deviate that much from that of the ResNet models, they also have a much bigger sim-to-real gap. The MobileNets lie between the ResNets and the VGGs in their performance on real-world images. However, they seem to have the best generalization capability as they often outperform all other models with the same performance on synthetic images in their performance on real-world images. Notably, the MobileNets also have the smallest number of trainable weights of the considered models, whose influence on the generalization is further discussed in Section 4.3.3.

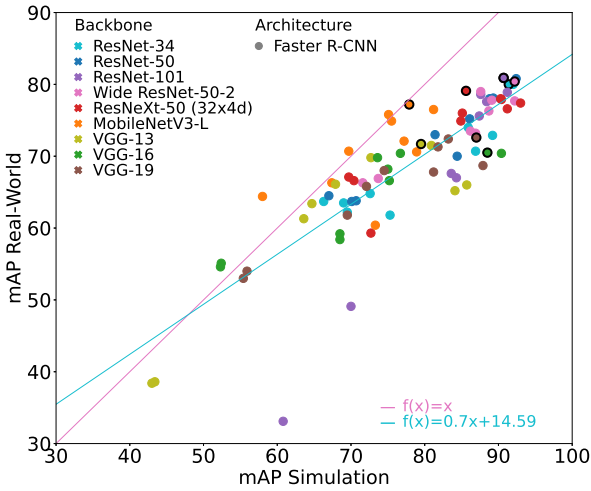


Figure 4.3: **Performance of the Faster R-CNN object detection models with various backbones on the evaluation datasets. Pink: Identity line. Turquoise: Linear regression through all datapoints. Bold edges: Best model variation with respect to real-world mAP for each backbone. First published in [90]**

In Fig. 4.3, the marker for the model with the highest performance on the real-world dataset for each backbone is shown with bold edges. The corresponding values are given in Table 4.3. When looking at these best performing models, it can be seen that the best ResNet models have a similar performance that deviates by only 1.8 pp on the real-world images and by 5.3 pp in sim-to-real gap. When not considering ResNeXt-50 (32x4d), the deviation is reduced even further to just 0.9 pp in performance on the real-world images and 2 pp in sim-to-real gap. With their performance, the ResNets outperform all other models on the real-world images.

The ResNets are followed by the best model variation with a MobileNet backbone in terms of real-world performance. While the model is not able to reach a performance as high as the ResNet-based one by performing 3.7 pp lower on the real-world images, it has a sim-to-real gap of only 0.7 pp. This is by

Table 4.3: **Performance of the Faster R-CNN object detection model variations that perform best on the real-world evaluation dataset for each backbone. Sim-to-real gap (mAP Diff) in percent points, mAP in percent. Best value for each column in bold, worst in italic**

Backbone	Optimizer	Learning Rate	Weight Decay	mAP Sim	mAP Real	mAP Diff
ResNet-101	SGD	1.0e-03	5e-4	90.7	**80.9**	9.8
ResNet-50	Adam	1.0e-04	0	90.7	**80.9**	9.8
Wide ResNet-50-2	Adam	1.0e-04	0	**92.2**	80.4	11.8
ResNet-34	Adam	1.0e-04	0	91.4	80.0	11.4
ResNeXt-50 (32x4d)	SGD	1.0e-02	5e-4	85.6	79.1	6.5
MobileNetV3-L	Adam	1.0e-05	0	*77.9*	77.2	**0.7**
VGG-19	Adam	1.0e-04	0	87.0	72.6	14.4
VGG-13	SGD	1.0e-02	0	79.5	71.7	7.8
VGG-16	Adam	1.0e-05	0	88.5	*70.5*	*18.0*

far the lowest sim-to-real gap of the best performing models. Notably, the MobileNet has by far the smallest number of trainable weights compared to the other investigated models. At the same time, however, this may also be the reason why the model has the worst performance on the synthetic images as the smaller number of trainable weights usually leads to a smaller model capacity.

The worst performances on the real-world images come from the models with a VGG backbone. Their performance on the real-world images is 8.3 to 10.4 pp worse than the performance of the best performing model. The models with the VGG-16 and VGG-19 backbone also have the highest sim-to-real gaps with values of 18.0 pp and 14.4 pp, respectively. It should be noted that VGG models are also the oldest backbones considered in this work with many improved networks building on them as presented in Section 2.3. Only the model with the smallest VGG-13 backbone has a sim-to-real gap comparable to the ResNets. This model size influence is further discussed in Section 4.3.3.

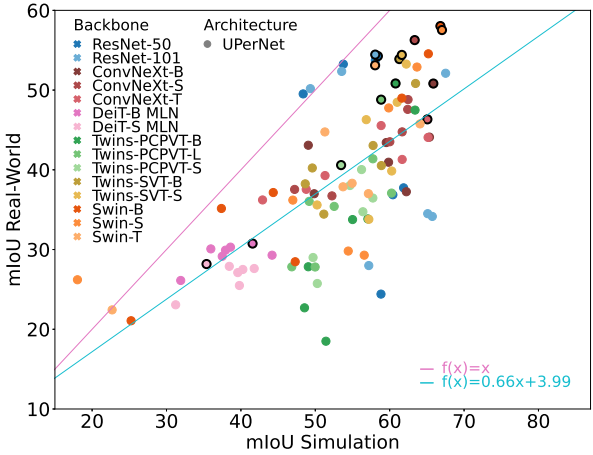
Semantic Segmentation

To investigate the influence of the backbone on the sim-to-real generalization of semantic segmentation models, an UPerNet architecture is trained with various backbones. The performance of these trained models is presented in Fig. 4.4. Consistent with the results for the object detection models, the backbone has a significant influence on the generalization capability of the models on both datasets.

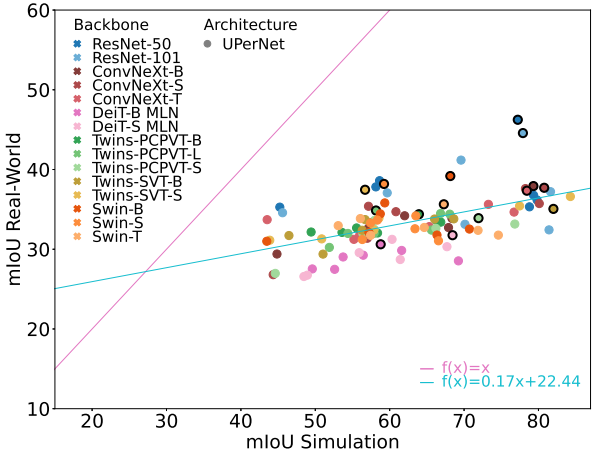
Among all trained models, those with a ResNet backbone seem to regularly have a strong real-world performance and a smaller sim-to-real gap than the other models as they frequently outperform the models with a similar performance on synthetic images in their performance on real-world images. Although they do not achieve an overall real-world performance as high as, e.g., Swin-based models on the KITTI dataset, they stand out on the Ruralscapes dataset. This trend is also confirmed again in Section 4.3.5.

When investigating the transformer models, the Swin transformer often yields the best results of the considered transformers on both real-world evaluation datasets. While the performance of most model variations on the real-world datasets is spread, the performance of the DeiT transformers on the synthetic as well as the real-world data is usually relatively low. Also, its generalization capability is relatively low, especially on the Ruralscapes data where the DeiT models often perform worse on the real-world data than all other models with a similar performance on the synthetic data. As explained in Section 2.3, DeiT was one of the first vision transformers and was further build upon and improved by later transformer models, as can be seen in the results.

In Fig. 4.4, the marker for the model variation with the best result on the real-world dataset for each backbone is shown with bold edges. The corresponding values are given in Table 4.4. They again underline that the backbone has a significant influence on the sim-to-real gap. It varies between 3.6 and 18.7 pp on the KITTI dataset and between 19.2 and 47.0 pp on the Ruralscapes dataset.



(a) **KITTI dataset**



(b) **Ruralscapes dataset**

Figure 4.4: **Performance of the UPerNet semantic segmentation models with various backbones on the evaluation datasets. Pink: Identity line. Turquoise: Linear regression through all datapoints. Bold edges: Best model variation with respect to real-world mIoU for each backbone. First published in [91]**

When investigating these best models in terms of absolute performance, the Swin transformer gives the highest mIoU on the real-world KITTI dataset, but the models with the CNN-based backbones, ResNet and ConvNeXt, have only a slightly lower performance. Especially when comparing Swin-S and ConvNeXt-S, which have a similar number of trainable weights, their performance on the real-world KITTI dataset deviates by only 1.2 pp. On the Ruralscapes real-world evaluation dataset, the CNN-based ResNet-50 backbone gives the best results and outperforms modern state-of-the-art transformers like Swin by 7 pp. The second-best model is also from the ResNet family.

In general, this demonstrates that there are no clear indications for the hypoth-

Table 4.4: **Performance of the UPerNet semantic segmentation model variations that perform best on the real-world evaluation dataset for each backbone. Sim-to-real gap (mIoU Diff) in percent points, mIoU in percent. Best value for each column in bold, worst in italic**

(a) **KITTI dataset**

Backbone	Optimizer	Learning Rate	Weight Decay	mIoU Sim	mIoU Real	mIoU Diff
Swin-B	SGD	1e-2	5e-4	66.8	**58.0**	8.8
Swin-S	SGD	1e-2	5e-4	**67.0**	57.5	9.5
ConvNeXt-S	SGD	1e-2	5e-4	63.4	56.3	7.1
ResNet-101	SGD	1e-3	0	58.1	54.5	**3.6**
Tw-SVT-S	SGD	1e-2	0	61.6	54.4	7.2
ResNet-50	SGD	1e-2	5e-4	58.4	54.3	4.1
Tw-SVT-B	SGD	1e-2	5e-4	61.3	53.9	7.4
Swin-T	SGD	1e-2	0	58.0	53.1	4.9
Tw-PCPVT-B	SGD	1e-2	5e-4	60.8	50.8	10.0
ConvNeXt-B	Adam	1e-3	0	65.9	50.8	15.1
Tw-PCPVT-L	SGD	1e-2	5e-4	58.9	48.8	10.1
ConvNeXt-T	Adam	1e-3	0	65.1	46.3	*18.7*
Tw-PCPVT-S	SGD	1e-2	0	53.5	40.6	12.9
DeiT-B MLN	AdamW	1e-3	1e-2	41.6	30.8	10.8
DeiT-S MLN	SGD	1e-4	0	*35.4*	*28.2*	7.2

(b) **Ruralscapes dataset**

Backbone	Optimizer	Learning Rate	Weight Decay	mIoU Sim	mIoU Real	mIoU Diff
ResNet-50	SGD	1e-2	5e-4	77.2	**46.2**	31.0
ResNet-101	SGD	1e-2	5e-4	77.9	44.6	33.3
Swin-B	SGD	1e-2	5e-4	68.2	39.2	29.0
Swin-S	SGD	1e-2	5e-4	59.2	38.2	21.1
ConvNeXt-B	Adam	1e-3	0	79.3	37.9	41.4
ConvNeXt-S	AdamW	1e-3	1e-2	80.8	37.7	43.1
Tw-SVT-S	SGD	1e-2	5e-4	*56.7*	37.5	**19.2**
ConvNeXt-T	AdamW	1e-3	1e-2	78.4	37.3	41.1
Swin-T	SGD	1e-2	5e-4	67.3	35.6	31.6
Tw-SVT-B	AdamW	1e-3	1e-2	**82.0**	35.1	*47.0*
Tw-PCPVT-L	SGD	1e-2	0	58.2	34.9	23.3
Tw-PCPVT-B	SGD	1e-2	0	63.9	34.4	29.5
Tw-PCPVT-S	Adam	1e-3	0	71.9	33.9	38.0
DeiT-S MLN	Nadam	2e-3	0	68.5	31.8	36.7
DeiT-B MLN	Adam	1e-3	0	58.8	*30.6*	28.2

esis that transformer-based models outperform CNN-based ones in sim-to-real generalization despite their observed shape-bias combined with the realistic representation of shape in game engines as discussed in Section 1.2. A potential explanation could be rooted in the relatively small size of the synthetic datasets as transformers usually have a strong dependence on large datasets and struggle to outperform CNNs on smaller ones, possibly because of their inherent lack of inductive bias [178, 179].

Overall, the experiments demonstrate that the backbone significantly influences the sim-to-real generalization of semantic segmentation models. Furthermore, there is no clear indication that newer models or transformer architectures substantially improve in their sim-to-real capability compared to traditional CNN-based backbones like ResNet.

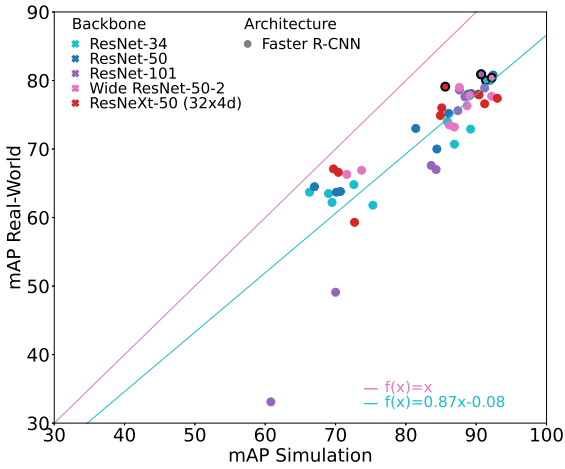
4.3.3 Influence of the Number of Trainable Weights

Object Detection

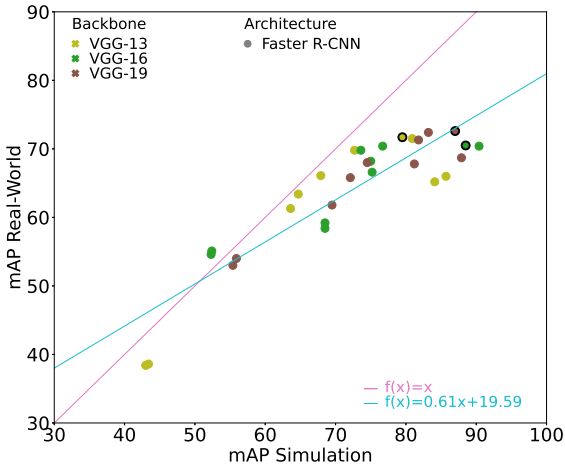
The various backbones of the ResNet family enable an investigation of the influence of the number of trainable weights and their structural organization on their sim-to-real generalization. Fig. 4.5a presents the performance of the different ResNet backbones. It displays a subset of Fig. 4.3 for a less cluttered visualization. The ResNet models exhibit a similar, linear-like behavior in terms of real-world performance and sim-to-real gap, with only a few exceptions for some models with seemingly suboptimal training hyperparameters. Especially the performance of the best models as highlighted by the bold edges is very similar and deviates only marginally on the real-world images.

These results demonstrate that increasing the number of trainable weights does not necessarily lead to a better sim-to-real generalization for the ResNet models. For example, the increase of trainable weights in the Wide ResNet-50-2 backbone by widening the layers does not yield significant improvements in real-world performance or sim-to-real gap. While the best performing model with a Wide ResNet-50-2 achieves the highest performance on the synthetic images, its performance on the real-world images does not improve compared to ResNet-50.

Similarly, increasing the number of layers, as seen in ResNet-34, -50, and -101, also does not lead to significant improvements. The contrary seems to be the case as ResNet-101 even seems to be more sensitive to training hyperparameters resulting in some models with a low real-world performance and a large sim-to-real gap as seen in the bottom half of Fig. 4.5a. Notably, the models with ResNet-50 and ResNet-101 backbones often exhibit similar performance despite ResNet-101 having more than twice the number of layers and more than 20,000 additional trainable weights. The residual connections enable the training of this deep network but also allow the gradient to skip multiple layers during training. It may therefore be possible that the capabilities of the ResNet-101 are either not needed or not yet fully exploited although care was



(a) **ResNet backbones**



(b) **VGG backbones**

Figure 4.5: **Performance of the Faster R-CNN object detection models with ResNet and VGG backbones on the evaluation datasets. Pink: Identity line. Turquoise: Linear regression through all datapoints. Bold edges: Best model variation with respect to real-world mAP for each backbone. First published in [90]**

taken to minimize the latter issue by utilizing a diverse set of hyperparameters. Nevertheless, this is in line with the common problem that deeper machine learning models are usually harder to train than shallow ones.

The introduced cardinality dimension of the ResNeXt model also does not show clear signs of improving the sim-to-real generalization. Most of the trained models are in proximity of the other models with ResNet backbones. While its sim-to-real gap is the lowest of the best performing models with ResNet backbones in Table 4.3, its overall performance on the real-world images is slightly worse.

The VGG backbones allow further investigations into the influence of the backbone depth on the sim-to-real generalization, as it is the main characteristic they differ in. Fig. 4.5b presents the performance of the different models with VGG backbones. Again, it displays a subset of Fig. 4.3 for a less cluttered visualization. Unlike the relatively linear slope of the ResNet models, the slope of the models with VGG backbones seems to be closer to a logarithmic function, where the sim-to-real gap gets much worse when performance on the synthetic images increases. The slope of the linear regression is also much smaller with 0.61 compared to 0.87 as shown in Fig. 4.5.

Comparing VGG models with a similar performance on synthetic images reveals that VGG-13 usually exhibits better real-world performance than VGG-16 and VGG-19, i.e., that it has a lower sim-to-real gap. Interestingly, the best models with VGG-13 and VGG-19 backbones differ only slightly in real-world performance (0.9 pp) but significantly in performance on synthetic images (7.5 pp), resulting in a larger sim-to-real gap, as shown in Table 4.3. This suggests that the VGG-13 backbone, with its fewer trainable weights, may learn more robust representations that are less prone to overfitting with respect to sim-to-real generalization. However, increasing the number of layers, and consequently trainable weights, further from VGG-16 to VGG-19 does not exhibit further degradation.

The performance of the MobileNet models further underscores that larger models are not necessarily better at generalizing from synthetic to real-world data. The MobileNets have by far the smallest number of trainable weights

but often a very small sim-to-real gap. At the same time, they have only a slightly lower real-world performance than the other models. This may be the downside of the smaller number of trainable weights which usually results in a lower model capacity. For some MobileNet variations, the sim-to-real gap is even negative while having a real-world performance higher than the best VGG models.

Semantic Segmentation

The findings from the object detection models, which demonstrated that deeper ResNets are not necessarily better at generalizing from synthetic to real-world images, can be confirmed for the investigated semantic segmentation models. While the best ResNet-50 and ResNet-101 variations exhibit an almost identical performance that deviates by only 0.2 pp on the KITTI real-world evaluation dataset, the best ResNet-50 variation outperforms the best ResNet-101 variation by 1.6 pp on the Ruralscapes real-world evaluation dataset as presented in Table 4.4.

In contrast, model size appears to play a significant role in the capability of Swin transformers to generalize to real-world images. The largest Swin model, Swin-B, performs best on both real-world evaluation datasets, followed by the second-largest model, Swin-S, which performs relatively similarly but slightly worse. The smallest model, Swin-T, lags behind on the real-world evaluation datasets by multiple percentage points.

For the other models, the influence of the model size seems quite mixed. ConvNeXt and Twins-PCPVT do not exhibit a clear advantage between their two largest models, but their smallest variants perform worst on the real-world images. This is particularly evident on KITTI, where a noticeable gap exists (56.3 and 50.8 compared to 46.3 for ConvNeXt, and 50.8 and 48.8 compared to 40.6 for Twins-PCPVT). However, for Twins-SVT, the smaller model variant outperforms its larger counterpart on real-world images, albeit by only a small margin of 0.5 pp on KITTI. For DeiT, both models variants perform poorly compared to other models on both datasets.

Overall, the findings from the object detection models that larger models do not necessarily possess a better sim-to-real generalization capability can be confirmed for ResNet backbones. However, the conclusion cannot be generalized to all models, as for the task of semantic segmentation, Swin transformers demonstrate improved sim-to-real generalization with increasing model size. Furthermore, many smaller model variants exhibit worse generalization performance compared to their larger counterparts. A possible explanation for this discrepancy is that the given semantic segmentation tasks are much more complex compared to the detection of a drogue in front of a simple background and may require models with greater capacity to achieve good performance. It may be the case that there exists a specific capacity threshold above which the models will tend to overfit again, leading to worse performance on the real-world images or a larger sim-to-real gap. However, this could not be shown in these experiments.

4.3.4 Influence of the Optimizer

Although not the primary focus of the investigation, and not part of the model itself but of the training process, the choice of hyperparameters allows a brief examination of the influence of the optimizer utilized during training on the sim-to-real generalization.

The results indicate that Adam and its variants are not bad for sim-to-real generalization, as most of the best-performing object detection model variations on the real-world evaluation dataset employ Adam as the optimizer, as shown in Table 4.3. However, it is noteworthy that the three models trained using SGD exhibit a sim-to-real gap of less than 10 pp. For the semantic segmentation model variations in Table 4.4, the nine best-performing models on the KITTI dataset and the four best-performing models on the Ruralscapes dataset all utilize SGD as the optimizer. With respect to sim-to-real gap, the eleven models with the lowest sim-to-real gap on KITTI and the three models with the lowest on Ruralscapes also use SGD. Furthermore, the models with the two largest sim-to-real gaps on KITTI and the four largest on Ruralscapes employ Adam or AdamW, suggesting potential overfitting to the synthetic data. These findings

seem to be further supported by the literature, which indicates that Adam can lead to increased overfitting compared to SGD [176, 180].

In conclusion, while the Adam-based optimizers do not necessarily result in models with low real-world performance, SGD appears to produce models with lower sim-to-real gaps and strong real-world performance. Therefore, SGD seems to be the preferred choice among the evaluated optimizers for training models with the goal of sim-to-real generalization.

4.3.5 Influence of the Architecture

Object Detection

To investigate the influence of the object detection architecture on the sim-to-real generalization, various architectures with a ResNet-50 backbone are trained and evaluated according to the setup presented in Section 4.2. The performance of the trained variations is shown in Fig. 4.6.

The experiments demonstrate that SSD with ResNet-50 exhibits relatively low performance on both synthetic and real-world images. However, the sim-to-real gap is also relatively small for most of these models. Following the general trend, the gap significantly increases again when investigating the best-performing SSD model. FCOS with ResNet-50 appears to be sensitive to training hyperparameters, resulting in comparably low performance on both synthetic and real-world images for some hyperparameter settings. In contrast to SSD models, the sim-to-real gap is significantly higher for these models. Nevertheless, when training hyperparameters are well-suited, FCOS models achieve good performance on synthetic images with a sim-to-real gap comparable to the best models. Faster R-CNN and RetinaNet exhibit similar behavior, with Faster R-CNN typically showing slightly higher performance on the real-world dataset. Their sim-to-real gap follows the general trend.

As a result, in scenarios where it is not possible to compare multiple training

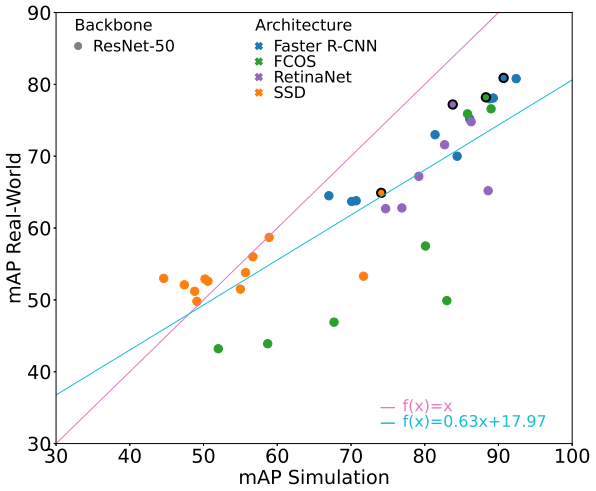


Figure 4.6: **Performance of the object detection architectures with ResNet-50 backbone on the evaluation datasets. Pink: Identity line. Turquoise: Linear regression through all datapoints. Bold edges: Best model variation with respect to real-world mIoU for each architecture backbone combination**

hyperparameters, FCOS may not be the recommended object detection architecture. In such cases, Faster R-CNN appears to be a better choice, as it generally yields models with relatively high real-world performance and a low sim-to-real gap.

In Fig. 4.6, the marker for the model variation with the best result on the real-world evaluation dataset for each architecture is shown with bold edges. The corresponding values are given in Table 4.5. They highlight that the architecture has an influence on the overall performance of the model with a deviation of more than 15 pp. However, this large bandwidth is mainly caused by the SSD architecture. When excluding it, the deviation of the other three architectures is only 3.7 pp. The sim-to-real gap is also relatively consistent across these models, varying only between 6.6 and 10.1 pp, resulting in a deviation of 3.5 pp. This deviation is much smaller than the deviation observed for the

Table 4.5: **Performance of the object detection model variations with ResNet-50 backbone that perform best on the real-world evaluation dataset for each architecture. Sim-to-real gap (mAP Diff) in percent points, mAP in percent. Best value for each column in bold, worst in italic**

Architecture	Optimizer	Learning Rate	Weight Decay	mAP Sim	mAP Real	mAP Diff
Faster R-CNN	Adam	1.0e-04	0	**90.7**	**80.9**	9.8
FCOS	Adam	1.0e-04	0	88.3	78.2	*10.1*
RetinaNet	Adam	1.0e-05	0	83.8	77.2	**6.6**
SSD	Adam	1.0e-04	0	*74.1*	*64.9*	9.2

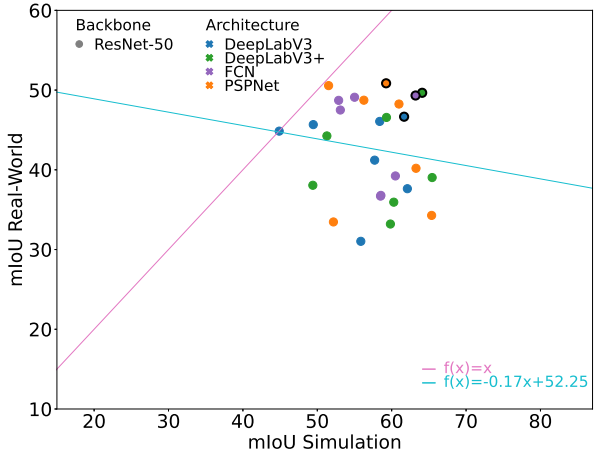
different backbones in Table 4.3 of 17.3 pp (0.7 to 18.0 pp).

In conclusion, the object detection architecture has an impact on the overall real-world performance of the object detection model, although the difference is relatively small for most architectures. While the object detection architecture influences the sim-to-real gap when training hyperparameters are not optimal, its impact appears to be limited when hyperparameters are chosen appropriately. Consequently, the object detection backbone generally has a much larger effect on the sim-to-real generalization.

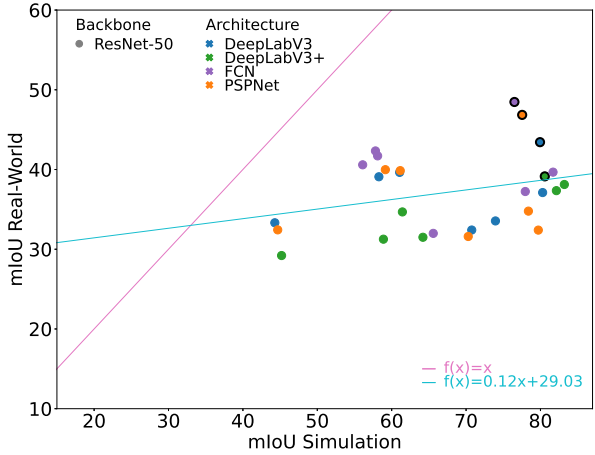
Additionally, while two-stage detection architectures often outperform one-stage detection architectures, as also observed in these experiments, the two-stage Faster R-CNN does not demonstrate clear advantages in terms of sim-to-real generalization.

Semantic Segmentation

To investigate the influence of the semantic segmentation architecture on the sim-to-real generalization, various architectures with the same backbone are trained and evaluated according to the setup presented in Section 4.2. To further improve generalization and achieve a broader investigation, the

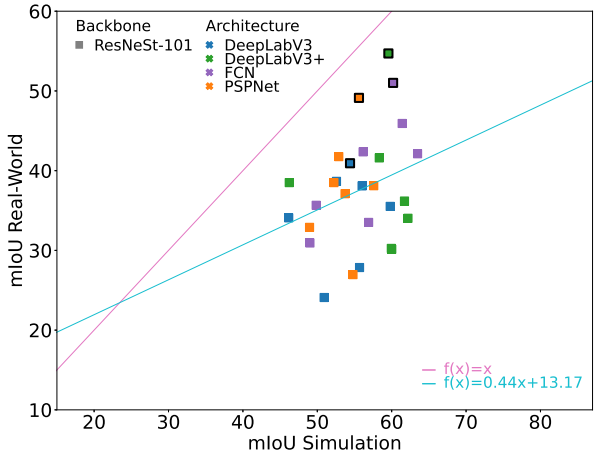


(a) **ResNet-50 backbone on KITTI dataset**

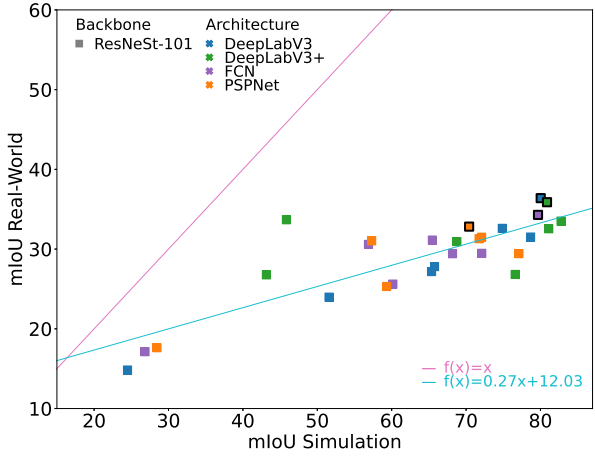


(b) **ResNet-50 backbone on Ruralscapes dataset**

Figure 4.7: **Performance of the semantic segmentation architectures with various backbones on the evaluation datasets. Pink: Identity line. Turquoise: Linear regression through all datapoints. Bold edges: Best model variation with respect to real-world mIoU for each architecture backbone combination**

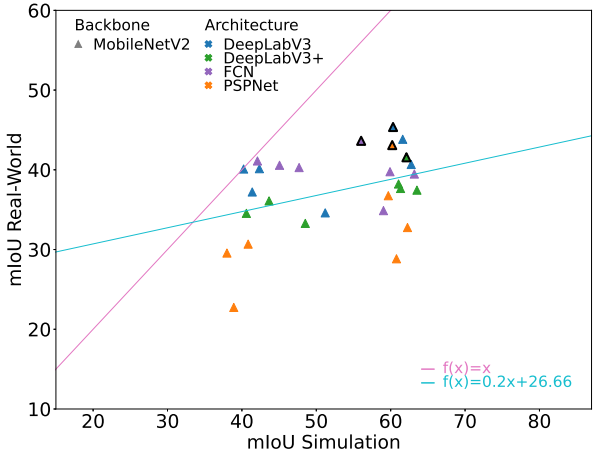


(c) **ResNeSt-101 backbone on KITTI dataset**

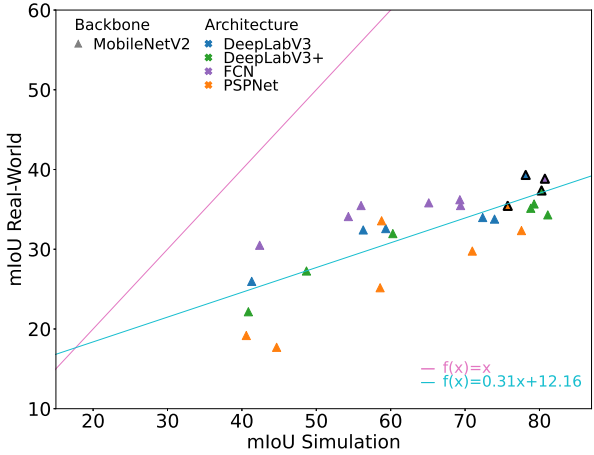


(d) **ResNeSt-101 backbone on Ruralscapes dataset**

Figure 4.7: **(Cont.) Performance of the semantic segmentation architectures with various backbones on the evaluation datasets. Pink: Identity line. Turquoise: Linear regression through all datapoints. Bold edges: Best model variation with respect to real-world mIoU for each architecture backbone combination**



(e) **MobileNetV2 backbone on KITTI dataset**



(f) **MobileNetV2 backbone on Ruralscapes dataset**

Figure 4.7: **(Cont.) Performance of the semantic segmentation architectures with various backbones on the evaluation datasets. Pink: Identity line. Turquoise: Linear regression through all datapoints. Bold edges: Best model variation with respect to real-world mIoU for each architecture backbone combination**

considered architectures are also evaluated with various backbones as shown in Table 4.2. The performance of all trained variations is shown in Fig. 4.7.

The experiments show no clear trends emerging across all architectures and datasets, confirming the observation from the object detection models that the architecture has a relatively minor influence on sim-to-real generalization compared to the backbone. However, some architecture-backbone combinations appear to perform slightly worse than others, although the effect is marginal. For instance, PSPNet-MobileNetV2 tends to be at the lower end of real-world performance, while FCN-MobileNetV2 is more at the upper end. Nevertheless, these influences are significantly smaller than those observed for different backbones.

In Fig. 4.7, the marker for the model variation with the best result on the real-world datasets for each architecture-backbone combination is shown with bold edges. The corresponding values are given in Table 4.6. Similar to the object detection architectures, the variation of the mIoU on the real-world images is much smaller when using different architectures compared to using different backbones. Except from the DeepLabV3 with ResNeSt-101 on KITTI and the DeepLabV3+ with ResNet-50 on Ruralscapes, the real-world mIoU deviates by less than 5.6 pp, which is less than for the different backbones in Table 4.4. Even when including the two outliers, the deviation is smaller.

The sim-to-real gap varies by 6.6 pp for ResNet-50, 8.1 pp for MobileNetV2, and 8.6 pp for ResNeSt-101 on the KITTI dataset. On the Ruralscapes dataset, it varies by 13.4 pp for ResNet-50, by 4.1 pp for MobileNetV2, and by 7.8 pp for ResNeSt-101. Compared to the range of sim-to-real gap for the different backbones used in the UPerNet architecture in Table 4.4, where the gap varied by 15.1 pp on the KITTI dataset and by 27.8 pp on the Ruralscapes dataset, the range of the gap is also much smaller.

Overall, the findings for the object detection models, that the architecture has a smaller influence on the sim-to-real generalization than the backbone, can be generalized to semantic segmentation. For both tasks, some architectures may occasionally lead to a worse overall performance than others. Therefore, selecting a suitable architecture for the task at hand should not be overlooked.

Table 4.6: **Performance of the semantic segmentation model variations that perform best on the real-world evaluation dataset for each architecture backbone combination. Sim-to-real gap (mIoU Diff) in percent points, mIoU in percent. Best value for each column in bold, worst in italic**

(a) **KITTI dataset**

	Architecture	Optimizer	Learning Rate	Weight Decay	mIoU Sim	mIoU Real	mIoU Diff
ResNet-50	PSPNet	SGD	1e-2	0	*59.2*	**50.9**	**8.4**
	DeepLabV3+	SGD	1e-2	5e-4	**64.1**	49.7	14.4
	FCN	SGD	1e-2	5e-4	63.2	49.3	13.9
	DeepLabV3	SGD	1e-2	5e-4	61.7	*46.7*	*15.0*
MobileNetV2	DeepLabV3	Nadam	2e-3	0	60.3	**45.4**	15.0
	FCN	SGD	1e-2	5e-4	*56.0*	43.6	**12.4**
	PSPNet	Nadam	2e-3	0	60.2	43.1	17.1
	DeepLabV3+	Adam	1e-3	0	**62.1**	*41.6*	*20.5*
ResNeSt-101	DeepLabV3+	SGD	1e-2	0	59.6	**54.7**	**4.9**
	FCN	SGD	1e-2	0	**60.2**	51.0	9.2
	PSPNet	SGD	1e-2	0	55.6	49.1	6.5
	DeepLabV3	AdamW	1e-3	1e-2	*54.4*	*40.9*	*13.5*

(b) **Ruralscapes dataset**

	Architecture	Optimizer	Learning Rate	Weight Decay	mIoU Sim	mIoU Real	mIoU Diff
ResNet-50	FCN	SGD	1e-2	5e-4	*76.5*	**48.5**	**28.0**
	PSPNet	SGD	1e-2	5e-4	77.5	46.9	30.7
	DeepLabV3	SGD	1e-2	5e-4	79.9	43.4	36.5
	DeepLabV3+	AdamW	1e-3	1e-2	**80.6**	*39.1*	*41.4*
MobileNetV2	DeepLabV3	AdamW	1e-3	1e-2	78.2	**39.3**	**38.8**
	FCN	Adam	1e-3	0	**80.7**	38.9	41.9
	DeepLabV3+	SGD	1e-2	5e-4	80.3	37.4	*42.9*
	PSPNet	Nadam	2e-3	0	*75.7*	*35.4*	40.3
ResNeSt-101	DeepLabV3	Adam	1e-3	0	80.0	**36.4**	43.6
	DeepLabV3+	AdamW	1e-3	1e-2	**80.9**	35.9	45.0
	FCN	AdamW	1e-3	1e-2	79.7	34.3	*45.4*
	PSPNet	AdamW	1e-3	1e-2	*70.4*	*32.8*	**37.6**

However, its influence on the generalization is much smaller and can mostly be neglected compared to the influence of the backbone.

Notably, as already found in Section 4.3.2, Fig. 4.7 appears to confirm again that ResNets are effective at generalizing to real-world images, particularly on the Ruralscapes dataset, where two smaller clusters are visible above the general trend.

4.4 Summary

This chapter addresses Research Question 2 from Section 1.3 and reveals that deep learning models differ significantly in their ability to generalize from synthetic to real-world images. The general trend is that the better a model performs on the synthetic images, the better it performs on the real-world images. However, the sim-to-real gap also increases, as the performance improvement on the real-world images is lower than that on the synthetic images. Although seldom, the sim-to-real gap can also be negative, i.e., a model performs better on real-world images than on synthetic images, even though it was trained on synthetic data.

The backbone is found to have a significant influence on the sim-to-real generalization of deep learning models. In contrast, while the architecture has influence on the overall real-world performance, its influence on the generalization is much smaller and can mostly be neglected compared to the backbone. When considering the model variations with the most suitable training hyperparameters in terms of real-world performance, the sim-to-real gap differed across various backbones by 17.3 pp for the drogue detection models, and by 15.1 pp and 27.8 pp for the automotive and low-altitude UAS semantic segmentation models, respectively. In contrast, across various architectures, the sim-to-real gap only differed by 3.5 pp, 8.6 pp, and 13.4 pp, respectively.

Although many new deep learning models are developed each year, the experi-

ments demonstrate that they do not necessarily improve in terms of sim-to-real generalization capability. In the object detection and the UAS semantic segmentation use-cases, models with a ResNet backbone yield the best results on the real-world evaluation datasets. In the autonomous driving semantic segmentation use-case, ResNet-based models achieve the second-best results. Notably, while ResNets were already developed in 2015. In general, ResNets seem to have strong generalization capabilities.

Furthermore, there is no clear indication that transformer backbones substantially improve overall sim-to-real generalization compared to CNN-based backbones like ResNet. Although transformers seem promising due to their shape-bias [70, 71] and the realistic representation of shape in game engines [30], the experiments do not confirm this assumption unconditionally. However, transformers do show the best overall performance on the real-world evaluation datasets in certain scenarios, such as the autonomous driving dataset and in [62]. As explained in Section 4.3.3, transformers should be considered for training on synthetic images, especially for large training datasets.

Regarding model size, the results suggest that larger models are not necessarily better at generalizing from synthetic images to real-world ones for visually simple tasks, such as drogue detection in front of a simple background. For this task, the model with the fewest weights, a MobileNetV3 [115], often exhibits the smallest sim-to-real gap and only slightly lower real-world performance than the best models. It can be assumed that this result translates to tasks with a similar visual simplicity, such as object detection on a conveyor. However, more complex tasks, such as the considered semantic segmentation use-cases, may require larger models as the performance of the Swin transformer declines on real-world semantic segmentation images when smaller model variants are used. Furthermore, the smaller variants of many of the other considered models are outperformed by larger ones. It is argued that these tasks require models with a greater capacity to achieve good performance and that a threshold may exist above which the models will tend to overfit again, leading to worse performance on the real-world images or a larger sim-to-real gap.

Although not the primary focus of the investigation, the SGD optimizer appears to be a preferable choice during training, often resulting in models with a

smaller sim-to-real gap and strong real-world performance in the conducted experiments.

Overall, this chapter provides a novel perspective on the sim-to-real generalization of deep learning models. It fills a significant gap in the literature, offers practical insights for deep learning model selection and training on synthetic data, and opens up new avenues for future research.

5 Image Augmentation Influences on the Sim-to-Real Generalization

5.1 Methodology

This section presents the methodology for investigating Research Question 3, which explores the potential of image augmentation techniques to improve the sim-to-real generalization of deep learning models. The influence of various image augmentation techniques is studied on multiple deep learning models, perception tasks, and datasets. This broad investigation aims to increase the generalizability of the results to other domains and to provide actionable insights for practical applications.

Previous research has demonstrated that basic image augmentations can improve performance when using real-world datasets [85, 86] as well as enhance robustness to real-world distribution shifts [87]. However, the extent to which image augmentations can reduce the effects of a shift from synthetic to real-world images, and which augmentations are most beneficial for sim-to-real generalization, remains unknown. Investigating the most beneficial augmentations can also provide insights into the shortcomings of synthetic images and the differences to real-world images that are relevant for deep learning models. Using image augmentations has the advantage that current state-of-the-art visual simulation environments can be used and that only image

post-processing techniques have to be applied during the training of the model. Nevertheless, such insights might also benefit the future development of better visual simulation environments.

The following approach is used to investigate the influence of different image augmentation techniques on the sim-to-real generalization of deep learning models. First, baseline models are trained on the synthetic training datasets without the use of image augmentations. Afterwards, the models are trained on the same synthetic training datasets but in combination with various image augmentation techniques. After training, the models are evaluated on the real-world evaluation datasets corresponding to the synthetic training data. As this research investigates the generalization to real-world images, real-world validation datasets are used to make sure that the model that performs best on real-world images is selected for evaluation. As real-world use-cases are ultimately interested in the best real-world performance of a model, this approach has more practical relevance than using synthetic validation datasets. However, it is important to note that the real-world validation datasets do not influence the training itself and that the weights of the final models are trained exclusively using synthetic images. All models are trained three times and the best-performing training run is selected for further evaluation.

To measure the influence d_a of the image augmentation a on the sim-to-real generalization of the model m, the performance $e(m_a)$ of the model m_a trained with augmentation a is related to the performance $e(m_b)$ of the baseline model m_b using

$$d_a = \frac{e(m_a) - e(m_b)}{e(m_b)}.$$

It quantifies the relative improvement of the model trained with augmentations compared to the baseline model.

To enhance the generalizability of the results, this work considers multiple datasets and perception tasks, including object detection and semantic segmentation. Both tasks use different evaluation metrics. This work employs mAP for object detection (cf. Section 2.1.1) and mIoU for semantic segmentation (cf. Section 2.1.2). As it is clear from the context which evaluation metric is used,

no explicit differentiation is made in the remainder of this chapter.

To further enhance the generalizability of the results, this work considers multiple deep learning models. Similar to Chapter 4, and for the same reasons, the backbones of the investigated object detection models come with initial weights pre-trained on the ImageNet dataset [173] and the semantic segmentation models on the ADE20K dataset [174].

Overall, following 25 pixel-level augmentations are investigated in this work: AdvancedBlur, Blur, CLAHE, ColorJitter, Defocus, Downscale, Emboss, Equalize, FancyPCA, GaussNoise, GaussianBlur, GlassBlur, HueSaturationValue, ISONoise, MedianBlur, MotionBlur, RGBShift, RandomBrightnessContr, RandomGamma, RandomToneCurve, RingingOvershoot, Sharpen, ToSepia, UnsharpMask, and ZoomBlur. To apply these augmentations during training, their implementation from the Albumentation library [85] is used. Each augmentation is applied with its standard parameters, which include a probability of application of 50% per image for each augmentation. The detailed experimental setup is presented in Section 5.2.

Section 5.3.1 present the results of using single image augmentations. It shows a positive effect of specific augmentations. Furthermore, Section 5.3.2 expands the results by giving insights into selected combinations of augmentations. A first version of the results was published in [92].

5.2 Experimental Setup

To enhance the generalizability of the exploration of the effect of image augmentations on the sim-to-real generalization, multiple deep learning models are trained and evaluated. For object detection, Faster R-CNN [107] architectures with VGG-16 [109], ResNet-50 [105] FPN and MobileNetV3-Large [115] FPN backbones are investigated with their implementation from the Torchvision library [103]. For semantic segmentation, UPerNet [108] architectures with

ResNet-50 and ResNet-101 [105] as well as Swin-B, -S and -T [106] backbones are investigated with their implementation from the MMSegmentation library [104]. It should be noted that the UPerNet architecture uses a form of FPN by default while Faster R-CNN does not. Therefore, the FPN enhancement of the ResNet-50 backbone integrated into the Faster R-CNN architecture is explicitly stated, but is not for the ResNet-50 backbone integrated into the UPerNet architecture. All considered deep learning models, their number of trainable weights, and the performance of the baseline model on the real-world evaluation images when trained without augmentations are shown in Table 5.1.

For object detection, this selection contains a backbone from each of the major families investigated in Chapter 4. For semantic segmentation, this selection contains the backbone families that lead to the best results in Chapter 4, including CNN- as well as transformer-based models. Since Chapter 4 shows that the backbone has a much larger influence on the sim-to-real generalization of deep learning models, the investigation focuses on different backbones rather than different architectures.

Each model is trained with the hyperparameters that led to the best real-world performance in Chapter 4. While the application of augmentations on the training data might lead to a different set of optimal hyperparameters, this approach reduces computational load drastically because of the combinatorial complexity arising from combining the number of augmentations with the number of potential hyperparameters. As a result, all object detection models are trained using Adam optimizer [175]. The models with VGG-16 and MobileNetV3-Large backbones use a learning rate of 0.00001 and the model with the ResNet-50 backbone a learning rate of 0.0001. All semantic segmentation models are trained using SGD optimizer with a learning rate of 0.01, momentum of 0.9, and weight decay of 0.0005.

The models are trained on their respective object detection and semantic segmentation datasets with the splits presented in Chapter 3 for a maximum of 200 epochs. The object detection models use a training batch size of 16 and the semantic segmentation models a training batch size of 2 because of the larger image sizes. Again, to reduce computation during training when

Table 5.1: **List of all investigated models in combination with their number of trainable weights and their baseline performance on the respective real-world evaluation dataset when trained without augmentations**

Dataset	Architecture	Backbone	Performance	Weights
Ruralscapes	UPerNet	ResNet-50	49.5	66M
Ruralscapes	UPerNet	ResNet-101	44.6	85M
Ruralscapes	UPerNet	Swin-T	35.6	60M
Ruralscapes	UPerNet	Swin-S	39.4	81M
Ruralscapes	UPerNet	Swin-B	39.2	121M
KITTI	UPerNet	ResNet-50	54.3	66M
KITTI	UPerNet	ResNet-101	52.8	85M
KITTI	UPerNet	Swin-T	51.2	60M
KITTI	UPerNet	Swin-S	61.2	81M
KITTI	UPerNet	Swin-B	58.4	121M
Drogue	Faster R-CNN	MobileNetV3-L FPN	76.7	19M
Drogue	Faster R-CNN	ResNet-50 FPN	78.1	41M
Drogue	Faster R-CNN	VGG-16	52.4	44M

the model has reached a local optimum and does not improve anymore, early stopping with a patience of 10 is used. The model performing best on the validation dataset is used for further evaluation.

Overall, 3 object detection models are trained 3 times on 1 dataset and 5 semantic segmentation models are trained 3 times on 2 datasets, each with 25 separate augmentations as well as a baseline without augmentations, resulting in 1014 trained variations. Furthermore, the 3 object detection models are trained 3 times with 4 combinations of augmentations on the object detection dataset and the 5 semantic segmentation models are trained 3 times with 5 combinations of augmentations on their 2 datasets, resulting in 186 additional trained model variations.

5.3 Results

5.3.1 Influence of Single Augmentations

Semantic Segmentation

The impact of various image augmentation techniques on the sim-to-real generalization is examined by applying each augmentation separately during training and evaluating its effect on the real-world performance, as described in Sections 5.1 and 5.2. The results for the semantic segmentation datasets are presented in Figs. 5.1 and 5.2, with detailed numerical values provided in Table 5.2. The experiments demonstrate that image augmentations can significantly enhance sim-to-real generalization. Even applying a single augmentation can improve the median performance of the models on both real-world datasets by over 5%. Some models can even achieve maximum improvements of up to 14.6% on KITTI and 14% on Ruralscapes using only a single augmentation.

Four augmentations improve the median performance of the trained models compared to the baseline on both datasets: ColorJitter, RGBShift, HueSaturationValue, and ToSepia. Notably, while only 4 of the 25 augmentations improve the median performance on both datasets, the augmentations that improve it the most are similar: The top-3 augmentations on both datasets include ColorJitter and RGBShift, which alter the colors of the synthetic images. This finding supports the conclusion from [24], that sim-to-real generalization issues are largely attributed to the general coloration of synthetic data, and aligns with [30], which shows that textures in synthetic environments lack realism.

Moreover, the experiments reveal that the transformer-based models also benefit from applying ColorJitter and RGBShift augmentations. Although current literature suggests that transformer architectures focus more on shape rather than on texture [70, 71], color differences appear to have a significant impact on the sim-to-real generalization for transformers as well.

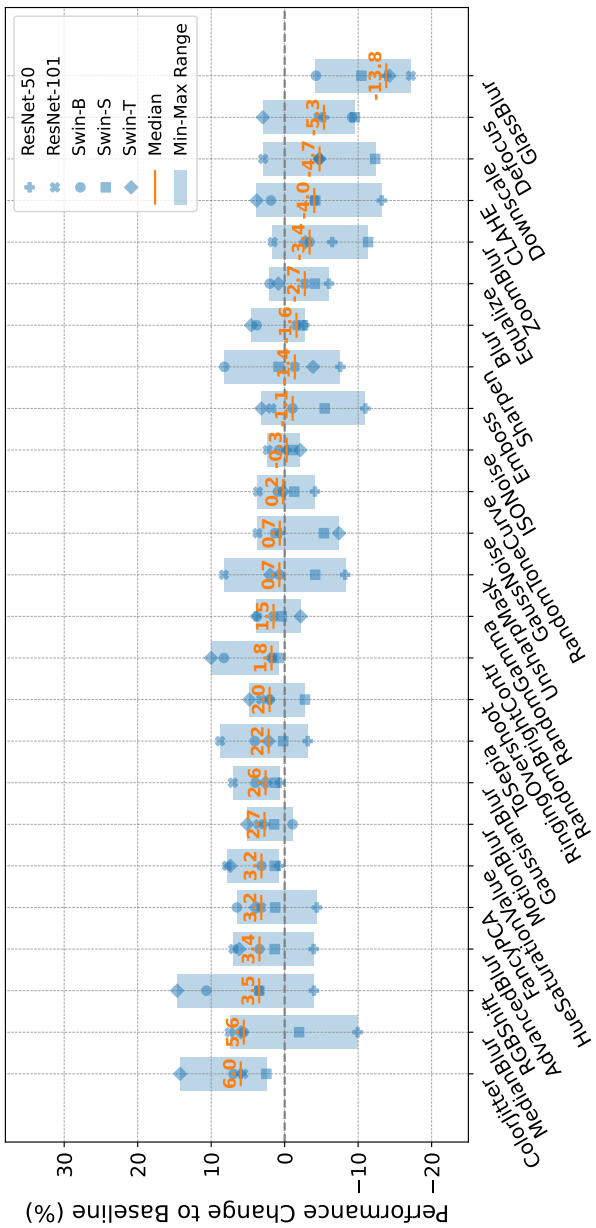


Figure 5.1: **Relative change of performance to the baseline model on the real-world KITTI evaluation dataset when applying the specified image augmentations during training. First published in [92]**

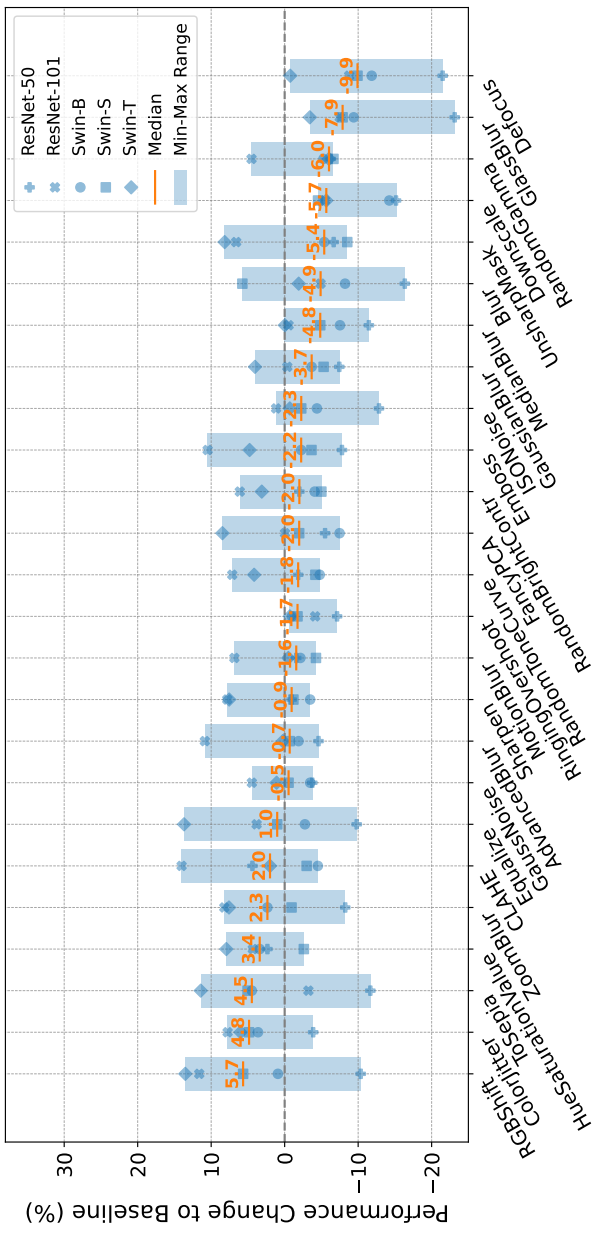


Figure 5.2: **Relative change of performance to the baseline model on the real-world Ruralscapes evaluation dataset when applying the specified image augmentations during training. First published in [92]**

Table 5.2: **Relative change of performance to the baseline model on the real-world semantic segmentation evaluation datasets when applying the specified image augmentations during training. Values greater than zero in bold. Bottom section gives statistical values. Improvement value counts augmentations with improvements. Abbreviations: R for ResNet, S for Swin**

	Ruralscapes					**KITTI**				
	R-101	**R-50**	**S-B**	**S-S**	**S-T**	**R-101**	**R-50**	**S-B**	**S-S**	**S-T**
AdvancedBlur	**10.9**	-4.6	-1.9	-0.7	**0.3**	**7.0**	-3.9	**3.4**	**1.4**	**6.2**
Blur	-4.9	-16.4	-8.2	**5.8**	-1.9	-1.6	-2.7	**3.8**	-2.5	**4.6**
CLAHE	**14.0**	**4.4**	-4.5	-3.0	**2.0**	-4.0	-13.2	**1.9**	-4.2	**3.8**
ColorJitter	**7.8**	-3.9	**3.6**	**4.8**	**6.0**	**5.6**	**6.0**	**7.0**	**2.5**	**14.2**
Defocus	-8.8	-21.5	-11.8	-9.9	-0.8	-4.6	-5.3	-9.2	-9.5	**3.0**
Downscale	-5.5	-15.2	-14.2	-4.5	-5.7	**2.9**	-4.8	-4.6	-12.3	-4.7
Emboss	**10.5**	-7.8	-2.2	-3.7	**4.8**	**1.8**	-11.0	-1.1	-5.4	**3.2**
Equalize	**3.8**	-9.8	-2.8	**1.0**	**13.7**	-2.7	-6.0	**2.0**	-4.1	**0.8**
FancyPCA	-0.0	-5.5	-7.5	-2.0	**8.5**	**3.2**	-4.4	**6.5**	**1.3**	**4.0**
GaussNoise	**4.5**	-3.8	-3.5	-0.5	**1.1**	**3.7**	**0.7**	**1.3**	-5.3	-7.4
GaussianBlur	-0.4	-7.4	-3.7	-5.3	**4.0**	**7.0**	**0.7**	**4.0**	**1.3**	**2.6**
GlassBlur	-7.5	-23.1	-9.4	-7.9	-3.4	-17.2	-13.8	-4.3	-10.4	-14.3
HueSaturationValue	**4.3**	**2.3**	**3.4**	-2.6	**7.9**	**7.9**	**0.8**	**3.2**	**1.4**	**7.4**
ISONoise	**1.2**	-12.8	-4.4	-2.3	-0.7	**2.3**	-0.3	**0.8**	-1.1	-2.1
MedianBlur	-0.5	-11.5	-7.5	-4.8	-0.0	**7.5**	-9.9	**5.6**	-2.0	**6.0**
MotionBlur	**6.8**	-1.6	-2.1	-4.3	-0.5	**3.5**	**2.7**	-1.1	**1.4**	**5.1**
RGBShift	**11.6**	-10.3	**0.9**	**5.7**	**13.5**	**3.5**	-4.0	**10.7**	**3.5**	**14.6**
Rand.Bright.Contr	**6.1**	-2.0	-4.1	-5.0	**3.1**	**1.7**	**1.8**	**8.3**	**0.8**	**10.0**
RandomGamma	**4.5**	-6.3	-5.3	-6.6	-6.0	**1.5**	**3.9**	**3.8**	**0.4**	-2.2
RandomToneCurve	**7.1**	-1.8	-4.8	-4.2	**4.2**	**3.6**	-4.1	**0.9**	-1.3	**0.2**
RingingOvershoot	-4.2	-7.1	-1.0	-1.7	-0.7	**3.2**	**2.0**	**2.0**	-2.8	**4.8**
Sharpen	**7.8**	-0.9	-3.5	-1.2	**7.5**	-1.4	-7.6	**8.2**	**0.8**	-3.9
ToSepia	-3.2	-11.7	**4.5**	**5.0**	**11.4**	**8.8**	-3.1	**4.1**	**0.2**	**2.2**
UnsharpMask	**6.6**	-6.7	-5.4	-8.5	**8.2**	**8.3**	-8.2	**0.7**	-4.2	**2.0**
ZoomBlur	**8.2**	-8.2	**2.3**	-0.9	**7.6**	**1.6**	-6.5	-3.4	-11.3	-2.8
Minimum	-8.8	-23.1	-14.2	-9.9	-6.0	-17.2	-13.8	-9.2	-12.3	-14.3
Maximum	14.0	4.4	4.5	5.8	13.7	8.8	6.0	10.7	3.5	14.6
Median	4.5	-7.1	-3.7	-2.6	3.1	3.2	-4.0	2.0	-1.3	3.0
Mean	3.2	-7.7	-3.7	-2.3	3.4	2.1	-3.6	2.2	-2.5	2.3
Improvement	16.0	2.0	5.0	5.0	16.0	19.0	8.0	19.0	11.0	18.0

The experiments further demonstrate that not all augmentations improve the sim-to-real generalization of the semantic segmentation models but instead may even deteriorate the median real-world performance of the models. These augmentations include ISONoise, Emboss, Sharpen, Blur, Downscale, Defocus, and GlassBlur. The two augmentations that deteriorate the results the most are the same on both datasets, namely Defocus and GlassBlur. Additionally, Downscale is among the bottom-4 on both datasets. This seems reasonable as both datasets do not contain any textured glass through which objects have to be detected, there are no images out-of-focus, and all images have the same resolution. Furthermore, the GlassBlur and the Downscale augmentations produce highly pixelated images that make the synthetic images look more different to the real-world images and many object classes, especially small ones, harder to detect and distinguish. The Defocus augmentation has a similar effect, although not as pronounced. This highlights that while augmentations may improve the generalization of the models, they have to be reasonable and reflect the phenomena faced during real-world deployment.

Notably, while some augmentations have a uniform positive or negative impact on both datasets, others exhibit differing effects between the two semantic segmentation datasets. For instance, 15 of the 25 augmentations improve the median performance of the models on the KITTI dataset, whereas only 7 augmentations lead to improvements on the Ruralscapes dataset. This disparity is primarily caused by the differing effectiveness of blur augmentations. While performance on KITTI improves with many of the blur augmentations, such as Median-, Advanced-, Motion-, and GaussianBlur, performance on Ruralscapes only improves with ZoomBlur. This is likely attributed to variations in real-world camera settings, resulting in different blur characteristics between the datasets.

Effects of the Color Augmentations

To further investigate the positive effects of the color augmentations, Tables 5.3 and 5.4 present the performance improvement relative to the baseline for each annotated class. On the Ruralscapes dataset, the highest performance improve-

Table 5.3: **Per-class IoU of each UPerNet semantic segmentation model on the Ruralscapes dataset when applying the specified image augmentations during training. Difference to baseline in brackets**

(a) **ColorJitter**

	Building	Car	Greenery	Person	Road	Background
ResNet-50	60.7 (-0.2)	22.0 (+0.0)	77.7 (+1.5)	21.2 (-16.8)	26.7 (+1.7)	77.0 (+2.2)
ResNet-101	59.0 (-0.1)	22.2 (+5.1)	77.5 (+2.0)	32.6 (+20.4)	18.1 (-8.1)	78.8 (+1.4)
Swin-B	58.8 (-4.8)	2.2 (-6.0)	77.4 (+2.8)	0.0 (+0.0)	28.3 (+7.6)	76.8 (+8.8)
Swin-S	65.8 (+0.6)	0.0 (-5.8)	79.0 (+3.6)	0.0 (+0.0)	26.1 (+4.2)	77.2 (+8.9)
Swin-T	57.1 (-1.6)	0.0 (-4.0)	77.1 (+4.1)	0.0 (+0.0)	17.3 (+4.7)	75.2 (+9.7)
Mean	60.3 (-1.2)	9.3 (-2.1)	77.7 (+2.8)	10.8 (+0.7)	23.3 (+2.0)	77.0 (+6.2)

(b) **RGBShift**

	Building	Car	Greenery	Person	Road	Background
ResNet-50	60.0 (-0.9)	17.2 (-4.8)	76.7 (+0.5)	13.0 (-25.0)	23.6 (-1.4)	75.6 (+0.8)
ResNet-101	61.3 (+2.2)	19.9 (+2.8)	77.3 (+1.8)	33.2 (+21.0)	30.6 (+4.4)	76.2 (-1.2)
Swin-B	61.3 (-2.3)	0.4 (-7.8)	76.9 (+2.3)	0.0 (+0.0)	25.0 (+4.3)	73.6 (+5.6)
Swin-S	61.0 (-4.2)	12.3 (+6.5)	75.4 (+0.0)	0.0 (+0.0)	27.8 (+5.9)	73.5 (+5.2)
Swin-T	62.0 (+3.3)	11.8 (+7.8)	75.0 (+2.0)	0.0 (+0.0)	30.0 (+17.4)	64.0 (-1.5)
Mean	61.1 (-0.4)	12.3 (+0.9)	76.3 (+1.3)	9.2 (-0.8)	27.4 (+6.1)	72.6 (+1.8)

ment when using the ColorJitter augmentation is achieved for the Background class, which predominantly features the sky, with a mean improvement of 6.2 pp across all models. Notably, the per-class IoU of each model exhibits improvement for this class. The second and third largest improvements are achieved for the Greenery and Road classes, with enhancements of 2.8 pp and 2 pp, respectively. Similar to the Background class, not only the median improves, but all models improve their performance in detecting the Greenery class. Employing RGBShift results in the same three classes showing the biggest improvements, albeit in a different order: Road class with 6.2 pp, Background with 1.7 pp, and Greenery with 1.3 pp. Given the diverse colors present in these classes, such as the blue of the sky in the Background class, the green tones

Table 5.4: **Per-class IoU of each UPerNet semantic segmentation model on the KITTI dataset when applying the specified image augmentations during training. Difference to baseline in brackets. Abbreviations: Veg. for Vegetation, Sign for Pole/Sign, R for ResNet, S for Swin**

(a) **ColorJitter**

	Building	Ground	Sign	Sky	Veg.	Vehicle	Other
R-50	63.3 (+2.1)	72.6 (+13.0)	29.9 (+2.4)	81.4 (+3.7)	74.0 (+3.5)	61.5 (-1.0)	19.8 (-0.9)
R-101	59.0 (+4.7)	69.3 (-2.1)	34.8 (+1.1)	71.1 (-0.9)	71.8 (-6.0)	66.4 (+23.8)	18.4 (+0.4)
S-B	66.9 (+6.2)	74.9 (-1.0)	34.4 (+6.4)	79.9 (+0.5)	74.9 (+3.1)	74.8 (+7.7)	31.1 (+5.5)
S-S	67.2 (+0.8)	75.2 (-2.0)	31.5 (-2.1)	82.0 (+4.6)	75.8 (-1.0)	74.1 (+0.7)	33.0 (+9.7)
S-T	62.9 (+10.5)	77.6 (+3.2)	21.1 (-1.2)	81.3 (+5.9)	74.6 (+8.6)	69.0 (+14.9)	22.8 (+9.1)
Mean	63.9 (+4.9)	73.9 (+2.2)	30.3 (+1.3)	79.1 (+2.8)	74.2 (+1.6)	69.2 (+9.2)	25.0 (+4.8)

(b) **RGBShift**

	Building	Ground	Sign	Sky	Veg.	Vehicle	Other
R-50	55.0 (-6.2)	70.2 (+10.6)	29.5 (+2.0)	67.9 (-9.8)	73.0 (+2.5)	57.4 (-5.1)	11.7 (-9.0)
R-101	62.0 (+7.7)	66.5 (-4.9)	33.6 (-0.1)	76.3 (+4.3)	72.9 (-4.9)	54.9 (+12.3)	16.5 (-1.5)
S-B	69.2 (+8.5)	78.3 (+2.4)	33.7 (+5.7)	82.6 (+3.2)	78.6 (+6.8)	77.0 (+9.9)	32.7 (+7.1)
S-S	68.7 (+2.3)	78.0 (+0.8)	33.8 (+0.2)	81.2 (+3.8)	77.8 (+1.0)	73.5 (+0.1)	30.0 (+6.7)
S-T	60.7 (+8.3)	76.7 (+2.3)	30.5 (+8.2)	78.2 (+2.8)	75.1 (+9.1)	68.3 (+14.2)	21.4 (+7.7)
Mean	63.1 (+4.1)	73.9 (+2.2)	32.2 (+3.2)	77.2 (+0.9)	75.5 (+2.9)	66.2 (+6.3)	22.5 (+2.2)

in Greenery, and the gray shades of the Road class, this suggests that a wide variety of colors seems to have some relevant mismatch between the synthetic and the real-world images and that using color augmentation techniques is an effective way to mitigate its effect.

On the KITTI dataset, both ColorJitter and RGBShift augmentations achieve the biggest improvements for the Vehicle class, with gains of 9.2 pp and 6.3 pp, respectively. As the cars in the real-world dataset come in many different colors, it is reasonable to assume that the augmentations may improve the representation of cars in general and not specific for sim-to-real generalization.

(a) **ColorJitter** (b) **RGBShift**

(c) **Without Augmentations**

Figure 5.3: **Visualization of an exemplary effect of applying ColorJitter and RGBShift augmentations on a synthetic image. Image without augmentations from [89]**

However, the mean performance of the models improves for all classes with both augmentations. This again indicates that many colors seem to have some relevant mismatch between the synthetic and the real-world images, whose effect can be effectively mitigated using these color augmentations. Overall, there does not seem to be one particular class or color that has a significantly larger negative impact on the sim-to-real generalization but a general difference is present.

Notably, the application of the color augmentations seems to reduce the negative effects of color differences although they sometimes produce images with unrealistic colors, which are more dissimilar to the real-world images than the original synthetic images, as presented in Figs. 2.3 and 5.3. This suggests that

the color augmentations may be inducing some form of domain randomization on the colors, instead of making them more realistic. As discussed in Section 1.2, one branch of domain randomization repeatedly swaps the texture of the object and intentionally renders unrealistic ones. This prevents the models from depending on texture for class detection but incentivizes them to learn object shapes. Building on the presented observation that object shapes in simulation environments are often realistic while textures are not, this should allow the models to generalize better to real-world images. During deployment, the real-world images can then be viewed as just another variation of the learned classes.

In conclusion, the findings support the hypothesis that synthetic images exhibit color differences relevant for sim-to-real generalization and show that this problem persists over various classes and colors. Furthermore, the results provide additional evidence for the benefits of domain randomization and showcase the effectiveness of a simple-to-use way for achieving color randomization through image augmentations. The results underscore the benefits of incorporating color augmentations into the training of deep learning models, enhancing their ability to generalize from simulated to real-world environments.

Object Detection

The impact of the various image augmentation techniques on the sim-to-real generalization on the drogue detection dataset are presented in Fig. 5.4, with detailed numerical values provided in Table 5.5. The object detection models also exhibit numerous improvements with the application of image augmentations. Notably, the median performance of the models improves for 18 out of the 25 augmentations, surpassing the number of improvements observed in the semantic segmentation models. The magnitude of these improvements is similar to those seen for the semantic segmentation models, with the exception of the VGG-16 models, which demonstrate improvements of up to over 30% compared to the baseline for multiple augmentations. This phenomenon is explored further in a later section.

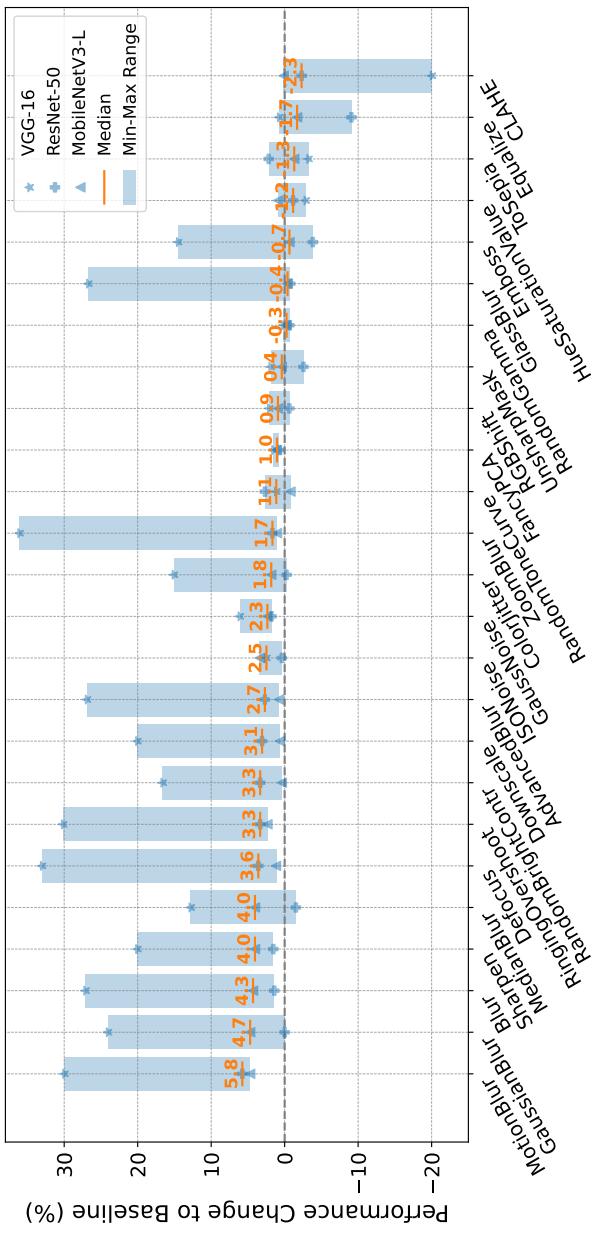


Figure 5.4: **Relative change of performance to the baseline model on the real-world drogue detection evaluation dataset when applying the specified image augmentations during training. First published in [92]**

Table 5.5: **Relative change of performance to the baseline model on the real-world drogue detection evaluation dataset when applying the specified image augmentations during training. Values greater than zero in bold. Bottom section gives statistical values. Improvement value counts augmentations with improvements**

	MobileNetV3-L	ResNet-50	VGG-16
AdvancedBlur	**0.8**	**2.7**	**26.9**
Blur	**4.3**	**1.4**	**27.1**
CLAHE	**0.1**	-2.3	-20.0
ColorJitter	**1.8**	-0.3	**15.1**
Defocus	**1.2**	**3.6**	**33.0**
Downscale	**0.7**	**3.1**	**20.0**
Emboss	-0.7	-3.8	**14.5**
Equalize	-1.7	-9.1	**0.8**
FancyPCA	**1.0**	**0.8**	**1.5**
GaussNoise	**2.3**	**1.8**	**6.1**
GaussianBlur	**4.7**	0.0	**24.0**
GlassBlur	-0.4	-0.8	**26.7**
HueSaturationValue	**0.9**	-1.2	-2.9
ISONoise	**3.4**	**0.4**	**2.5**
MedianBlur	**4.0**	-1.5	**12.8**
MotionBlur	**4.7**	**5.8**	**30.0**
RGBShift	**0.9**	-0.6	**2.1**
Rand.Bright.Contr	**0.4**	**3.3**	**16.6**
RandomGamma	-0.3	-0.6	**0.2**
RandomToneCurve	-0.8	**2.7**	**1.1**
RingingOvershoot	**2.3**	**3.3**	**30.2**
Sharpen	**4.0**	**1.5**	**20.0**
ToSepia	-1.3	**2.2**	-3.2
UnsharpMask	**0.4**	-2.6	**1.9**
ZoomBlur	**1.0**	**1.7**	**36.1**
Minimum	-1.7	-9.1	-20.0
Maximum	4.7	5.8	36.1
Median	0.9	0.8	14.5
Mean	1.4	0.5	12.9
Improvement	19.0	14.0	22.0

In contrast to the semantic segmentation models, the number of augmentations that improve the performance of all models, rather than just the median, is also significantly higher. Specifically, 12 augmentations lead to improvements for all models, compared to 4 and 0 for the semantic segmentation datasets. This disparity may be attributed to the comparably simple visual appearance of the drogue detection dataset. As the dataset does not contain that much variation, the diversity introduced by the various forms of augmentations may have a much greater impact than for the semantic segmentation datasets which are much more diverse and complex by themselves.

Consistent with the findings for the semantic segmentation datasets, ColorJitter and RGBShift improve the median performance of the deep learning drogue detection models compared to the baseline. As discussed for semantic segmentation above, this aligns with the literature, which states color differences as a primary contributor to sim-to-real generalization issues. However, the improvements achieved by these color augmentations are smaller on the drogue detection dataset. Furthermore, HueSaturationValue, which improved performance on the semantic segmentation datasets, does not provide any improvements on the object detection dataset. A possible explanation for this discrepancy again lies in the visual simplicity of the drogue detection dataset. As the images do not contain as much color variation but mostly a blue background and gray objects, the influence of potential color differences seems to be much smaller. Nevertheless, there is a measurable effect highlighting again the influences of colors on the sim-to-real generalization.

Similar to the semantic segmentation datasets, GlassBlur and Emboss deteriorate the results, which is reasonable given that the synthetic and real-world images are not captured through textured glass and do not exhibit embossing effects. Therefore, these augmentations should not be relevant and there seems to be no positive impact on the sim-to-real generalization.

While the effect of using blur differed for the semantic segmentation datasets, the object detection models improve much when using blur augmentations. The largest improvements are achieved with MotionBlur, which is reasonable given the consistent movement of the drogue in the depicted situations. Additionally, the models improve in median using GaussianBlur, Blur, MedianBlur,

AdvancedBlur, and ZoomBlur. Since the synthetic drogue detection dataset does not contain blurred images due to the data generation process, these augmentations appear to align the synthetic dataset more closely with the real-world images, which sometimes appear blurry even to the human eye.

Model and Capacity Influences

From a model-centric perspective, the semantic segmentation models with a ResNet-101 backbone improve with the highest number of augmentations across both semantic segmentation datasets as shown in Table 5.2. Conversely, the semantic segmentation models with ResNet-50 backbone improve with the lowest number of augmentations across both datasets, also displaying the lowest median and mean improvements among all considered semantic segmentation models. It would be reasonable to relate this disparity to the baseline performance as models with a lower baseline performance have more room for improvement. However, based on the experiments, the disparity cannot be solely attributed to the baseline performance of the models as the models with the ResNet-50 and ResNet-101 backbone have the best and second-best baseline on the Ruralscapes dataset. Instead, it is likely that the ResNet-50 backbone, with its lower number of trainable weights, is already near its capacity limit, making it unable to effectively learn from the added diversity provided by the augmentations. In contrast, the ResNet-101 backbone, with its greater number of trainable weights, may have more capacity left to learn from the information introduced by the augmentations, thereby benefiting more from their application.

A similar trend is observed in the object detection models in Table 5.5, where the MobileNetV3-Large, with the fewest trainable weights, exhibits the smallest maximal improvement and deterioration. This suggests that the limited capacity of the model enables it to learn relatively robust weights by itself, reducing overfitting but instead leading to good generalization. However, on the other hand, this limited capacity also seems to prevent the model from significant improvements when applying image augmentations, as it is unable to effectively learn from the additional diversity and information.

While the capacity of the model is a contributing factor, the results cannot be explained solely based on it, as shown by the Swin-B and Swin-S models, which possess more trainable weights than ResNet-101 but do not consistently exhibit more improvements. Instead, the performance of the baseline model also seems to play a role. For instance, the Swin-T model, with the smallest capacity among the transformers, benefits the most from the augmentations, likely due to its low baseline performance as shown in Table 5.1. This underscores the influence of the baseline performance on the effectiveness of augmentations, as models with a lower baseline performance have more room for improvement and vice versa.

The object detection model with the VGG-16 backbone seems to combine a relatively high capacity and low baseline performance. As a result, it exhibits the highest improvement and has the greatest number of augmentations that improve its performance on the real-world evaluation data, as shown in Table 5.5. However, even for this model, not all augmentations are beneficial, as evidenced by its maximum deterioration of -20% for some augmentation.

Furthermore, the application of image augmentations does not yield greater improvements for transformer-based models compared to CNN-based models. Section 4.3.2 showed that transformer-based models do not substantially outperform CNN-based ones in terms of sim-to-real generalization despite their promising characteristics. It was hypothesized that this might be because of the relatively small dataset size, as transformers usually have a strong dependence on large datasets and struggle to outperform CNNs on smaller ones [178, 179]. The use of image augmentations during training increases dataset diversity and size, but transformers do not exhibit significantly larger improvements than the CNN-based ResNet models. The median improvements and the number of augmentations that improve performance are in-between the ResNet models, with the mean improvements sometimes marginally better for the transformers. This suggests that the small dataset size and diversity may not be the only reasons why the transformers-based models do not outperform the CNN-based ones in sim-to-real generalization, despite their promising characteristics.

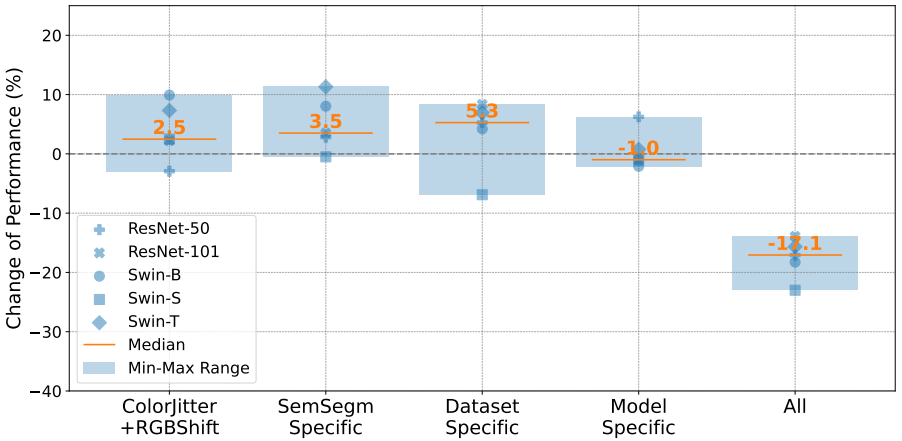
5.3.2 Influence of Combinations of Augmentations

In this section, the effect of combining multiple image augmentations on the sim-to-real generalization of deep learning models is investigated. Building on the findings from Section 5.3.1, which demonstrate that certain augmentations improve model performance when applied in isolation, this section explores whether combining these augmentations can further enhance performance.

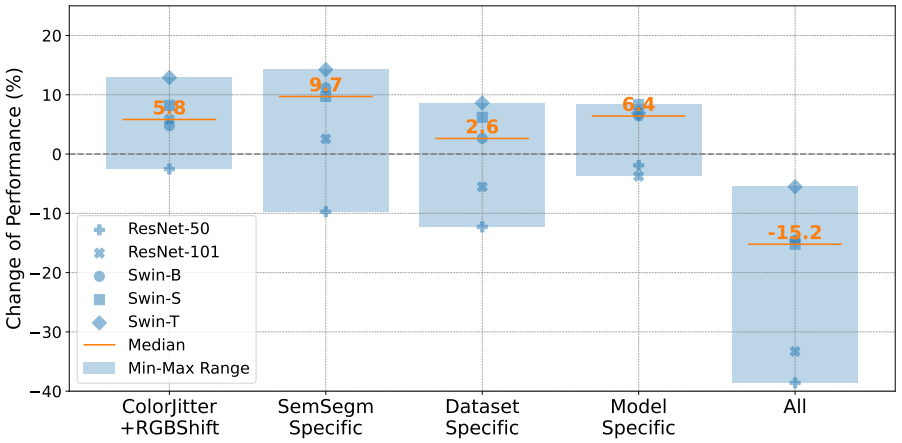
As a first baseline, a combination of all investigated augmentations, each applied with a probability of 50%, is evaluated. However, as shown in Table 5.6, this brute-force approach results in a deterioration of the median real-world performance of the models, rather than an improvement. Notably, even on the object detection dataset, where most augmentations have a positive effect, combining all augmentations leads to a decline in median performance. Specifically, all models except the object detection model with a VGG-16 backbone experience a decrease in performance. Therefore, more fine-grained combinations are investigated.

Given the large number of potential combinations of the augmentations, this study focuses on evaluating four promising combinations, in addition to combining all augmentations. The results are visually presented in Figs. 5.5 and 5.6, with numerical values provided in Table 5.6.

The first combination aggregates ColorJitter and RGBShift, which lead to median improvements on all three datasets. While this combination results in slight median improvements on all three datasets, it does not necessarily surpass the performance achieved by applying either augmentation in isolation. On the Ruralscapes dataset, the combination results in marginally better results than the best augmentation in isolation, while on the object detection dataset, the improvement is greater than that achieved by either augmentation alone. Although the combination also leads to an improvement on the KITTI dataset, it is inferior to the performance achieved by either of the two augmentations in isolation. Nevertheless, only two out of the 13 considered models experience a decline in performance when applying this combination during training.



(a) **KITTI dataset**



(b) **Ruralscapes dataset**

Figure 5.5: **Relative change of performance to the baseline model on the real-world semantic segmentation evaluation datasets when applying the specified combinations of image augmentations during training**

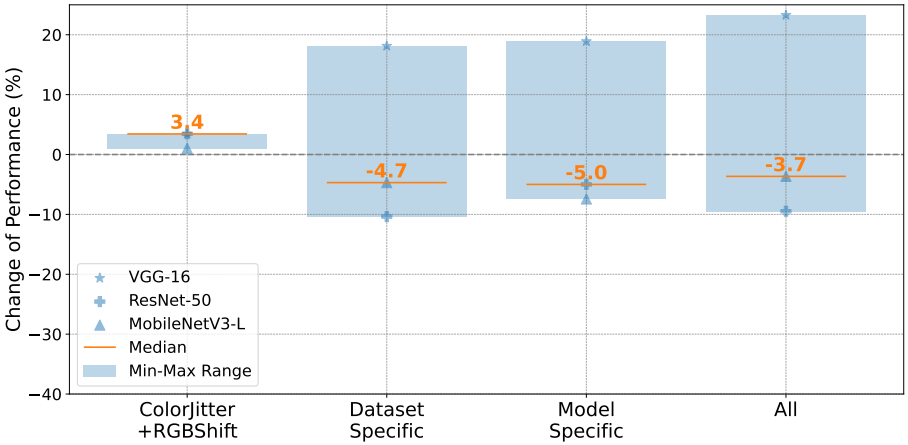


Figure 5.6: **Relative change of performance to the baseline model on the real-world drogue detection evaluation dataset when applying the specified combinations of image augmentations during training**

The second combination contains ColorJitter, RGBShift, HueSaturationValue, and ToSepia, which all improve performance on both semantic segmentation datasets. Because of that, this combination is only evaluated on the semantic segmentation datasets. The results demonstrate that combining these augmentations improves the median performance on both datasets and leads to better median real-world performance than combining only ColorJitter and RGBShift. While the median improvement using this combination is still smaller than that achieved by some isolated augmentations on the KITTI dataset, it outperforms the previous best combination on the Ruralscapes dataset by 3.9%. Notably, the Swin-T model achieves an improvement of 14.3%, which is large, but still 0.3% lower than the improvement achieved by RGBShift in isolation.

The third combination is dataset-specific, consisting of all augmentations that lead to a median improvement on the specific dataset. Therefore, for each dataset, it contains the augmentations for which the median improvement in Figs. 5.1, 5.2 and 5.4 is greater than zero. The fourth combination is model-

Table 5.6: **Relative change of performance to the baseline model on the real-world evaluation datasets when applying the specified combinations of image augmentations during training. Best value for each column in bold, worst in italic**

(a) **KITTI dataset**

	ColorJitter +RGBShift	SemSegm Specific	Dataset Specific	Model Specific	All
ResNet-101	2.2	3.5	**8.3**	-1.0	**-13.9**
ResNet-50	*-2.9*	2.8	5.3	**6.2**	-17.1
Swin-B	**9.9**	8.0	4.2	*-2.1*	-18.3
Swin-S	2.5	*-0.5*	*-6.9*	-1.0	*-23.0*
Swin-T	7.3	**11.3**	6.9	0.8	-15.7

(b) **Ruralscapes dataset**

	ColorJitter +RGBShift	SemSegm Specific	Dataset Specific	Model Specific	All
ResNet-101	5.8	2.6	-5.5	*-3.7*	-33.3
ResNet-50	*-2.5*	*-9.7*	*-12.3*	-1.9	*-38.6*
Swin-B	4.8	11.2	2.6	6.4	-14.5
Swin-S	8.2	9.7	6.2	**8.4**	-15.2
Swin-T	**12.9**	**14.3**	**8.6**	6.9	**-5.5**

(c) **Drogue detection dataset**

	ColorJitter +RGBShift	SemSegm Specific	Dataset Specific	Model Specific	All
MobileNetV3-L	*1.0*	—	-4.7	*-7.4*	-3.7
ResNet-50	**3.5**	—	*-10.4*	-5.0	*-9.5*
VGG-16	3.4	—	**18.1**	**18.9**	**23.3**

specific, comprising all augmentations that result in an improvement for the specific model on the dataset. Therefore, for each model on each dataset, it contains all augmentations that lead to an improvement greater than zero in Table 5.2. The findings suggest there is potential to be found but combining multiple augmentations is a complex task that does not necessarily lead to improved results. On the positive side, the model-specific augmentation combi-

Figure 5.7: **Example image on which multiple augmentations are applied consecutively. Each of the 25 investigated augmentations is applied with a probability of 50%. As a reference, the top left image is without augmentations**

nations result in a median improvement larger than that of any augmentation in isolation on the Ruralscapes dataset. However, the other combinations do not achieve median improvements compared to augmentations in isolation. For half of the combinations, the median of the model performances even deteriorates.

Especially the model-specific combinations suggest that combining too many augmentations may be a source of this problem. On the KITTI and the object detection dataset, where 8-19 and 15-22 augmentations are combined, the selections lead to a median deterioration while on the Ruralscapes dataset, where only 2-16 are combined, they lead to a median improvement. Fig. 5.7 shows some example images on which multiple augmentations are applied consecutively. Each of the 25 investigated augmentations is applied with a probability of 50%. The figure illustrates that the combination of too many

augmentations can lead to images in which some objects, like the humans on the street, are hardly visible anymore, preventing the model from learning effective features to detect them.

These findings underscore that using image augmentations is a promising approach to improve the sim-to-real generalization but careful consideration must be given to the selection and combination of augmentations to achieve optimal results.

5.4 Summary

This chapter addresses Research Question 3 from Section 1.3 and explores the potential of basic pixel-level image augmentation techniques to enhance the sim-to-real generalization capability of deep learning models. An extensive evaluation of 25 augmentations is conducted on 5 semantic segmentation models and 3 object detection models across multiple datasets.

The experiments demonstrate that image augmentations can significantly boost the real-world performance of deep learning models trained on synthetic data. Notably, even a single augmentation can improve the median real-world performance by over 5% on all datasets, with some models achieving maximum improvements of up to 36.1% on the object detection, 14.6% on the autonomous driving, and 14% on the UAS dataset.

The results highlight the impact of color differences between the synthetic and the real-world images on the sim-to-real generalization issues, as RGBShift and ColorJitter are the only augmentations that improve the median performance of the models across all datasets. Furthermore, transformer models also show improved performance with these two augmentations, indicating that the coloration plays a significant role in their sim-to-real generalization issues as well, although current literature states that they pay more attention to shape than to texture. The analysis of the per-class performance reveals that the color

augmentations provide benefits for various classes and colors, highlighting a general color difference instead of only between particular classes or colors. Additionally, the results provide further evidence for the benefits of domain randomization and showcase the effectiveness of a simple-to-use way for achieving color randomization through image augmentations.

The experiments also reveal that applying image augmentations does not yield greater improvements for transformer-based models than for CNN-based models. Transformers seem to be promising for sim-to-real generalization because of their shape-bias and the realistic representation of shape in game engines (cf. Section 1.2) but did not manage to outperform CNN-based models in Chapter 4. It was hypothesized that this might be because of the relatively small dataset size and diversity, as transformers usually have a strong dependence on large datasets and struggle to outperform CNNs on smaller ones [178, 179]. However, despite the increased diversity and size of the dataset provided by image augmentations, transformers did not exhibit significantly larger improvements than CNN-based models. This suggests that the dataset size and diversity are not the only factors that limit the sim-to-real generalization of the transformers but that there might be more hidden factors.

The study further confirms the literature that adding noise and blur can improve generalization from synthetic to real-world images. However, the effectiveness of these augmentations depends on the specific dataset and the conditions the model will face during deployment. Although not all considered datasets improve with the addition of blur augmentations, its usage should be considered especially when faced with situations in which the camera or the observed objects move a lot.

While the results show that image augmentations can have a positive impact on sim-to-real generalization, they also reveal that not all augmentations are beneficial but some may even degrade performance. The effect of the augmentations partly differed between the datasets but the experiments identified GlassBlur and Emboss as augmentations that resulted in lower median real-world performance across all datasets. To some extent, augmentations seem to be more effective when they reflect real-world phenomena faced during deployment.

Additionally, the experiments show that not all models benefit equally from image augmentations. Models with a higher capacity and more trainable weights tend to benefit more from augmentations. However, the general learning capability of the model also plays a role, as models with high baseline performance usually have less room for improvement.

A natural way for trying to increase the promising results of applying image augmentations in isolation is to use combinations of different ones. However, the experiments reveal that combining augmentations without careful consideration may harm the real-world performance and, in general, combining multiple augmentations in a useful way seems to be a difficult task.

Overall, this chapter provides valuable insights into the effective use of image augmentations for improving the sim-to-real generalization of deep learning models. It highlights the benefits of color and blur augmentations, the influence of the model capacity, and the importance of considering real-world phenomena when selecting augmentations. While increasing photorealism remains a promising approach to improve sim-to-real generalization, image augmentations offer a practical solution for improving generalization when using current state-of-the-art game engines.

6 Suitability of Synthetic Data for Selected Aviation Use-Cases

This chapter presents the final investigation of Research Question 1, which explores how well deep learning models trained on synthetic images perform in terms of accuracy-related metrics on real-world images in selected aviation use-cases. Building on the knowledge gained in the preceding chapters, this chapter evaluates deep learning models trained on synthetic images compared to those trained on real-world images directly. The primary objectives are to assess how well these models perform relative to their real-world counterparts and to determine the suitability of synthetic images for training deep learning models in aviation use-cases.

The chapter is organized along the two Research Sub-Questions 1.1 and 1.2, each addressing a distinct aviation use-case with specific characteristics, that has received limited attention in the literature. Specifically, Section 6.1 focuses on drogue detection in air-to-air refueling (Research Sub-Question 1.1), while Section 6.2 explores semantic segmentation from low-altitude UAS perspective (Research Sub-Question 1.2). By examining these two use-cases, this chapter provides a comprehensive understanding of the potential benefits and limitations of using synthetic images for training deep learning models in aviation applications.

6.1 Drogue Detection

This section investigates Research Sub-Question 1.1, which explores the suitability of synthetic data for training an object detection model for drogue detection in an air-to-air refueling scenario. Unlike many use-cases examined in the literature, this scenario presents an interesting combination of conditions that may facilitate better generalization from synthetic to real-world images. Specifically, the high altitude and limited variability of participating objects reduce the occurrence of shadows, reflections, distractions, and occlusions, resulting in images with reduced visual complexity, as discussed in Section 1.3. Furthermore, the limited number of objects in sight, typically one drogue and one tanker aircraft, as well as the absence of complex ground textures, contribute to a more controlled and reproducible environment. This seemingly simpler setting makes drogue detection a suitable starting point for investigating the sim-to-real generalization of deep learning models in aviation use-cases.

The development of the data generation toolchain and process used to generate the synthetic images are presented in Section 3.1.3. Utilizing a publicly available game engine and additional tools, the data generation process employed parameters according to standardized air-to-air refueling procedures. The generation process was further designed to ensure that the drogue could be depicted in all potential positions within the image and to introduce some diversity in environmental conditions. Overall, the effort required for developing the toolchain and generating the synthetic data was manageable with the publicly available tools and guidance.

In this section, deep learning drogue detection models trained on synthetic images are compared to a corresponding model trained on real-world images directly. As found in Chapter 4, the choice of the deep learning model significantly influences sim-to-real generalization, with a Faster R-CNN [107] with a ResNet-50 [105] FPN backbone achieving the best real-world drogue detection performance when trained on synthetic images. Therefore, this model, trained without augmentations with its performance as given in Table 5.1, is selected as the first model for the comparison.

The impact of image augmentations on the generalization was further investigated in Chapter 5, revealing that the real-world performance can be improved using image augmentation techniques during training. For the Faster R-CNN with ResNet-50 model, applying motion blur to the training data yielded the largest improvement and the best overall real-world performance. This model variation is selected as the second model for comparison.

To assess the suitability of synthetic images for training drogue detection models, the models are compared to a model trained directly on real-world images. To do so, the same Faster R-CNN model with ResNet-50 FPN backbone is trained on the designated real-world dataset presented in Section 3.1.2, comprising 414 images. Although this dataset is significantly smaller than the synthetic one, this setting aligns with the assumption that usually only limited real-world data is available in aviation use-cases, if any exists at all.

The training of the model on real-world images is done with the same hyperparameters that were used for the other two models and presented in Section 5.2. Therefore, it is trained with Adam optimizer [175], a learning rate of 0.0001, for a maximum of 200 epochs, and with a batch size of 16. Furthermore, early stopping with a patience of 10 is employed again to reduce computation when the model reaches a local optimum and does not improve anymore. Similar to the models trained on synthetic images considered in this comparison, the model is trained three times and the variation with the best real-world performance is used for further evaluation.

The performance of the three models on the real-world evaluation dataset is presented in Table 6.1. Notably, the models trained on synthetic images outperform the model trained directly on real-world images, despite not having seen real-world images to adapt their weights. In contrast, the model trained on real-world images was trained on images from the same setting as the evaluation dataset but only from a different refueling approach. The accuracy advantage of the models trained on synthetic images over the model trained on real-world images is also substantial, with improvements of 13.8 pp for the model trained without augmentations and 18.3 pp for the one trained with motion blur augmentations.

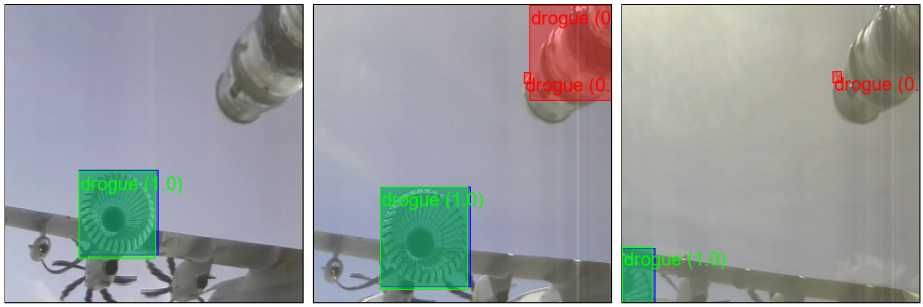
Table 6.1: **Performance of the drogue detection models on the real-world evaluation dataset when trained on different training datasets**

Training Dataset	mAP
Real-World	64.3
Synthetic	78.1
Synthetic with Augmentations	82.6

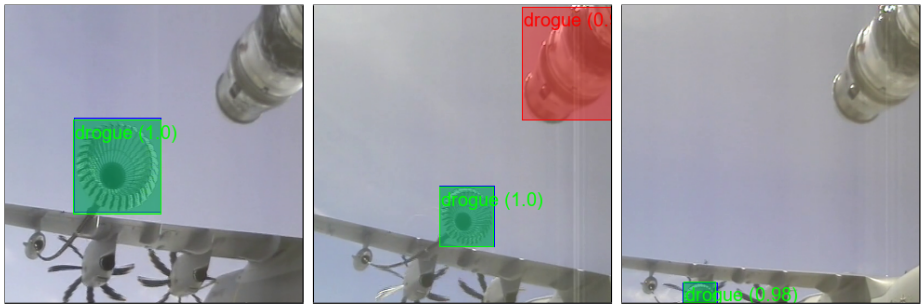
Example detections from the three models on the evaluation dataset are shown in Fig. 6.1. These images do not represent the real distribution of the detections but highlight key strengths and weaknesses of the models. The left column demonstrates that all models are able to detect the drogue with a high confidence under fair conditions and when it is fully visible.

However, the models also make false positive detections, as illustrated in the middle column. One common failure case for the models trained on synthetic images is the detection of the probe as a drogue, possibly due to the simplified, noticeably different appearance of the probe in the synthetic training images to the real-world one. Nevertheless, these detections often have a low confidence, allowing for effective filtering during deployment. A deployed system could also ignore the area around the probe altogether, removing many false positives and improving the overall performance of the models further. In contrast, the real-world model rarely detects the probe as a drogue, having learned to ignore it from the real-world training images as they show exactly the same probe from the same camera position. However, the model trained on the real-world images also makes false positive detections and some of them are much harder to filter out. The example shows a false positive detection of the drogue where the drogue is predicted larger than it actually is. The model also detects the same drogue with the correct size but both detections are paired with the same high confidence making it difficult to select the correct detection without additional information.

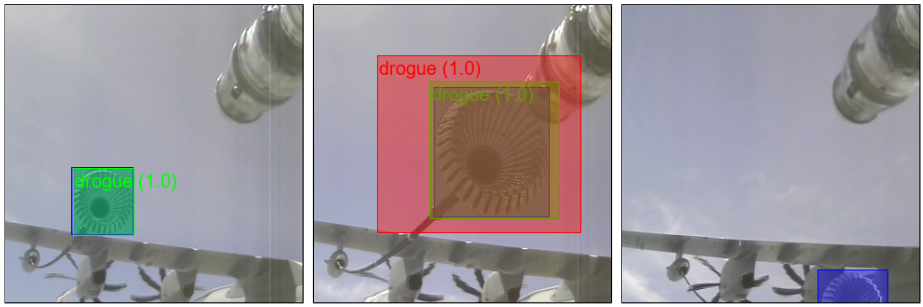
Another failure case frequently observed in the model trained on the real-world data is that it misses drogues that are not fully visible in the image, as shown in the right column. The models trained on the synthetic data are more capable of



(a) **Synthetic training dataset without augmentations**



(b) **Synthetic training dataset with motion blur augmentations**



(c) **Real-world training dataset**

Figure 6.1: **Example images from the evaluation dataset with predictions from the drogue detection models to highlight different strengths and weaknesses of the models**

detecting such drogues, often even with high confidence, as shown in the figure. This phenomenon likely results from the larger size and diversity of the synthetic dataset, which was specifically created to depict drogues in various positions, including partially occluded or outside the image. Therefore, the models are able to detect such drogues as well. The real-world dataset however, which is much smaller as real-world data is often unavailable in aviation use-cases, does not contain that much variety in drogue positions. Therefore, the model trained on real-world images did not have as many partially visible drogues to learn from. Overall, for this use-case, the larger diversity in drogue positions is seemingly more important than training on images from the real-world.

In conclusion, the results demonstrate that the generated synthetic images provide a suitable basis for training an object detection model with strong real-world performance. Contrary to most current state-of-the-art research, which suggests that training on synthetic images yields poor results on real-world images, this work shows that models trained on synthetic images can even outperform those trained on real-world data in the context of drogue detection when only limited real-world data is available. The visually simple and controlled environment of air-to-air refueling, characterized by a limited number of objects and minimal distractions, contributes to a situation where a systematic data generation process can create diverse datasets that provide a good foundation for learning, even if the synthetic images are not perfectly photorealistic. Since this use-case is a representative example of a simple perception task in aviation as presented in Section 1.3, these findings have direct benefits for practical applications.

6.2 Semantic Segmentation from Low-Altitude UAS Perspective

This section investigates Research Sub-Question 1.2, which explores the suitability of synthetic data for training a semantic segmentation model for application from a low-altitude UAS perspective. This scenario is inherently more complex due to the presence of multiple objects, varying object sizes, and diverse perspectives.

Because of the more complex situation, the data generation presented in Section 3.2.2 proved to be more challenging compared to the air-to-air refueling setting and required more effort. The dataset was replicated stylistically in a game engine, meaning that the synthetic dataset shows a visually similar scenery but does not mirror the positions and appearances of every object, as this would not be feasible. It was furthermore presented that the extracted annotations of the trees required some post-processing as the game engine produced much more fine-grained annotations compared to the real-world dataset that was annotated manually. Overall, the effort to create the synthetic dataset was higher compared to the air-to-air refueling use-case but still manageable under the constraint that the dataset was only recreated stylistically.

Similar to Section 6.1, the deep learning model with the best real-world performance when trained on the synthetic images without augmentations, as evaluated in Chapter 4, is selected as the first model for further comparison: An UPerNet [108] with ResNet-50 backbone [105]. Specifically, the model trained without augmentations with its performance as presented in Table 5.1 is considered first.

For this model, applying CLAHE augmentations during training on synthetic data yielded the largest improvements and the best overall real-world performance, as shown in Chapter 5. Therefore, the UPerNet with ResNet-50 backbone, trained on the synthetic images with CLAHE augmentations, is selected as the second model for comparison.

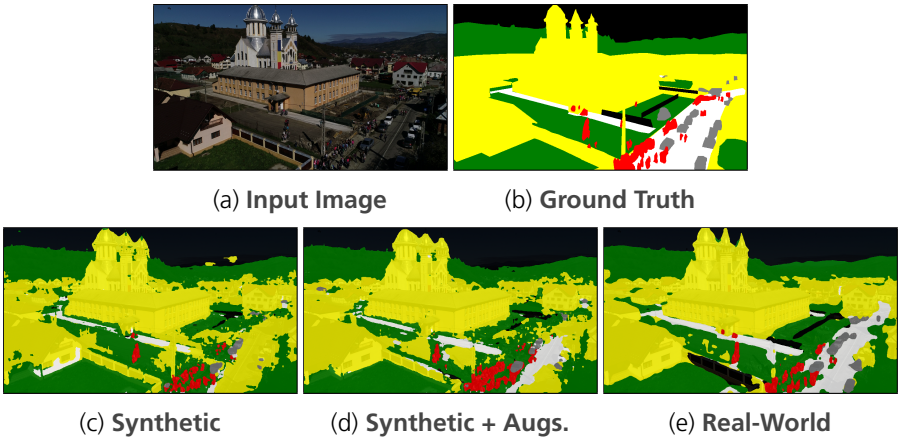
Table 6.2: **Per-class IoU of the semantic segmentation models on the real-world evaluation dataset when trained on different training datasets**

Dataset	Mean	Building	Car	Greenery	Person	Road	Background
Real-World	67.0	78.4	35.2	86.9	58.4	56.1	86.8
Synthetic	49.5	60.9	22.0	76.2	38.0	25.0	74.8
Synthetic + Augs.	51.6	61.9	25.4	75.1	43.2	30.1	74.1

To investigate the suitability of synthetic images for training semantic segmentation models, the selected models are compared to a model trained directly on real-world images. To do so, the same UPerNet with ResNet-50 backbone is trained on the real-world dataset designated for training, presented in Section 3.2.1. It contains 652 training images which is a large amount considering that the manual annotation of one image took 45 minutes on average [170]. The model trained on the real-world images is then evaluated on the same real-world evaluation dataset as the models trained on synthetic images. As described in Section 3.2.1, the evaluation dataset contains images from the same region but from different videos than in the training dataset.

The model is trained with the same hyperparameters that were used for the other two models and presented in Section 5.2. Therefore, it is trained using SGD optimizer with a learning rate of 0.01, momentum of 0.9, and weight decay of 0.0005. The model is trained for the same maximum of 200 epochs with a training batch size of two and early stopping with a patience of 10 to reduce computation when it has reached a local optimum and does not improve anymore. Similar to the models trained on synthetic images considered in this comparison, the model is trained three times and the variation with the best real-world performance is used for further evaluation.

The performance of the three models on the real-world evaluation dataset is presented in Table 6.2. In contrast to the drogue detection use-case, the comparison reveals a clear performance difference in favor of the model trained on real-world images. The models trained on synthetic images perform significantly worse than the model trained directly on the real-world dataset, with a



(a) **Input Image** (b) **Ground Truth**

(c) **Synthetic** (d) **Synthetic + Augs.** (e) **Real-World**

Figure 6.2: **Predictions from the semantic segmentation models on the evaluation dataset for the different training datasets. Annotations: Buildings in yellow, Cars in gray, Greenery in green, Persons in red, Road in white, Background in black. Input image and ground truth from [170]**

mean deviation of 17.5 pp for the model trained without augmentations and 15.4 pp for the model trained with augmentations.

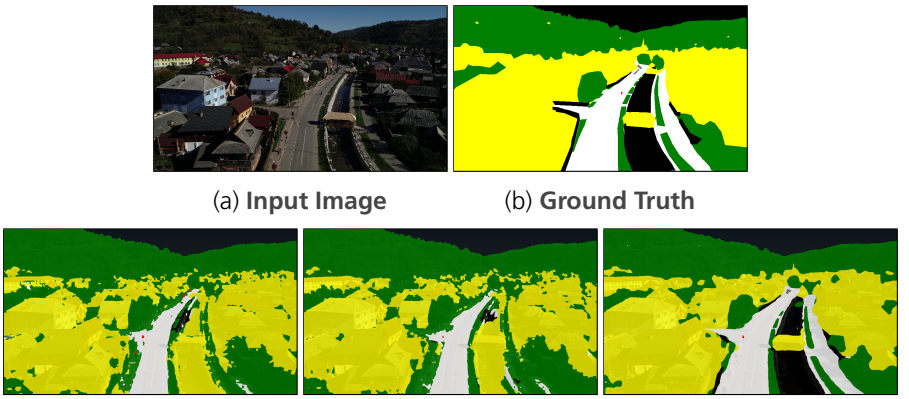
When comparing the two models trained on synthetic images, the use of augmentations leads to an improved mean real-world performance over all classes and the model outperforms the model trained without augmentations on all individual classes, except background and greenery. However, the model trained on real-world images directly exhibits better performance on all classes than the two models trained on synthetic images, particularly for the road (26 pp), building (16.5 pp), and person (15.2 pp) classes.

An example prediction from the three models on the evaluation dataset is given in Fig. 6.2. While the numerical results indicate a significant difference, visual differences can be observed but do not appear as pronounced. All predictions look relatively good, with most buildings and greenery accurately detected.

The model trained only on synthetic images misses the street, whereas the model trained with augmentations detects most of it. All models miss some cars on the street, but the models trained on synthetic images do so more frequently. Furthermore, the persons might not be detected as detailed as from the model trained on the real-world images but these fine prediction differences might not be that important from the perspective of a flying aircraft: For most use-cases it should usually be sufficient to know that there is a person and a small undetected part of the person should not influence the overall system.

Notably, the predictions from the models trained on synthetic images for the area in front of the church are very mixed and noisy. This might be attributed to the complexity of the area, which shows a construction site. In the ground truth annotations, the area is labeled as greenery but this is arguably a corner case that could as well be labeled differently. Additionally, the construction vehicle is labeled as a car, which is reasonable to some extend but unlikely to be predicted by the models trained on the synthetic images as such vehicles are not present in the synthetic training data.

In contrast, the model trained on the real-world images predicts this complex area remarkably well. A similar phenomenon of very precise predictions from the model trained on real-world images can be observed in Fig. 6.3, where the predictions of the river and the street appear to be almost perfect. These predictions also look much more accurate than in other regions, giving rise to the assumption of overfitting or memorizing. As presented in Section 3.2.1, the images in the designated real-world training and evaluation datasets are from different video sequences but from the same region. The UAS often flies along the main road, which might result in similar images from similar perspectives in both the training and evaluation datasets. A visual inspection confirms that there are images in the real-world training dataset that look very similar to the presented images from the evaluation dataset, as shown in Fig. 6.4. While not exactly the same, these images show the same situation only from a slightly different perspective, giving the model trained on them a clear advantage on the evaluation dataset. This could skew the results to some extent. However, it might also represent a real-world situation where the UAS is expected to operate in that specific region and some real-world data has been recorded.



(a) **Input Image** (b) **Ground Truth**

(c) **Synthetic** (d) **Synthetic + Augs.** (e) **Real-World**

Figure 6.3: **Predictions from the semantic segmentation models on the evaluation dataset for the different training datasets with remarkable accuracy of the real-world model for the street and the river. Annotations: Buildings in yellow, Cars in gray, Greenery in green, Persons in red, Road in white, Background in black. Input image and ground truth from [170]**

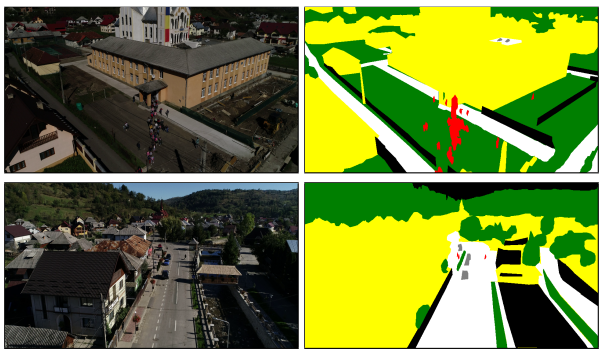
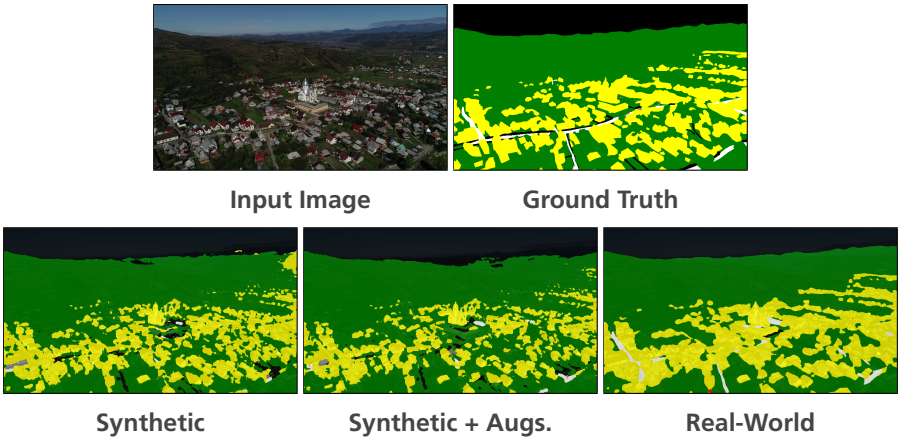


Figure 6.4: **Example images and annotations from the real-world training dataset that look similar to images from the evaluation dataset. Images and ground truth from [170]**

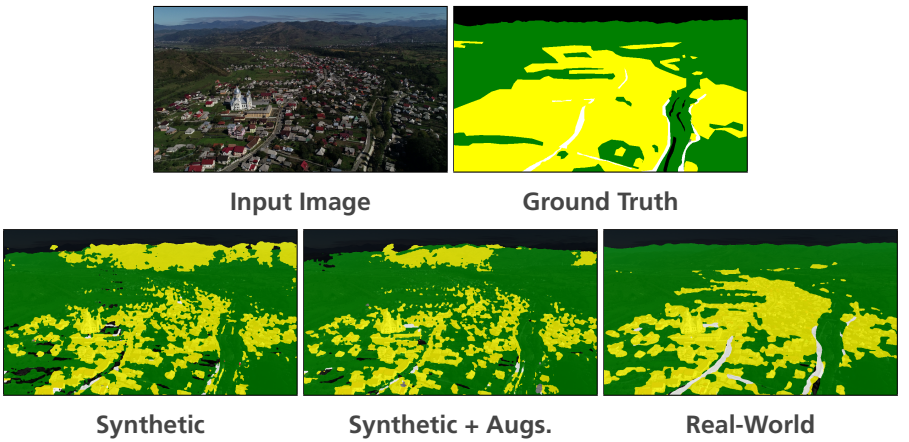
Having this advantage of the model trained on the real-world images in mind and only investigating the models trained on the synthetic images, their general real-world prediction capabilities appear promising and might even be competitive on datasets with more dissimilar real-world training and evaluation data. Furthermore, a more exact virtual representation of the region in the simulation environment, possibly derived from map data using an automated generation approach, could improve the similarity of the synthetic images and ultimately the results of the models trained on it.

Another factor that likely impacts the numerical results is inconsistencies in the manual annotations, as shown in Fig. 6.5. While both images show a similar area from slightly different perspectives, the annotations vary a lot. In Fig. 6.5a, the annotations are relatively fine-grained but in Fig. 6.5b, they are coarser, which is the more frequent case in the real-world dataset. This was probably done to reduce workload in the time-consuming manual annotation process. The model trained on these real-world annotations adapts to their coarser granularity, whereas the models trained on the synthetic images with their pixel-perfect annotations, as presented in Section 3.2.2, output much more detailed predictions. This results in differences between the predictions and the real-world ground truth annotations, leading to a lower numerical performance. However, while the predictions of the buildings on the mountains from the models trained on synthetic images in Fig. 6.5b are wrong, the fine-grained predictions in the building area look accurate and, in part, even more accurate than the coarse ground truth annotations.

Whether these fine-grained predictions and annotations contain useful information depends on the specific use-case. Nevertheless, in the current configuration, they seem to influence the results and lead to worse numerical values than a visual inspection might suggest. If fine-grained annotations are not needed, the synthetic annotations of the buildings could be made coarser using a similar post-processing method as applied to the tree annotations presented in Section 3.2.1, which would likely bring the performance of the models trained on the synthetic images closer to the real-world baseline. On the other hand, if fine-grained annotations are useful, the synthetic annotations can provide a significant benefit compared to the time-consuming effort of creating manual annotations of real-world images with such a high level of detail.



(a) **Example image with relatively fine-grained annotations**



(b) **Example image with relatively coarse annotations**

Figure 6.5: **Example images from the evaluation dataset with inconsistencies in the manual annotations, combined with predictions from the semantic segmentation models for the different training datasets. Annotations: Buildings in yellow, Cars in gray, Greenery in green, Persons in red, Road in white, Background in black. Input images and ground truth from [170]**

In conclusion, while the synthetic images currently lead to models with lower real-world performance, they demonstrate promising visual prediction accuracy and characteristics. For specific situations, they might be able to train models with a performance comparable to those trained on real-world images, especially if the synthetic images and annotations can be made more similar to the real-world data, e.g., through automated generation approaches or post-processing techniques.

6.3 Summary

This chapter addresses Research Question 1 from Section 1.3 and explores the suitability of synthetic images to train deep learning models for selected aviation use-cases. Building on the knowledge gained in the preceding chapters, this chapter evaluates deep learning models trained on synthetic images compared to models trained on real-world images directly. The primary objective is to assess the applicability of synthetic images for training deep learning models in aviation contexts and to determine how well these models perform relative to real-world counterparts. Corresponding to the partition in Research Sub-Questions 1.1 and 1.2, the chapter investigates synthetic images for two aviation use-cases with specific characteristics that have received limited attention in the literature: Drogue detection during air-to-air refueling and semantic segmentation from low-altitude UAS perspective.

For the drogue detection use-case, the experiments reveal that synthetic images provide a suitable basis for training deep learning models, with the models achieving superior performance to those trained on real-world images directly when only limited real-world data is available. This finding is contrary to most current research, which suggests that models trained on synthetic images yield poor results on real-world images (cf. Section 1.2.1). The results have widespread relevance, showcasing the potential of synthetic data to improve model performance and to reduce the need for expensive and time-consuming real-world data collection. As this use-case is a representative aviation use-case

as presented in Section 1.3, these findings further have direct benefits for practical applications.

For the much more challenging semantic segmentation use-case, the models trained on synthetic images also yield promising visual results. While the numerical results show an inferior performance compared to a model trained on real-world images directly, the differences seem to be mainly due to the presence of similar images in the real-world training and evaluation datasets, ambiguous annotations in some edge cases, and differences in level-of-detail between the pixel-perfect synthetic annotations and the coarser manual ones for the real-world images. Overall, the evaluation shows that the semantic segmentation models trained on synthetic images from low-altitude aerial perspective are able to generalize to real-world images and demonstrate promising visual prediction accuracy. Depending on the use-case, the pixel-perfect annotations of the synthetic images can further provide a significant benefit compared to the time-consuming effort required for creating real-world images with such a high level of detail in annotations.

Overall, this chapter demonstrates the potential of synthetic images for training deep learning models in aviation applications, highlighting both the benefits and limitations of this approach. The results provide valuable insights into the factors that influence the effectiveness of synthetic images for specific use-cases and fill a significant gap in the literature. By leveraging synthetic images, the costs and challenges associated with real-world data collection can be reduced, while still being able to train models with a high performance for a range of aviation applications.

7 Conclusion and Future Work

This work investigated the suitability of synthetic images for training deep learning models for selected aviation use-cases. As the two main use-cases, the visually simple but critical task of drogue detection during air-to-air refueling and the visually more complex task of semantic segmentation from low-altitude UAS perspective were considered. The investigation involved creating synthetic datasets for these use-cases and evaluating the real-world performance of deep learning models trained on the synthetic datasets compared to those trained on real-world images directly.

Furthermore, influences on the sim-to-real generalization capability of deep learning models were explored. In this context, a comprehensive evaluation of the influence of the deep learning model architecture and the potential of image augmentation techniques to improve the sim-to-real generalization of the models was conducted. By considering these underexplored factors, this work provides novel perspectives on the sim-to-real generalization of deep learning models. To ensure a broad investigation and to improve the generalizability of the findings, the experiments on the influencing factors were conducted on the two aviation datasets and a dataset from the automotive sector. Overall, a wide range of experiments was carried out using multiple datasets, perception tasks, and deep learning models, providing new insights into the potential and limitations of using synthetic data for training deep learning models as well as influences on their sim-to-real generalization capability. First presentations of parts of the results were published in [88–92].

The following section concisely answers the considered research questions. More detailed summaries for the specific research questions are provided in the summaries in Sections 4.4, 5.4 and 6.3.

Research Question 1: How well do selected deep learning models trained on synthetic images perform in terms of accuracy-related metrics on real-world images in specific aviation use-cases compared to a baseline model trained on real-world images?

The first research question investigates the suitability of synthetic images for training deep learning models in aviation. This question was further divided into two sub-questions, focusing on the specific use-cases of drogue detection during air-to-air refueling (Research Sub-Question 1.1) and semantic segmentation from low-altitude UAS perspective (Research Sub-Question 1.2).

The results presented in Chapter 6 show for the visually simple drogue detection use-case that synthetic images can provide a suitable basis for training deep learning models, with models achieving superior real-world performance to those trained on real-world images directly, outperforming them by up to 18.3 pp in mAP, when only limited real-world data is available. For the more challenging semantic segmentation use-case from low-altitude UAS perspective, the models trained on synthetic images also achieve promising visual prediction accuracy on the real-world images. While their numerical performance is 15.4 pp lower in mIoU compared to models trained on real-world images directly, the difference in performance is largely attributed to factors such as similar images in the available real-world training and evaluation datasets, ambiguous annotations, and differences in level-of-detail between the pixel-perfect annotations of the synthetic images and the coarser manual real-world annotations. A more comprehensive summary of the findings is given in Section 6.3.

Overall, contrary to most current research, the results demonstrate that synthetic images can provide a suitable basis for training deep learning models, particularly in visually simple situations and when only limited real-world data is

available. The detailed results from Chapter 6 highlight both the benefits and limitations of synthetic images and have direct benefits for practical applications. By leveraging synthetic data, the costs and challenges associated with real-world data collection can be reduced, while still being able to train models with a high real-world performance.

Research Question 2: How much do selected deep learning model architectures differ in accuracy-related metrics on real-world images when trained on synthetic images?

The second research question explores the influence of the deep learning model architecture on its sim-to-real generalization capability. The results presented in Chapter 4 show that influential and widespread models differ significantly in their sim-to-real generalization capability, with sim-to-real gaps varying by up to 17.3 pp for the investigated object detection models and by up to 15.1 pp and 27.8 pp for the semantic segmentation models on the considered datasets.

Deeper investigations revealed that the backbone of a model has a significant influence on the sim-to-real generalization and further influencing factors were identified. Notably, transformer backbones do not achieve better sim-to-real generalization than CNNs, despite literature suggesting that they have promising characteristics. Additionally, the experiments demonstrate that larger models are not necessarily better at generalizing from synthetic images to real-world ones for visually simple tasks, but that they may be required to achieve competitive results for more complex tasks like semantic segmentation. A more comprehensive summary of the findings is given in Section 4.4.

Overall, the findings provide practical insights on how to select deep learning models for good real-world performance when trained on synthetic images and present a novel perspective on the sim-to-real generalization capability of the models.

Research Question 3: What influence do selected image augmentation techniques during training of deep learning models on synthetic images have on accuracy-related metrics on real-world images?

The third research question examines the potential of image augmentations to enhance the sim-to-real generalization capability of deep learning models. The results presented in Chapter 5 show that image augmentations can significantly boost the real-world performance of the models trained on synthetic data. Using a single image augmentation technique, the median real-world performance of the considered models can be improved by over 5% on all investigated tasks and datasets. Some models can even achieve maximum improvements of up to 36.1% on the object detection dataset, as well as 14.6% on the automotive, and 14% on the aviation semantic segmentation datasets.

Color augmentations are found to be particularly effective, with RGBShift and ColorJitter being the only augmentations to improve median performance across all datasets, highlighting the impact of color differences between synthetic and real-world images on the sim-to-real generalization issues. Blur augmentations also show large improvements on some datasets but their effectiveness seems to be dependent on the specifics of the dataset. Furthermore, the results demonstrate that not all augmentations are beneficial, with some even degrading real-world performance. A more comprehensive summary of the findings is given in Section 5.4.

Overall, the findings provide practical insights into the effective use of image augmentations for improving the sim-to-real generalization of deep learning models. While increasing photorealism remains a promising approach to improve sim-to-real generalization, image augmentations offer a practical solution for improving generalization when using current state-of-the-art game engines.

In conclusion, this work provides a comprehensive evaluation of the suitability of synthetic images for training deep learning models for aviation use-cases

and explores influences on the sim-to-real generalization capability of these models. It demonstrates the potential of synthetic images to bring advances of deep learning to the aviation sector by reducing the need for expensive and time-consuming real-world data collection, and offers practical insights for deep learning model selection and training. By understanding influences on the sim-to-real generalization capability of deep learning models, practitioners can develop more effective strategies for leveraging synthetic data and academia can pay attention to reducing the underlying causes. The broad investigations using multiple perception tasks and datasets should further allow the transfer of the results to other industries with limited access to real-world data, advancing the deployment of deep learning-based perception to new domains.

While this study provides novel perspectives and fills significant gaps in the literature, it also opens up new avenues for future research. Building on the results from Research Question 1, more datasets and use-cases can be explored, including the influence of the datasets on the generalization. Particularly, exploring the generalization of the semantic segmentation models from aerial perspective, when trained on a more similar virtual representation of the real-world flight area, e.g., using automated virtual world generation approaches based on map data, could further improve the real-world performance of the models. Although this work considered datasets from different modern game engines, future work could further investigate synthetic data from other game engines as well, especially from new developments in the coming years. Additionally, by creating synthetic datasets for new scenarios, the results could also be transferred to other domains like maritime or space.

Building on the investigations of the various deep learning models in Chapter 4, additional models can be investigated on their sim-to-real generalization. In this direction, ablation studies could be useful to identify the influence of specific model components. While different models, published over the span of multiple years, have been investigated, further exploring the trend of the sim-to-real generalization capability over time could provide additional valuable insights. Especially large families of models that build on each other over time, such as certain YOLO models, could be a good starting point to do so. Furthermore, it would be interesting to investigate whether larger datasets help to improve the sim-to-real generalization capability of transformer models, as suggested in

Chapter 4. Moreover, future work should also focus explicitly on developing models with better sim-to-real generalization capability. Ultimately, this could allow the usage of deep learning models in more domains in which real-world data is difficult to acquire.

The investigations on the effects of image augmentations on the sim-to-real generalization focused on the standard parameters provided by the utilized library. Building on that, exploring the influence of changing the parameter ranges of the augmentations could lead to additional improvements. While this work shows many positive effects of using isolated image augmentations on the sim-to-real generalization, it also reveals difficulties in combining them effectively. As a result, investigating possible benefits and strategies for combining multiple augmentations seems to be a promising direction for future research. Since the number of possible combinations grows exponentially with the number of augmentations, using optimization frameworks or learning optimal augmentation strategies for synthetic images could be a good starting point for such investigations and promise to further improve the real-world performance of models trained on synthetic images.

Bibliography

[1] U.S. Navy Office of Information. (2022, 03) Image 220301-N-NO874-1008. Photo courtesy of Strike Fighter Squadron 11, Public Domain, accessed 3 January 2023. [Online]. Available: https://www.navy.mil/Resources/Photo-Gallery/igphoto/2002951692/

[2] Deutsches Zentrum für Luft- und Raumfahrt e.V. (CC BY-NC-ND 3.0).

[3] Deutsches Zentrum für Luft- und Raumfahrt e.V./Marek Kruszewski.

[4] B. Werner. (2018, 09) F-35C, Super Hornet damaged during at-sea aerial refueling. Accessed 13 December 2022. [Online]. Available: https://news.usni.org/2018/09/04/f-35c-damaged-36249

[5] G. Ziezulewicz. (2019, 12) 'Basket slap' damages Super Hornet. Accessed 13 December 2022. [Online]. Available: https://www.navytimes.com/news/your-navy/2019/12/23/basket-slap-damages-super-hornet/

[6] C. Mabeus. (2020, 02) Navy jet damaged in mid-air refueling mishap. Accessed 13 December 2022. [Online]. Available: https://www.navytimes.com/news/your-navy/2020/02/05/navy-jet-damaged-in-mid-air-refueling-mishap/

[7] J. Watson. (2020, 10) Fuel tanker pilot praised for keeping crew safe after mid-air collision with F-35. Accessed 13 December 2022. [Online]. Available: https://www.marinecorpstimes.com/news/your-

marine-corps/2020/10/01/fuel-tanker-pilot-praised-for-keeping-crew-safe-after-mid-air-collision-with-f-35/

[8] B. Lendon. (2019, 09) Training and leadership failures blamed for US Marine Corps air crash that killed 6. Accessed 13 December 2022. [Online]. Available: https://edition.cnn.com/2019/09/24/asia/japan-marine-corps-air-crash-investigation-intl-hnk/index.html

[9] European Commission Directorate-General for Mobility and Transport, "Commission Implementing Regulation (EU) 2019/947 of 24 may 2019 on the rules and procedures for the operation of unmanned aircraft," 2019.

[10] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, 2009.

[12] European Union Aviation Safety Agency (EASA) and Deadalean AG, "Concepts of Design Assurance for Neural Networks (CoDANN)," 2020.

[13] N. Aranjuelo, S. García, E. Loyo, L. Unzueta, and O. Otaegui, "Key strategies for synthetic data generation for training intelligent systems based on people detection from omnidirectional cameras," *Computers & Electrical Engineering*, vol. 92, 2021.

[14] M. Fabbri, G. Brasó, G. Maugeri, O. Cetintas, R. Gasparini, A. Ošep, S. Calderara, L. Leal-Taixé, and R. Cucchiara, "MOTSynth: How can synthetic data help pedestrian detection and tracking?" in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021.

[15] T. R. Dieter, A. Weinmann, S. Jäger, and E. Brucherseifer, "Quantifying the simulation–reality gap for deep learning-based drone detection," *Electronics*, vol. 12, no. 10, 2023.

[16] H. Lee, J. Jeon, D. Lee, C. Park, J. Kim, and D. Lee, "Game engine-driven synthetic data generation for computer vision-based safety monitoring of construction workers," *Automation in Construction*, vol. 155, 2023.

[17] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?" in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[18] M. Wrenninge and J. Unger, "Synscapes: A photorealistic synthetic dataset for street scene parsing," *arXiv preprint arXiv:1810.08705*, 2018.

[19] J. Shermeyer, T. Hossler, A. van Etten, D. Hogan, R. Lewis, and D. Kim, "RarePlanes: Synthetic data takes flight," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[20] B. Alvey, D. T. Anderson, J. M. Keller, A. Buck, G. Scott, D. Ho, C. Yang, and B. Libbey, "Improving explosive hazard detection with simulated and augmented data for an unmanned aerial system," in *Proceedings of the Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXVI*, vol. 11750, 2021.

[21] B. Kiefer, D. Ott, and A. Zell, "Leveraging synthetic data in object detection on unmanned aerial vehicles," in *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2022.

[22] K. Konen and T. Hecking, "Increased robustness of object detection on aerial image datasets using simulated imagery," in *Proceedings of the IEEE International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, 2021.

[23] M. Krump, M. Ruß, and P. Stütz, "Deep learning algorithms for vehicle

detection on UAV platforms: first investigations on the effects of synthetic training," in *Proceedings of the International Conference on Modelling and Simulation for Autonomous Systems (MESAS)*, 2019.

[24] M. Krump and P. Stütz, "UAV based vehicle detection on real and synthetic image pairs: Performance differences and influence analysis of context and simulation parameters," in *Proceedings of the International Conference on Modelling and Simulation for Autonomous Systems (MESAS)*, 2022.

[25] E. Bondi, D. Dey, A. Kapoor, J. Piavis, S. Shah, F. Fang, B. Dilkina, R. Hannaford, A. Iyer, L. Joppa, and M. Tambe, "AirSim-W: A simulation environment for wildlife conservation with UAVs," in *Proceedings of the ACM SIGCAS Conference on Computing and Sustainable Societies*, 2018.

[26] L. Laux, S. Schirmer, S. Schopferer, and J. C. Dauer, "Build your own training data - synthetic data for object detection in aerial images," in *Proceedings of the Workshop on Avionics Systems and Software Engineering*, 2022.

[27] F. X. Viana, G. M. Araujo, M. F. Pinto, J. Colares, and D. B. Haddad, "Aerial image instance segmentation through synthetic data using deep learning," *Learn. Nonlinear Model*, vol. 18, 2020.

[28] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[29] A. Shafaei, J. J. Little, and M. Schmidt, "Play and learn: Using video games to train computer vision models," *arXiv preprint arXiv:1608.01745*, 2016.

[30] F. S. Saleh, M. S. Aliakbarian, M. Salzmann, L. Petersson, and J. M. Alvarez, "Effective use of synthetic data for urban scene semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[31] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[32] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison, "SceneNet RGB-D: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation?" in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[33] B.-C.-Z. Blaga and S. Nedevschi, "Semantic segmentation learning for autonomous uavs using simulators and real data," in *Proceedings of the IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2019.

[34] S. Beery, Y. Liu, D. Morris, J. Piavis, A. Kapoor, N. Joshi, M. Meister, and P. Perona, "Synthetic examples improve generalization for rare classes," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[35] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[36] Y. Movshovitz-Attias, T. Kanade, and Y. Sheikh, "How useful is photo-realistic rendering for visual learning?" in *Proceedings of the Computer Vision–ECCV 2016 Workshops*, 2016.

[37] Epic Games, Inc. The most powerful real-time 3d creation tool - unreal engine. Accessed 08 Febuary 2025. [Online]. Available: https://www.unrealengine.com/en-US

[38] Unity Technologies. Unity real-time development platform. Accessed 08 Febuary 2025. [Online]. Available: https://www.unity.com/

[39] Rockstar Games, Inc. Grand theft auto v - rockstar games. Accessed 09 May 2025. [Online]. Available: https://www.rockstargames.com/gta-v

[40] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the Annual Conference on Robot Learning*, 2017.

[41] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," *arXiv preprint arXiv:1705.05065v2*, 2017.

[42] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, "Sim4CV: A photo-realistic simulator for computer vision applications," *International Journal of Computer Vision*, vol. 126, 2018.

[43] W. Qiu, F. Zhong, Y. Zhang, S. Qiao, Z. Xiao, T. S. Kim, and Y. Wang, "UnrealCV: Virtual worlds for computer vision," in *Proceedings of the ACM International Conference on Multimedia*, 2017.

[44] F. C. Akyon, O. Eryuksel, K. A. Ozfuttu, and S. O. Altinuc, "Track boosting and synthetic data aided drone detection," in *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2021.

[45] F. Reway, A. Hoffmann, D. Wachtel, W. Huber, A. Knoll, and E. Ribeiro, "Test method for measuring the simulation-to-reality gap of camera-based object detection algorithms for autonomous driving," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 2020.

[46] A. Barisic, F. Petric, and S. Bogdan, "Sim2Air - synthetic aerial dataset for uav monitoring," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, 2022.

[47] F. E. Nowruzi, P. Kapoor, D. Kolhatkar, F. A. Hassanat, R. Laganiere, and J. Rebut, "How much real data do we actually need: Analyzing object

detection performance using synthetic and real data," *arXiv preprint arXiv:1907.07061v1*, 2019.

[48] J. Quinonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset shift in machine learning*. MIT Press, 2008.

[49] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognition*, vol. 45, no. 1, 2012.

[50] Z. Liu, T. Lian, J. Farrell, and B. A. Wandell, "Neural network generalization: The impact of camera parameters," *IEEE Access*, vol. 8, 2020.

[51] A. Carlson, K. A. Skinner, R. Vasudevan, and M. Johnson-Roberson, "Modeling camera effects to improve visual learning from synthetic data," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018.

[52] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, corresponding image dataset is available at https://www.cvlibs.net/datasets/kitti/index.php. Accessed 7 October 2024.

[53] P. Zhu, L. Wen, D. Du, X. Bian, H. Fan, Q. Hu, and H. Ling, "Vision meets drones: Past, present and future," *arXiv preprint arXiv:2001.06303*, 2020.

[54] I. Bozcan and E. Kayacan, "AU-AIR: A multi-modal unmanned aerial vehicle dataset for low altitude traffic surveillance," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[55] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, *Mathematics for Machine Learning*. Cambridge University Press, 2020.

[56] A. Kar, A. Prakash, M.-Y. Liu, E. Cameracci, J. Yuan, M. Rusiniak,

D. Acuna, A. Torralba, and S. Fidler, "Meta-Sim: Learning to generate synthetic datasets," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[57] A. E. Ceron-Lopez, R. Ranjan, and N. Koganti, "Realism assessment for synthetic images in robot vision through performance characterization," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[58] M. Krump and P. Stütz, "UAV based vehicle detection with synthetic training: Identification of performance factors using image descriptors and machine learning," in *Proceedings of the International Conference on Modelling and Simulation for Autonomous Systems (MESAS)*, 2020.

[59] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, 2013.

[60] K. Hagn and O. Grau, "Improved sensor model for realistic synthetic data generation," in *Proceedings of the ACM Computer Science in Cars Symposium (CSCS)*, 2021.

[61] K. Saad and S.-A. Schneider, "Camera vignetting model and its effects on deep neural networks for object detection," in *Proceedings of the IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, 2019.

[62] F. A. Ruis, A. M. Liezenga, F. G. Heslinga, L. Ballan, T. A. Eker, R. J. den Hollander, M. C. van Leeuwen, J. Dijk, and W. Huizinga, "Improving object detector training on synthetic data by starting with a strong baseline methodology," in *Synthetic Data for Artificial Intelligence and Machine Learning: Tools, Techniques, and Applications II*, vol. 13035, 2024.

[63] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang, and Q. Tian, "The unmanned aerial vehicle benchmark: Object detection

and tracking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[64] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[65] H. Hoyez, C. Schockaert, J. Rambach, B. Mirbach, and D. Stricker, "Unsupervised image-to-image translation: A review," *Sensors*, vol. 22, no. 21, 2022.

[66] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, 2019.

[67] X. Wu, D. Sahoo, and S. C. Hoi, "Recent advances in deep learning for object detection," *Neurocomputing*, vol. 396, 2020.

[68] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[69] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy, "Do vision transformers see like convolutional neural networks?" *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[70] N. Park and S. Kim, "How do vision transformers work?" *arXiv preprint arXiv:2202.06709*, 2022.

[71] M. M. Naseer, K. Ranasinghe, S. H. Khan, M. Hayat, F. Shahbaz Khan, and M.-H. Yang, "Intriguing properties of vision transformers," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[72] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increas-

ing shape bias improves accuracy and robustness," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[73] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, 2009.

[74] A. Arnold, R. Nallapati, and W. W. Cohen, "A comparative study of methods for transductive transfer learning," in *Proceedings of the IEEE International Conference on Data Mining Workshops (ICDMW)*, 2007.

[75] G. Csurka, R. Volpi, and B. Chidlovskii, "Unsupervised domain adaptation for semantic image segmentation: a comprehensive survey," *arXiv preprint arXiv:2112.03241*, 2021.

[76] S. Zhao, X. Yue, S. Zhang, B. Li, H. Zhao, B. Wu, R. Krishna, J. E. Gonzalez, A. L. Sangiovanni-Vincentelli, S. A. Seshia *et al.*, "A review of single-source deep unsupervised visual domain adaptation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, 2020.

[77] M. Toldo, A. Maracani, U. Michieli, and P. Zanuttigh, "Unsupervised domain adaptation in semantic segmentation: A review," *Technologies*, vol. 8, no. 2, 2020.

[78] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[79] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[80] A. Prakash, S. Boochoon, M. Brophy, D. Acuna, E. Cameracci, G. State, O. Shapira, and S. Birchfield, "Structured domain randomization: Bridg-

ing the reality gap by context-aware synthetic data," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2019.

[81] X. Yue, Y. Zhang, S. Zhao, A. Sangiovanni-Vincentelli, K. Keutzer, and B. Gong, "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 (ICCV)*, 2019.

[82] J. Huang, D. Guan, A. Xiao, and S. Lu, "Fsdr: Frequency space domain randomization for domain generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[83] European Union Aviation Safety Agency (EASA), "EASA Artificial Intelligence (AI) Concept Paper Issue 2: Guidance for Level 1&2 machine learning applications," 2024.

[84] Wehrtechnische Dienststelle für Luftfahrzeuge und Luftfahrtgerät der Bundeswehr (WTD 61), images of air-to-air refueling kindly provided to German Aerospace Center (DLR) for research purposes, unpublished.

[85] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: fast and flexible image augmentations," *Information*, vol. 11, no. 2, 2020.

[86] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, 2019.

[87] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo *et al.*, "The many faces of robustness: A critical analysis of out-of-distribution generalization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[88] J. Rüter and R. Schmidt, "Using only synthetic images to train a drogue detector for aerial refueling," in *Proceedings of the International Confer-*

ence on Modelling and Simulation for Autonomous Systems (MESAS), 2023.

[89] C. Hinniger and J. Rüter, "Synthetic training data for semantic segmentation of the environment from uav perspective," *Aerospace*, vol. 10, no. 7, 2023.

[90] J. Rüter, U. Durak, and J. C. Dauer, "Investigating the sim-to-real generalizability of deep learning object detection models," *Journal of Imaging*, vol. 10, no. 10, 2024.

[91] J. Rüter, J. C. Dauer, and U. Durak, "There is no model to beat them all: Recommendations for deep learning model selection when training on synthetic images," in *Proceedings of the International Conference on Image Analysis and Processing (ICIAP)*, 2025.

[92] J. Rüter, U. Durak, and J. C. Dauer, "Investigating the influence of image augmentations on the sim-to-real generalization of deep learning perception models," in *Proceedings of the Synthetic Data for Computer Vision Workshop @CVPR*, 2025.

[93] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[94] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[95] ——, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[96] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, 2009.

[97] M. Everingham and J. Winn. (2012) The pascal visual object classes challenge 2012 (voc2012) development kit. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/devkit_doc.pdf

[98] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, 2015.

[99] R. Padilla, W. L. Passos, T. L. Dias, S. L. Netto, and E. A. Da Silva, "A comparative analysis of object detection metrics with a companion open-source toolkit," *Electronics*, vol. 10, no. 3, 2021.

[100] R. Szeliski, *Computer vision: algorithms and applications*, 2nd ed. Springer Nature, 2022.

[101] A. Rosebrock. (2016, 11) A visual equation for intersection over union (jaccard index). Published under CC BY-SA 4.0, original source: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/, accessed 7 March 2023. [Online]. Available: https://en.wikipedia.org/wiki/File:Intersection_over_Union_-_visual_equation.png

[102] ——. (2016, 11) Iou for bounding boxes showing a poor, good and excellent prediction. Published under CC BY-SA 4.0, original source: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/, accessed 7 March 2023. [Online]. Available: https://en.wikipedia.org/wiki/File:Intersection_over_Union_-_poor,_good_and_excellent_score.png

[103] TorchVision maintainers and contributors, "TorchVision: Pytorch's computer vision library," https://github.com/pytorch/vision, 2016.

[104] MMSegmentation Contributors, "MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark," https://github.com/open-mmlab/mmsegmentation, 2020.

[105] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[106] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[107] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[108] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[109] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[110] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[111] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[112] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, Z. Zhang, Y. Sun, T. He, J. Mueller, R. Manmatha *et al.*, "ResNeSt: Split-attention networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[113] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[114] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[115] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for MobileNetV3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[116] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[117] X. Chu, Z. Tian, Y. Wang, B. Zhang, H. Ren, X. Wei, H. Xia, and C. Shen, "Twins: Revisiting the design of spatial attention in vision transformers," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[118] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[119] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[120] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[121] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[122] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks

for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[123] L.-C. Chen, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[124] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[125] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2010.

[126] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[127] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[128] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[129] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[130] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[131] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze, and H. Adam, "Netadapt: Platform-aware neural network adaptation for mobile applications," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[132] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[133] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[134] X. Chu, Z. Tian, B. Zhang, X. Wang, and C. Shen, "Conditional positional encodings for vision transformers," *arXiv preprint arXiv:2102.10882*, 2021.

[135] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[136] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[137] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[138] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015.

[139] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, 2013.

[140] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[141] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[142] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," *Advances in Neural Information Processing Systems*, vol. 24, 2011.

[143] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, 2017.

[144] Albumentations contributors. Blur transforms. Accessed 22 March 2024. [Online]. Available: https://albumentations.ai/docs/api_reference/augmentations/blur/transforms/

[145] ——. Transforms. Accessed 22 March 2024. [Online]. Available: https://albumentations.ai/docs/api_reference/augmentations/transforms/

[146] Y. Cabon, N. Murray, and M. Humenberger, "Virtual kitti 2," *arXiv preprint arXiv:2001.10773*, 2020.

[147] The Joint Air Power Competence Centre (JAPCC), "Air-to-air refuelling consolidation - an update," 2014.

[148] North Atlantic Treaty Organization (NATO), "NATO Standard ATP-3.3.4.2 Air-to-Air Refueling Edition D Version 1," 2019.

[149] Christian Turner USAF. (2019, 4) KC-46 Refuels B-2 over Edwards AFB.

Accessed 18 July 2023. [Online]. Available: https://en.wikipedia.org/wiki/File:KC-46_Refuels_B-2_over_Edwards_AFB.jpg

[150] P. R. Thomas, U. Bhandari, S. Bullock, T. S. Richardson, and J. L. Du Bois, "Advances in air to air refuelling," *Progress in Aerospace Sciences*, vol. 71, 2014.

[151] North Atlantic Treaty Organization (NATO), "NATO Standard ATP-3.3.4.10 Automated Air-to-Air Refueling (A3R) Edition A Version 1," 2021.

[152] M. Finstad and Github Contributors, "Losslesscut," https://github.com/mifi/lossless-cut.

[153] Ffmpeg. Accessed 18 March 2025. [Online]. Available: https://www.ffmpeg.org/

[154] A. Sanders, *An introduction to Unreal Engine 4.* CRC Press, 2016.

[155] Epic Games, Inc. Game engine - build multi-platform video games - unreal engine. Accessed 13 July 2023. [Online]. Available: https://www.unrealengine.com/en-US/solutions/games

[156] J. Drake. (2023, 03) 24 great games that use the unreal 4 game engine. Accessed 13 July 2023. [Online]. Available: https://www.thegamer.com/great-games-use-unreal-4-game-engine/

[157] Epic Games, Inc. Advanced automotive and car design software and visualization - unreal engine. Accessed 13 July 2023. [Online]. Available: https://www.unrealengine.com/de/solutions/automotive-transportation

[158] ——. Architecture design software and 3d rendering visualization engine - unreal engine. Accessed 13 July 2023. [Online]. Available: https://www.unrealengine.com/en-US/solutions/architecture

[159] ——. Unreal engine marketplace is now fab - ue marketplace. Accessed

19 March 2025. [Online]. Available: https://www.unrealengine.com/marketplace/en-US/store

[160] ——. Fab. Accessed 19 March 2025. [Online]. Available: https://www.fab.com/

[161] ——. Volumetric clouds | unreal engine document. Accessed 13 July 2023. [Online]. Available: https://docs.unrealengine.com/4.26/en-US/BuildingWorlds/LightingAndShadows/VolumetricClouds/

[162] Cesium GS, Inc. Cesium ion - cesium. Accessed 17 July 2023. [Online]. Available: https://cesium.com/platform/cesium-ion/

[163] ——. Cesium for unreal - cesium. Accessed 17 July 2023. [Online]. Available: https://cesium.com/platform/cesium-for-unreal/

[164] Epic Games, Inc. Cesium for unreal in code plugins - UE Marketplace. Accessed 17 July 2023. [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/87b0d05800a545d49bf858ef3458c4f7

[165] Cesium GS, Inc. Cesium world terrain - cesium. Accessed 17 July 2023. [Online]. Available: https://cesium.com/platform/cesium-ion/content/cesium-world-terrain/

[166] ——. Bing maps aerial imagery. Accessed 17 July 2023. [Online]. Available: https://cesium.com/platform/cesium-ion/content/bing-maps-imagery/

[167] ——. Using a geospatially accurate sun - cesium. Accessed 17 July 2023. [Online]. Available: https://cesium.com/learn/unreal/unreal-geospatially-accurate-sun/

[168] Hum3D.com, a ParaART, LLC company. General dynamics f-16c block 52 3d model - aircraft on hum3d. Accessed 17 July 2023. [Online]. Available: https://hum3d.com/3d-models/general-dynamics-f-16c-block-52/

[169] Joint Air Power Competence Centre (JAPCC), "ATP-3.3.4.2 National Standards Related Document (SDR) Germany," 2022.

[170] A. Marcu, V. Licaret, D. Costea, and M. Leordeanu, "Semantics through time: Semi-supervised segmentation of aerial videos with iterative label propagation," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2020, corresponding image dataset is available at https://sites.google.com/site/aerialimageunderstanding/semantics-through-time-semi-supervised-segmentation-of-aerial-videos. Accessed 5 March 2022.

[171] I. Krešo, D. Čaušević, J. Krapac, and S. Šegvić, "Convolutional scale invariance for semantic segmentation," in *Proceedings of the German Conference on Pattern Recognition (GCPR)*, 2016.

[172] G. Ros, S. Ramos, M. Granados, A. Bakhtiary, D. Vazquez, and A. M. Lopez, "Vision-based offline-online perception paradigm for autonomous driving," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2015.

[173] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, 2015.

[174] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ADE20k dataset," *International Journal of Computer Vision*, vol. 127, 2019.

[175] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2017.

[176] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2019.

[177] T. Dozat, "Incorporating nesterov momentum into adam," in *Proceed-*

ings of the International Conference on Learning Representations (ICLR), 2016.

[178] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu *et al.*, "A survey on vision transformer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, 2022.

[179] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "CvT: Introducing convolutions to vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[180] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," *arXiv preprint arXiv:1705.08292*, 2018.

Additional Notes

Note on Contributions in Previous Publications

First presentations of some results of this work were published before this dissertation. For the relevant parts of these publications, the co-authors had the following contributions:

[88] J. Rüter and R. Schmidt, "Using only synthetic images to train a drogue detector for aerial refueling," in Proceedings of the International Conference on Modelling and Simulation for Autonomous Systems (MESAS), 2023:

> R. Schmidt supported implementing the toolchain used for the data generation in this work and created drafts of some figures during her internship at DLR under the supervision of J. Rüter.

[89] C. Hinniger and J. Rüter, "Synthetic training data for semantic segmentation of the environment from uav perspective," Aerospace, vol. 10, no. 7, 2023:

> C. Hinniger created the simulation environment and extracted the synthetic images used in this work during his master thesis at DLR under the supervision of J. Rüter.

[90] J. Rüter, U. Durak, and J. C. Dauer, "Investigating the sim-to-real generalizability of deep learning object detection models," Journal of Imaging, vol. 10, no. 10, 2024:

> U. Durak and J. C. Dauer supervised and reviewed the research conducted by J. Rüter.

[91] J. Rüter, J. C. Dauer, and U. Durak, "There is no model to beat them all: Recommendations for deep learning model selection when training on synthetic images," in Proceedings of the International Conference on Image Analysis and Processing (ICIAP), 2025:

J. C. Dauer and U. Durak supervised and reviewed the research conducted by J. Rüter.

[92] J. Rüter, U. Durak, and J. C. Dauer, "Investigating the influence of image augmentations on the sim-to-real generalization of deep learning perception models," in Proceedings of the Synthetic Data for Computer Vision Workshop at CVPR, 2025:

U. Durak and J. C. Dauer supervised and reviewed the research conducted by J. Rüter.

Utilization of AI Tools

Language refinements of this work were supported by the large language models *gpt-oss:20b*[1] and *llama3.3:latest*[2] as provided by DLR Ollama service.

[1] `https://ollama.com/library/gpt-oss`, last accessed 25 October 2025.
[2] `https://ollama.com/library/llama3.3`, last accessed 25 October 2025.