IAS-Forschungsberichte

Forschungsbericht
Institut für Automatisierungstechnik und Softwaresysteme

Hrsg.: Prof. Dr.-Ing. Dr. h. c. Michael Weyrich

Hannes Vietz

Generierung synthetischer Trainingsdaten zur Leistungssteigerung bei maschinellen Lernaufgaben in der Automatisierungstechnik

Band 3/2026

Universität Stuttgart

Generierung synthetischer Trainingsdaten zur Leistungssteigerung bei maschinellen Lernaufgaben in der Automatisierungstechnik

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

Vorgelegt von
Hannes Vietz
aus Grafenau

Hauptberichter:	Prof. Dr.-Ing. Dr. h. c. Michael Weyrich
Mitberichter:	Prof. Dr.-Ing. Mike Barth
Tag der Einreichung:	17.12.2024
Tag der mündlichen Prüfung:	08.08.2025

Institut für Automatisierungstechnik und Softwaresysteme
der Universität Stuttgart

2024

IAS-Forschungsberichte

Band 3/2026

Hannes Vietz

Generierung synthetischer Trainingsdaten zur Leistungssteigerung bei maschinellen Lernaufgaben in der Automatisierungstechnik

D 93 (Diss. Universität Stuttgart)

Shaker Verlag
Düren 2026

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Zugl.: Stuttgart, Univ., Diss., 2025

Copyright Shaker Verlag 2026
Alle Rechte, auch das des auszugsweisen Nachdruckes, der auszugsweisen oder vollständigen Wiedergabe, der Speicherung in Datenverarbeitungsanlagen und der Übersetzung, vorbehalten.

Printed in Germany.

Print-ISBN 978-3-8191-0761-0
PDF-ISBN 978-3-8191-0698-9
ISSN 1610-4781
eISSN 2944-8417

Shaker Verlag GmbH • Am Langen Graben 15a • 52353 Düren
Telefon: 02421 / 99 0 11 - 0 • Telefax: 02421 / 99 0 11 - 9
Internet: www.shaker.de • E-Mail: info@shaker.de

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Automatisierungstechnik und Softwaresysteme (IAS) der Universität Stuttgart.

Mein Dank gilt meinem Doktorvater, Institutsleiter Herrn Prof. Dr.-Ing. Dr. h.c. Michael Weyrich, für die Übernahme des Hauptberichts sowie die vielen wertvollen Diskussionen, Hinweise und Anregungen.

Herrn Prof. Dr.-Ing. Mike Barth danke ich für das Interesse an meiner Arbeit und die Übernahme des Mitberichts.

Allen Kolleg:innen am IAS gilt mein herzlichster Dank für die fortwährende Unterstützung, die sehr gute Zusammenarbeit und die vielen wertvollen und konstruktiven Diskussionen. Ein besonderer Dank gilt Herrn Dr.-Ing. Nasser Jazdi für seine fortwährende Unterstützung während der Arbeit und die vielen wertvollen Gespräche.

Ebenso möchte ich den zahlreichen Studierenden danken, die im Rahmen ihrer Abschlussarbeiten einen wesentlichen Beitrag zur technischen Umsetzung des Konzepts geleistet haben. Besonders hervorheben möchte ich dabei Tom Dengler, Johannes Sigel und Sebastian Baum.

Mein besonderer Dank gilt auch meiner Partnerin Stephanie, meinen Freund:innen und meiner Familie, die mich stets mit aufbauender Unterstützung begleitet und mir geholfen haben, nie aufzugeben. Zuletzt möchte ich meiner Mutter danken, die sich während unserer gemeinsamen Laufrunden geduldig jeden meiner Vorträge angehört und immer wieder wertvolle, kritische Fragen gestellt hat.

Stuttgart, im Dezember 2024

Hannes Vietz

Widmung

Diese Arbeit widme ich meiner Mutter Hilde Vietz.

Inhaltsverzeichnis

Abbildungsverzeichnis ... **iii**
Tabellenverzeichnis ... **v**
Abkürzungsverzeichnis ... **vi**
Begriffsverzeichnis ... **viii**
Zusammenfassung ... **x**
Abstract ... **xi**
1 Einleitung ... **1**
1.1 Motivation ... 1
1.2 Problemstellung und Herausforderungen ... 2
1.3 Zielsetzung der Arbeit ... 4
1.4 Aufbau und Methodik der Arbeit ... 7
2 Stand der Wissenschaft und Technik ... **9**
2.1 Trainingsdatengenerierung durch Augmentierung ... 9
2.1.1 Eingabedatenaugmentierung ... 9
2.1.2 Featureraumaugmentierung ... 11
2.1.3 Adversariale Augmentierung ... 12
2.2 Simulative Methoden zur Trainingsdatengenerierung ... 14
2.2.1 Umgebungssimulationen ... 14
2.2.2 Simulation von Bilddaten ... 16
2.2.3 Simulation von Funkdaten ... 18
2.3 Trainingsdatengenerierung durch generative Modelle ... 19
2.3.1 Generative Modelle ... 19
2.3.2 Modell-agnostische Trainingsdatengenerierung ... 25
2.3.3 Modell-informierte Trainingsdatengenerierung ... 27
2.4 Bewertung der bestehenden Ansätze ... 30
2.4.1 Bewertungskriterien ... 30
2.4.2 Durchführung der Bewertung ... 32
2.5 Aufzeigen des Forschungsbedarfs ... 35
3 Konzept zur leistungssteigernden Trainingsdatengenerierung durch generative Modelle ... **38**
3.1 Motivation der Konzeptidee ... 38

3.2 Überblick über das Gesamtkonzept ... 39
3.3 Entwurf des leistungssteigernden Trainingsdatengenerators ... 41
3.3.1 Generatives neuronales Netz ... 41
3.3.2 Ergebnisüberprüfungskomponente ... 43
3.3.3 Steuerungskomponente ... 46
3.4 Erörterung der Zielerreichung: Prüfung der Eignung des Konzepts zur Erfüllung der Teilziele ... 48
4 Prototypische Realisierungen und Evaluierung des Ansatzes ... 50
4.1 Vorgehen bei der Realisierung ... 50
4.2 Evaluierungsfall 1: Handschrifterkennung ... 52
4.2.1 Anwendungsfall ... 52
4.2.2 Realisierung ... 53
4.2.3 Evaluierung ... 57
4.3 Evaluierungsfall 2: 5G-Positionsbestimmung ... 69
4.3.1 Anwendungsfall ... 69
4.3.2 Realisierung ... 72
4.3.3 Evaluierung ... 75
4.4 Evaluierungsfall 3: Industrieller Objektdetektor ... 80
4.4.1 Anwendungsfall ... 80
4.4.2 Realisierung ... 82
4.4.3 Evaluierung ... 87
4.5 Diskussion der Evaluierungsergebnisse ... 93
4.5.1 Bewertung der Evaluierungsergebnisse ... 93
4.5.2 Anwendbarkeit des Verfahrens in der Automatisierungstechnik ... 95
5 Schlussbetrachtung ... 97
5.1 Zusammenfassung der Ergebnisse ... 97
5.2 Ausblick auf weiterführende Forschungsaktivitäten ... 100
6 Literaturverzeichnis ... 102

Abbildungsverzeichnis

Abbildung 1.1: Aufbau der Arbeit 7
Abbildung 2.1: Das Originalbild links wurde durch die semantische Augmentierungsmethode unter der Annahme erstellt, dass die Lackfarbe des PKW keine Bedeutung für die Detektion haben sollte. 11
Abbildung 2.2: Grundstruktur eines Generative Adversarial Networks 20
Abbildung 3.1: Das Konzept zur leistungssteigernden und automatisierten Trainingsdatengenerierung für das maschinelle Lernen im Gesamten dargestellt…….. 39
Abbildung 3.2: Aufbau des leistungssteigernden, automatisierten Trainingsdatengenerators 41
Abbildung 3.3: Ablaufdiagramm der Ergebnisüberprüfungskomponente. Kern der Komponente sind die drei nummerierten Entscheidungen, die den Programmablauf kontrollieren und beenden. 43
Abbildung 3.4: Beispiel eines divergierenden Gradientenabstiegs, der auf Grund der großen Schrittweite das Minimum nie finden kann. 45
Abbildung 4.1: Übersicht der Evaluierungsszenarien in ansteigender Komplexität. 50
Abbildung 4.2: Übersicht über die Realisierung des Anwendungsfalls 1 53
Abbildung 4.3: Die realisierte CGAN-Architektur bestehend aus Generator und Discriminator, die durch Convolution-Operationen umgesetzt werden. 53
Abbildung 4.4: Verlauf des Trainingslosses von Generator und Discriminator. 54
Abbildung 4.5: Experiment zum Vergleich der Abhängigkeit von ϵ bei FGSM und der Latentvektoranpassung 56
Abbildung 4.6: MNIST-Bilder, die durch den Datengenerator generiert wurden. Die gezielt generierten Daten führen zu einer Klassifizierungsänderung und steigern die Modellleistung. 57
Abbildung 4.7: Position der Pixeländerung erzeugt durch den FGSM-Ansatz (links) und die Latentvektoranpassung (rechts). 59
Abbildung 4.8: 3D-Ansicht auf die Pixeländerungen FGSM (links) und die Latentvektoranpassung (rechts). 60
Abbildung 4.9: Transformation der 1 durch die Latentvektoranpassung 60
Abbildung 4.10: Transformation der Klassen "6" und "9" durch die Latentvektoranpassung 61
Abbildung 4.11: Unterschied zwischen Trainingsdaten und generierten Daten von FGSM (links) und Latentvektoranpassung (rechts) 62
Abbildung 4.12: Verlauf der Zusammensetzung des Trainingsdatensatzes über das Konvergenztraining hinweg. 64
Abbildung 4.13: Verlauf der Klassifizierungsgenauigkeit des neuronalen Netzes über das Konvergenztraining hinweg. 65

Abbildung 4.14: Neuer MNIST-ähnlicher Datensatz. Links Originalaufnahme auf Papier und rechts digitalisierte Bilder.....66
Abbildung 4.15: Klassifizierungsergebnis auf dem neu erstellten Datensatz. Die farbigen Ränder zeigen, von welchem Modell das Bild korrekt klassifiziert wurde. 67
Abbildung 4.16: Fotografie eines AGV während es zur Datenerfassung durch den Shopfloor der Arena2036 fährt.....71
Abbildung 4.17: Übersicht der Realisierung des Trainingsdatengenerators für den 2. Anwendungsfall. Das gelbe Bild ist eine Visualisierung der vorverarbeiteten Kanalantwort der Antennen.....72
Abbildung 4.18: Histogramm der Positionierungsgenauigkeiten des Positionierungsmodells im 5G-Testbed. In dem rot umrandeten Bereich ist die Genauigkeit besonders schlecht und soll durch synthetische Daten verbessert werden.....76
Abbildung 4.19: Histogramme der Positionsfehler. Training nur mit realen Daten (links). Training mit realen und generierten Daten (rechts). 77
Abbildung 4.20: Histogramm aller Positionierungsfehler, unabhängig von ihren Koordinaten. Das Histogramm wird nur für Fehler bis 4,0 m dargestellt, da die Bins für höhere Fehler aufgrund ihrer geringen Höhe nicht angezeigt werden können.78
Abbildung 4.21: Verteilungsfunktion (cumulative distribution function) der Positionierungsgenauigkeiten. Die Wahrscheinlichkeit eines Positionierungsfehlers von unter 50 cm ist durch die Verwendung der generierten Daten um 5 % gestiegen. 79
Abbildung 4.22: Transportbox, die durch das AGV in der Arena2036 erkannt werden soll. 80
Abbildung 4.23: Bilder der Transportbox, die für das Training aufgenommen wurden. 81
Abbildung 4.24: Übersicht über die Realisierung des Anwendungsfalls 3 82
Abbildung 4.25: Backward Guidance der Layout Guidance 85
Abbildung 4.26: Vereinfachte Darstellung des verwendeten Foundation-Modells Stable Diffusion mit dem Fokus auf die latenten z-Vektoren. 86
Abbildung 4.27: Beispiele der synthetischen Daten, die mit Dreambooth und Layout Guidance durch Stable Diffusion generiert wurden. 88
Abbildung 4.28: Synthetische Bildpaare. Das jeweils linke Bild zeigt ein Bild, das mit zufälligem z-Vektor generiert wurde. Das rechte Bild zeigt das darauf basierende, durch die Steuerungskomponente angepasste, Bild. 90

Tabellenverzeichnis

Tabelle 2.1: Übersicht über die Bewertung von Ansätzen zur Generierung von Trainingsdaten 34
Tabelle 4.1: Überblick über die Hyperparameter in Evaluierungsfall 1 54
Tabelle 4.2: Schwellwerte der Binarisierung für die Digitalisierung 66
Tabelle 4.3: Vergleich der Objektdetektoren mit dem mAP50-Score 92
Tabelle 4.4: Vergleich der Objektdetektoren mit dem mAP50:95-Score 92

Abkürzungsverzeichnis

CARLA	Car Learning to Act
VIVID	VIrtual environment for VIsual Deep learning
TORCS	The Open Racing Car Simulator
GTA	Grand Theft Auto
PreSil	Precise Synthetic Image and LiDAR
BS	Basisstation
VAE	Variational Autoencoder
FGSM	Fast Gradient Sign Method
GAN	Generative Adversarial Network
DCGAN	Convolutional Generative Adversarial Network
CNN	Convolutional Neural Network
WGAN	Wasserstein Generative Adversarial Network
JS	Jensen-Shannon-Divergenz
KL	Kullback-Leibler-Divergenz
CGAN	Conditional Generative Adversarial Network
SVM	Support Vector Machine
DDPM	Denoising Diffusion Probabilistic Model
AMC	Automatic Modulation Classification
SDGAN	Surface Defect-Generation Adversarial Network
SAR	Synthetic Aperture Radar
FID	Fréchet Inception Distance
RTLS	Echtzeit-Ortungssysteme
AGV	Automated Guided Vehicle

SRS	Sondierungsreferenzsignal
FFT	schnelle Fourier-Transformation
mAP	mean Average Precision
AP	Average Precision
IoU	Intersection over Union

Begriffsverzeichnis

5G: Bezeichnet die fünfte Mobilfunkgeneration und ist damit direkter Nachfolger von 4G. Es verspricht ein schnelleres mobiles Netz und eine wachsende Zahl vernetzter Geräte im alltäglichen und industriellen Umfeld

Adversariale Daten: Manipulierte Eingaben, die gezielt darauf ausgelegt sind, maschinelle Lernmodelle zu täuschen oder deren Leistung zu beeinträchtigen, wobei sie einem Menschen nicht auffallen

Augmentierung: Die Erweiterung eines Datensatzes durch künstliche Veränderung von Datenpunkten, um die Vielfalt und Menge für das Training maschineller Lernmodelle zu erhöhen

Conditional Generative Adversarial Network: Eine Sonderform des GAN, die in der Lage ist, Daten für ein gegebenes Label zu generieren

Convolutional Generative Adversarial Network: Eine Sonderform des GAN, die aus Convolutional-Schichten besteht

Diskriminierende Modelle: Maschinell trainierte Modelle, die als Ausgabe eine Entscheidung treffen, im Gegensatz zu den generativen Modellen

Feature: Individuelle messbare Eigenschaft oder ein Merkmal, das als Eingabevariable für Modelle im Bereich maschinelles Lernen und Datenanalyse verwendet wird. Deep-Learning-Modelle lernen die Merkmale selbstständig

Featureraum: Mathematischer Raum, der alle möglichen Merkmalswerte abdeckt

Generative Adversarial Network: Klasse von maschinellen Lernalgorithmen, bei denen zwei neuronale Netzwerke – ein Generator und ein Diskriminator – gegeneinander antreten, um neue, synthetische Daten zu erzeugen, die von echten Daten nicht zu unterscheiden sind

Generative Modelle: Algorithmen, die darauf abzielen, die Wahrscheinlichkeitsverteilung von Datensätzen zu lernen, um neue Datenpunkte zu erzeugen

Hard-Negative-Mining: Verfahren, bei dem aktiv schwierig zu klassifizierende negative Beispiele während des Trainings gesucht werden, um die Leistung eines Modells zu verbessern

Label: Eine Kategorie oder ein Wert, der einem Datenpunkt in einem überwachten Lernkontext zugewiesen wird.

Labelingprozess: Zuweisen von Labels zu Datenpunkten

Modell-informiert: Ansätze, die Zugriff auf ein existierendes Modell haben

Overfitting: Ein Szenario, in dem ein maschinelles Lernmodell zu sehr an die Trainingsdaten angepasst ist und dadurch seine Generalisierungsfähigkeit auf neue Daten verliert

Regularisierung: Technik, um Overfitting zu vermeiden, indem die Komplexität eines Modells beschränkt wird

Synthetische Daten: Künstlich erzeugte Daten, die reale Datensätze imitieren oder ergänzen

Transformer: Typ eines tiefen Lernmodells, das auf Selbst-Attention-Mechanismen beruht

Zusammenfassung

Die Anwendung maschinell gelernter Algorithmen gewinnt in der Automatisierungstechnik zunehmend an Bedeutung, da sie datenbasierte Problemlösungen ermöglicht. Ein wesentlicher Engpass für den Einsatz leistungsfähiger Deep-Learning-Modelle im Feldeinsatz ist jedoch die Verfügbarkeit umfangreicher, gelabelter Trainingsdaten, deren Akquisition in der industriellen Praxis oft hohe Kosten und Betriebsanpassungen erfordert. Ein vielversprechender Ansatz zur Lösung dieses Problems liegt in der Nutzung synthetischer Daten, die das Training ohne den Zugang zu großen realen Datensätzen ermöglichen. Jedoch erzeugen derzeitige Verfahren häufig redundante Szenarien, und es bleibt unklar, welche spezifischen synthetischen Daten zur Verbesserung der Modellgüte beitragen.

In dieser Dissertation wird ein Konzept zur leistungssteigernden Generierung synthetischer Daten entwickelt, das spezifisch darauf ausgerichtet ist, bestehende Schwächen in trainierten Modellen zu adressieren und deren Leistungsmetriken gezielt zu verbessern. Das iterative Konzept umfasst ein generatives neuronales Netz, eine Steuerungskomponente zur Optimierung der Datengenerierung sowie eine Entscheidungslogik, die bestimmt, ob ein generierter Datenpunkt für das Training verwendet werden sollte.

Die Effektivität des Konzepts wird anhand von Anwendungsfällen aus der Automatisierungstechnik demonstriert: Der MNIST-Datensatz zur Handschrifterkennung dient als kontrollierte Umgebung zur Bewertung der Fähigkeit des Generators, realitätsnahe, interpretierbare Bilddaten zu erzeugen; ein datengetriebenes 5G-Positionsbestimmungssystem wird genutzt, um die Übertragbarkeit des Konzepts auf eine industrielle Produktionsumgebung zu belegen, wobei eine signifikante Verbesserung der Modellgenauigkeit und Generalisierung gezeigt wird; der dritte Anwendungsfall untersucht die optische Objektdetektion in einer Industrieumgebung ohne verfügbare öffentliche Datensätze und zeigt das Potenzial des Konzepts, spezifische industrielle Anforderungen zu adressieren.

Die Evaluierung zeigt, dass das vorgestellte Konzept in der Lage ist, realistische und gezielte Trainingsdaten zu generieren, die die Robustheit und Leistungsfähigkeit neuronaler Netze im Kontext maschinellen Lernens und industrieller Automatisierung signifikant erhöhen können. In allen Anwendungsfällen führte der Einsatz der synthetisch generierten Daten zu einer substantiellen Verbesserung der Modellgüte, was das Potenzial des Ansatzes für praktische Anwendungen in der Automatisierungstechnik unterstreicht.

Schlagwörter: Synthetische Daten, Generative Methoden, Generative Adversarial Networks, 5G-Positionierung, Diffusionsmodelle

Abstract

The use of machine-learned algorithms is becoming increasingly important in automation technology as it enables data-driven problem solving. However, a major bottleneck for the use of powerful deep learning models in the field is the availability of extensive, labelled training data, the acquisition of which often requires high costs and operational adjustments in industrial practice. A promising approach to solving this problem lies in the use of synthetic data, which enables training without access to large real data sets. However, current methods often generate redundant scenarios, and it remains unclear which specific synthetic data contribute to improving the quality of the model.

In this dissertation, a concept for the targeted generation of synthetic data is developed that is specifically designed to address existing weaknesses in trained models and to improve their performance metrics in a targeted manner. The iterative concept comprises a generative neural network, a control component to optimise data generation and a decision logic that determines whether a generated data point should be used for training.

The effectiveness of the concept is demonstrated using use cases from automation technology: The MNIST handwriting recognition dataset is used as a controlled environment to evaluate the generator's ability to produce realistic, interpretable image data; a data-driven 5G position determination system is used to prove the transferability of the concept to an industrial production environment, showing a significant improvement in model accuracy and generalisation; the third use case investigates optical object detection in an industrial environment without available public datasets, and shows the potential of the concept to address specific industrial requirements.

The evaluation shows that the presented concept is able to generate realistic and targeted training data that can significantly increase the robustness and performance of neural networks in the context of machine learning and industrial automation. In all application cases, the use of synthetically generated data led to a substantial improvement in model quality, which emphasises the potential of the approach for practical applications in automation technology.

Keywords: synthetic data, generative methods, generative adversarial networks, 5G positioning, diffusion models

1 Einleitung

In diesem Kapitel wird zunächst die Motivation für die Forschungsarbeit aufgezeigt, die in der zunehmenden Bedeutung trainierter Algorithmen in der Automatisierungstechnik liegt. Die zentrale Problemstellung besteht darin, dass zur effektiven Nutzung von Deep-Learning-Modellen im Feldeinsatz viele gelabelte Trainingsdaten benötigt werden, deren Sammlung kostenintensiv und oft mit Anpassungen der Produktionsprozesse verbunden ist. Die Herausforderung liegt darin, gezielt synthetische Daten zu generieren, die das Modell verbessern, ohne überflüssige Daten zu produzieren. Auf Basis dieser Problemstellung werden die Ziele eines Lösungsansatzes formuliert, der es ermöglicht, Daten selektiv zu erzeugen, um eine verbesserte Modellgüte zu erreichen.

1.1 Motivation

Mit dem zunehmenden Einsatz maschinell gelernter Algorithmen im Bereich der Automatisierungstechnik steigt der Bedarf an robusten und leistungsfähigen Modellen, die auf spezifische Anforderungen industrieller Umgebungen zugeschnitten sind. Diese Modelle bewältigen komplexe Aufgaben, was durch das Training mit umfangreichen und qualitativ hochwertigen Datensätzen ermöglicht wird. Insbesondere Deep-Learning-Modelle sind datenhungrig und profitieren stark von einer großen Menge an gut annotierten Trainingsdaten, um auch unter variablen und störanfälligen Bedingungen zuverlässig zu funktionieren [1].

Das Sammeln von ausreichend vielen gelabelten Trainingsdaten, um ein Deep-Learning-Modell mit guter Leistung im Feldeinsatz zu trainieren, ist besonders im Bereich der industriellen Automatisierung schwierig [2], [3], [4]. Je nach Art der Daten kann es erforderlich sein, Produktionsprozesse zu unterbrechen oder den Betrieb zu stören, um die Daten zu sammeln. Zusätzlich ist oft unklar, welche Art von Daten für die Verbesserung des trainierten Modells relevant ist. Bei Bildverarbeitungsmodellen können es bestimmte Blickwinkel oder Lichtverhältnisse sein. Bei Modellen, die auf Funksignalen basieren, können es bestimmte Reflexionen sein, die durch sich bewegende metallische Oberflächen verursacht werden. Außerdem ist das Labeln der gesammelten Daten kostspielig, da der Labelingprozess immer noch meist manuell vorgenommen wird.

Eine Möglichkeit, das kostspielige Sammeln und Labeln von echten Daten zu vermeiden, ist die Verwendung von synthetischen Daten. Synthetische Daten ermöglichen es, den Bedarf an realen Daten signifikant zu senken. Sie ermöglichen ein effizientes, flexibles Training ohne Zugang zu riesigen realen Datensätzen, die oft schwer zu beschaffen sind. Synthetische Daten können auch dazu verwendet werden, komplexe oder seltene Szenarien zu simulieren, die in realen Daten selten vorkommen. Dies verbessert die Robustheit und Fehlertoleranz der Modelle, die sich so besser an unerwartete Situationen anpassen können.

1.2 Problemstellung und Herausforderungen

In diesem Kapitel wird die Problemstellung beschrieben, dass für die effektive Nutzung von Deep-Learning-Modellen im Feldeinsatz große Mengen gelabelter Trainingsdaten erforderlich sind. Die Sammlung solcher Daten ist jedoch kostspielig und erfordert häufig Anpassungen in den Produktionsprozessen. Aus dieser Problemstellung werden die spezifischen Herausforderungen abgeleitet, die bei der Erstellung geeigneter Trainingsdatensätze in der Domäne der Automatisierungstechnik bestehen.

Insbesondere im Kontext der Fabrikautomatisierung und Intralogistik stellt die Sammlung hochwertiger, realer, annotierter Trainingsdaten eine besondere Herausforderung dar: Das Sammeln von Trainingsdaten in der Automatisierungstechnik ist aufwändig, weil seltene, aber wichtige Systemzustände oft unterrepräsentiert sind. Da das System meist stabil läuft, sind Daten über seltene Systemzustände rar, was das Training erschwert. Zudem erfordert das Labeling dieser Daten teure manuelle Arbeit von Experten [5]. Die Komplexität der Daten, die für Menschen nicht intuitiv verständlich sind, verstärkt diese Herausforderungen weiter. Zusätzlich müssen gesammelte Daten auf ihre Relevanz und Qualität überprüft werden, um sicherzustellen, dass sie für das Training von Automatisierungssystemen geeignet sind [2], [3]. Diese Überprüfung ist kritisch, da ungenaue oder irreführende Daten die Leistung dieser Systeme beeinträchtigen können [3], [4].

Herausforderung 1 (H1): Das Sammeln realer Daten in der Domäne der Automatisierungstechnik ist aufwändig

Die Sammlung von relevanten Daten erfordert ein tiefes Verständnis der Prozesse und Abläufe innerhalb der Fabrikautomatisierung und Intralogistik, was den Prozess komplex und aufwändig macht [6]. Die Entwicklung effizienter Automatisierungssysteme in der Fabrikautomatisierung und Intralogistik basiert auf dem Verständnis und der Vorhersage komplexer physischer und logistischer Prozesse. Da sich Technologien und Produktionsmethoden kontinuierlich weiterentwickeln, kann es schwierig sein, vorherzusagen, welche spezifischen Daten oder Szenarien in Zukunft für die Optimierung der Systemleistung erforderlich sein werden [7]. Trainierte Modelle zeigen auch den Entwicklern gegenüber ein Black-Box-Verhalten. Es ist oft nicht vorhersehbar, welche Daten gesammelt oder generiert werden müssen, um die Modellgüte zu verbessern [7], [8], [9]. Dies stellt eine bedeutende Herausforderung dar, da es eine kontinuierliche Zusammenarbeit zwischen Datenwissenschaftlern, Ingenieuren und Prozessspezialisten erfordert, um die Relevanz und Effektivität der synthetischen Daten sicherzustellen [10].

Herausforderung 2 (H2): Es ist unsicher, welche weiteren Trainingsdaten benötigt werden

Die Anforderungen an spezifische Datentypen in der Domäne der Fabrikautomatisierung und Intralogistik sind besonders hoch, da Automatisierungssysteme oft unter extremen, variablen oder einzigartigen Bedingungen operieren müssen. Beispielsweise können Daten, die das Verhalten

von Robotern unter verschiedenen Lastbedingungen, einzigartige Objekte wie Spezialmaschinen oder die optimale Route von autonomen Fahrzeugen in einem sich ständig ändernden Lagerhaus darstellen, erforderlich sein [4]. Auch aufgezeichnete 5G-Empfangsdaten, die nur in der entsprechenden Umgebung nutzbar sind, sind ein Beispiel für solche industriellen Spezialdaten [11] (eigene Vorarbeit). Solche speziellen Datentypen sind in bestehenden Datensätzen nicht enthalten und es existieren auch keine vortrainierten Modelle [12], [13]. Dies macht die Erstellung neuer Datensätze erforderlich.

Herausforderung 3 (H3): Spezielle und einzigartige Datentypen

Zusammenfassung: Die Erstellung von Trainingsdatensätzen in der Domäne der Automatisierungstechnik ist durch signifikante Herausforderungen geprägt, die von der aufwändigen Sammlung realer Daten über die Unsicherheit zukünftig benötigter Daten bis hin zur Notwendigkeit spezieller Datentypen reichen. Diese Herausforderungen verdeutlichen die Notwendigkeit neuer Methoden und Techniken für die Erstellung von Datensätzen in der Automatisierungstechnik.

1.3 Zielsetzung der Arbeit

Ausgehend von der Problemstellung, dass zum Training von Deep-Learning-Modellen im Feldeinsatz umfangreiche Mengen gelabelter Trainingsdaten benötigt werden, deren Sammlung jedoch mit hohen Kosten verbunden ist, wird in diesem Kapitel die Zielsetzung der Dissertation formuliert. Dabei werden zunächst Teilaspekte der Zielsetzung aus den Herausforderungen der Problemstellung abgeleitet. Diese Teilaspekte bilden im weiteren Verlauf der Arbeit die Grundlage für die Architektur des Konzepts, das im Sinne des Design Science Research (DSR) durch die Umsetzung spezifischer Lösungsartefakte realisiert wird. Jedes Artefakt ist darauf ausgelegt, einen bestimmten Teil der Zielsetzung zu erfüllen und somit zur Gesamtlösung und zur Erreichung des übergeordneten Ziels beizutragen. Das manuelle Sammeln realer Daten sowie das Simulieren von Trainingsdaten sind insbesondere in der Domäne der Automatisierungstechnik sehr aufwändig (H1). Die automatische Generierung von Trainingsdaten bietet hier eine Lösung, indem sie die rasche Erstellung umfangreicher und vielfältiger Datensätze ermöglicht, ohne die logistischen und finanziellen Belastungen der realen Datenerfassung zu verursachen. Dies beschleunigt nicht nur den Entwicklungsprozess von Automatisierungssystemen, sondern ermöglicht auch die Generierung seltener oder gefährlicher Szenarien, die in der realen Welt nur schwer zu erfassen sind, wodurch die Effektivität dieser Systeme in der Praxis verbessert werden kann [4], [9], [14].

Automatisierte Trainingsdatengenerierung (Teilziel 1)

Wenn Trainingsdaten automatisiert generiert werden, lässt sich aus der Grundprämisse des maschinellen Lernens ableiten, dass die generierten Daten realistisch sein müssen: Für die Entwicklung effektiver und zuverlässiger Systeme ist es entscheidend, dass die Trainingsdaten die realen Betriebsbedingungen und Szenarien möglichst genau widerspiegeln. Unrealistische oder stark vereinfachte Daten können zu Modellen führen, die zwar in kontrollierten Umgebungen gut funktionieren, jedoch in der realen Welt versagen, weil sie auf unzureichenden oder irreführenden Informationen trainiert wurden [8], [15]. Dies kann insbesondere in kritischen Anwendungsfeldern der Automatisierungstechnik, wo Sicherheit von höchster Bedeutung ist, gravierende Folgen haben. Realistische Trainingsdaten gewährleisten, dass das Modell die Komplexität und Variabilität realer Prozesse erfasst und angemessen darauf reagieren kann. Dies beinhaltet die Berücksichtigung von Rauschen in den Daten, unerwarteten Ereignissen und der Vielfalt von Betriebszuständen, die in einer typischen Produktions- oder Logistikumgebung auftreten können [10] [14]. Die Fähigkeit, genaue Vorhersagen zu treffen und adäquat auf dynamische Veränderungen zu reagieren, hängt somit direkt von der Qualität und Realitätsnähe der verwendeten Trainingsdaten ab. Die Erstellung realistischer generierter Daten ist daher eine grundlegende Voraussetzung für

die Entwicklung robuster und zuverlässiger Automatisierungssysteme, die den Anforderungen realer Anwendungsszenarien gerecht werden [11].

Generierte Daten müssen realistisch sein (Teilziel 2)

Neben der Realitätstreue ist die gezielte Generierung von Trainingsdaten für die Entwicklung neuronaler Netze von entscheidender Bedeutung. Dies gilt insbesondere angesichts der Herausforderungen, die sich aus dem begrenzten Verständnis darüber ergeben, welche spezifischen Daten benötigt werden, um die Leistung der Modelle zu verbessern [9], [16]. Experten stehen oftmals vor dem Problem, dass neuronale Netze aufgrund ihrer "Black-Box"-Natur keine klaren Hinweise darauf geben, welche Arten von Daten oder Szenarien ihre Vorhersagegenauigkeit oder Anpassungsfähigkeit an realweltliche Bedingungen erhöhen können (H3). Die bloße Generierung von vielen synthetischen Daten ohne spezifische Zielsetzung kann zu einer ineffizienten Ressourcennutzung führen, da dadurch möglicherweise Daten erzeugt werden, die für das Training des jeweiligen Modells wenig bis gar keinen Mehrwert bieten [9]. Zudem erhöht sich das Risiko, dass die Modelle durch irrelevante oder redundante Daten überladen werden, was die Trainingszeiten unnötig verlängert und potenziell die Modellleistung durch Overfitting verschlechtern kann [17], [18].

Generierte Daten müssen gezielt generiert werden (Teilziel 3)

Die Generierung von synthetischen Daten in der Automatisierungstechnik muss ein hohes Maß an Flexibilität und Anpassungsfähigkeit aufweisen, um der Vielfalt der Datentypen und Einsatzszenarien gerecht zu werden (H3) [5], [11] (eigene Vorarbeit), [13]. Angesichts der unterschiedlichen Anforderungen von Anwendungen wie visueller Inspektion, prädiktiver Wartung und Robotersteuerung ist es essenziell, dass Konzepte zur Datengenerierung vielseitig einsetzbar sind und eine breite Palette von Daten simulieren können. Dies ermöglicht es Entwicklern, effizient leistungssteigernde Datensätze zu erstellen, die spezifische Anforderungen erfüllen, und trägt so zur Beschleunigung der Entwicklung und Optimierung von Automatisierungssystemen bei. Eine solche Anpassungsfähigkeit ist entscheidend, um die Entwicklung fortschrittlicher Automatisierungslösungen zu unterstützen und deren Leistungsfähigkeit und Zuverlässigkeit zu maximieren [10].

Übertragbarkeit für verschiedene Datenarten (Teilziel 4)

Ziel dieser Dissertation ist es, ein Konzept zur automatisierten Generierung von Trainingsdaten zu entwickeln, das gezielt Daten erzeugt, die zur Leistungsverbesserung von Deep-Learning-

Modellen in der Automatisierungstechnik beitragen. Die automatische Generierung soll die manuelle Datenerfassung und -annotation ergänzen und so die Effizienz des Trainingsprozesses steigern, indem sie spezifische, für das Modell relevante Daten abdeckt. Durch die Erfüllung der Teilziele strebt das Konzept an, die Effektivität von Automatisierungslösungen in der Praxis zu verbessern und die Leistung der Modelle gezielt zu steigern, indem es die Lücke zwischen der Notwendigkeit präziser Trainingsdaten und den beschränkten Möglichkeiten der manuellen Sammlung oder ungerichteten Generierung schließt.

1.4 Aufbau und Methodik der Arbeit

Die Struktur dieser Arbeit (Abbildung 1.1) orientiert sich am Design Science Research (DSR)-Ansatz, der als methodischer Rahmen für das wissenschaftliche Arbeiten in den ingenieurswissenschaftlichen Disziplinen etabliert ist. DSR zielt darauf ab, Probleme zu identifizieren, innovative Lösungen zu entwerfen und deren Wirksamkeit empirisch zu überprüfen.

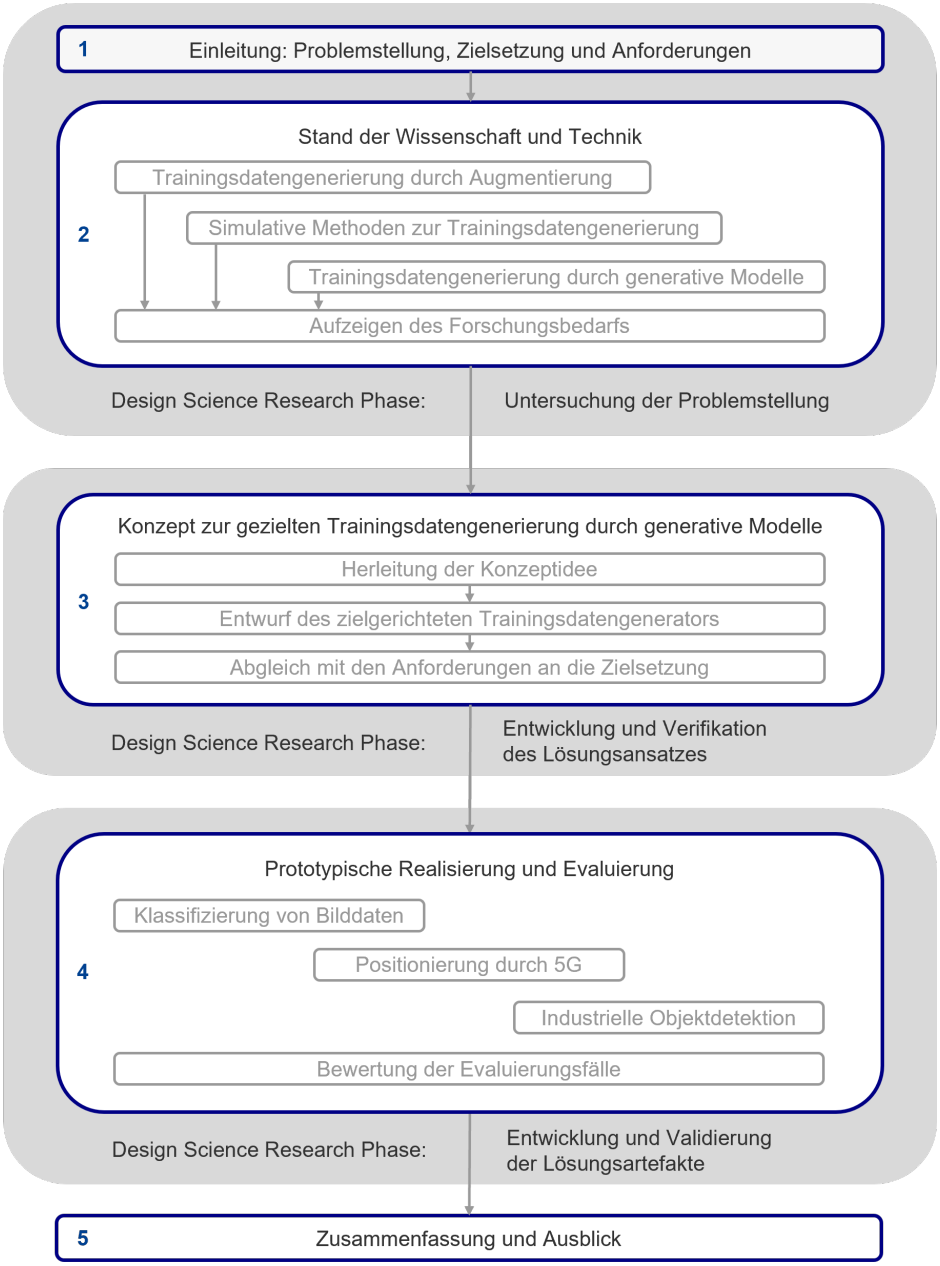

Abbildung 1.1: Aufbau der Arbeit

In Anlehnung an die klassischen DSR-Phasen der Problemdefinition und -analyse, der Entwicklung und Verifikation des Lösungsansatzes sowie der Implementierung und Validierung der Lösungsartefakte [19], [20] ist die Arbeit so strukturiert, dass jede Phase des Forschungsprozesses

die methodischen Anforderungen erfüllt. Kapitel 1 und 2 dienen der Untersuchung und Darstellung der Problemstellung, während Kapitel 3 die Entwicklung und theoretische Überprüfung eines Lösungsansatzes darlegt. Die Implementierung und Validierung der entworfenen Artefakte erfolgt in Kapitel 4, um deren Praxistauglichkeit zu evaluieren. Abschließend reflektiert Kapitel 5 die Forschungsarbeit und diskutiert weiterführende Fragestellungen im Kontext der gewonnenen Ergebnisse. Im Folgenden werden die einzelnen Kapitel genauer erläutert.

Kapitel 1 bildet den Ausgangspunkt der Untersuchung und stellt die Relevanz synthetischer Trainingsdaten in den Vordergrund. Hier wird die zentrale Zielsetzung der Arbeit formuliert und die spezifischen Anforderungen definiert, die für die Erreichung dieser Ziele erforderlich sind. Dieses Kapitel entspricht der DSR-Phase der Problemdefinition und -analyse, da es die Problemstellung präzisiert und die Notwendigkeit einer fundierten Lösung verdeutlicht.

Kapitel 2 befasst sich umfassend mit dem aktuellen Stand von Wissenschaft und Technik im Bereich der Generierung synthetischer Trainingsdaten. Eine systematische Einteilung der Ansätze in Augmentierung, Simulationen und generative Modelle schafft eine fundierte Basis zum Verständnis der vorhandenen Methoden und ihrer jeweiligen Stärken und Schwächen. Kapitel 2 trägt ebenfalls zur DSR-Phase der Untersuchung der Problemstellung bei, da es die theoretischen Grundlagen beleuchtet und eine Basis für die Entwicklung geeigneter Lösungsansätze bietet.

Kapitel 3 führt das Konzept eines gezielten Trainingsdatengenerators ein. Die drei zentralen Komponenten dieses Konzepts werden detailliert erläutert und ihre Eignung zur Erfüllung der zuvor definierten Anforderungen bewertet. Dieses Kapitel spiegelt die DSR-Phase der Entwicklung und Verifikation des Lösungsansatzes wider und bildet die theoretische Grundlage für die praktische Umsetzung und Anwendung des Konzepts.

Kapitel 4 dokumentiert die Implementierung des Konzepts anhand dreier spezifischer Anwendungsfälle, die sich in ihrer Komplexität und Problemstellung sukzessive steigern. Jeder Fall wird schrittweise evaluiert, um die Praxistauglichkeit des Konzepts zu überprüfen und den Nachweis für die Erfüllung der Forschungsziele zu erbringen. Damit wird in diesem Kapitel die DSR-Phase der Entwicklung und Validierung der Lösungsartefakte umgesetzt, da das entwickelte Konzept in der Praxis angewendet und seine Wirksamkeit getestet wird. Konkret werden pro Anwendungsfall drei Artefakte entwickelt: Das generative Modell, das die eigentliche Trainingsdatengenerierung durchführt, der Kontrollalgorithmus zur Steuerung der Datengenerierung und die Entscheidungskomponente, die entscheidet, ob die Trainingsdaten zur Leistungssteigerung geeignet sind.

Kapitel 5 fasst die zentralen Erkenntnisse der Arbeit zusammen und reflektiert die Zielerreichung sowie die Bedeutung der Ergebnisse. Ein Ausblick auf potenzielle zukünftige Forschungsrichtungen und Anwendungsfelder, die sich aus den Erkenntnissen dieser Arbeit ergeben, rundet die Darstellung ab. Dieses abschließende Kapitel bildet den Übergang zur Evaluation und Erweiterung innerhalb des DSR-Prozesses und ermöglicht die Reflexion des gesamten Forschungsprozesses.

2 Stand der Wissenschaft und Technik

In diesem Kapitel wird der Stand der Wissenschaft und Technik im Bereich der Trainingsdatengenerierung vorgestellt und untersucht. Dabei ist der Begriff weit gefasst und bezieht auch die Veränderung bereits existierender Trainingsdaten mit dem Ziel, den Trainingserfolg zu verbessern, als eine Form der Generierung ein. Deshalb beginnt das Kapitel mit ebendiesen Augmentierungsmethodiken. Der Übergang zwischen Augmentierungsmethoden, die in den Featureraum eingreifen und den allgemeineren Regularisierungsansätzen wie Dropout [21] ist fließend, weil beide den Featureraum beim Training verändern. Für dieses Kapitel wird die Grenze so gezogen, dass eine Methodik, sobald sie vollständig unabhängig von den zugrunde liegenden Daten ist, nicht mehr als Teil des Forschungsstands im Bereich der Trainingsdatengenerierung angesehen und hier nicht weiter besprochen wird.

2.1 Trainingsdatengenerierung durch Augmentierung

Datenaugmentierung bezeichnet den Prozess der Erhöhung der Vielfalt und Menge von Trainingsdaten in maschinellen Lernmodellen durch das Anwenden von Transformationen oder Techniken, ohne dabei die Labels der Daten zu verändern [22]. Das Forschungsfeld der Datenaugmentierung ist sehr kleinteilig nach Anwendungsdomänen unterteilt. Eine detaillierte Aufarbeitung des gesamten Forschungsfeldes ist hier daher nicht möglich. Ziel dieses Kapitels ist es, die Basismethoden der Augmentierung aufzuzeigen, um so den Zusammenhang zum Thema dieser Arbeit herausarbeiten zu können. Das Feld der Datenaugmentierung kann anhand der Eingriffsposition in die Daten kategorisiert werden in Eingabedatenaugmentierung und Featureraumaugmentierung [23], [24], [25]. Diese sind in den folgenden Abschnitten besprochen.

2.1.1 Eingabedatenaugmentierung

Techniken der Datenaugmentierung an den Eingabedaten repräsentieren eine Kategorie von Verfahren, die darauf abzielen, direkte Modifikationen an den Eingangsdaten oder deren Segmenten vorzunehmen, um eine erhöhte Variabilität zu induzieren und dadurch die Generalisierungsfähigkeit von maschinellen Lernmodellen zu verbessern. Dabei lassen sich die Eingabedatenaugmentierungsverfahren in zwei Unterkategorien einteilen: Filteraugmentierungen und semantische Datenaugmentierungen. Im Folgenden werden die beiden Unterkategorien erläutert.

2.1.1.1 Filteraugmentierungen

Filteraugmentierungen basieren auf Transformationen, die unabhängig vom semantischen Inhalt auf die Daten angewendet werden können. Der primäre Vorteil dieser Verfahren liegt in ihrer intuitiven Anwendbarkeit und der Möglichkeit, maßgeschneiderte Augmentierungen für

spezifische Aufgabenstellungen durch die Implementierung einfacher Transformationsoperationen zu realisieren. Insbesondere bei Bilddaten werden diese Transformationen seit Ende der 1990er-Jahre weit verbreitet eingesetzt [26]. Mittlerweile sind sie in modernen Deep-Learning-Softwarebibliotheken wie PyTorch bereits direkt enthalten [27]. Populäre Beispiele für diese Transformationen sind das Spiegeln, Zoomen, Perspektivverschiebungen und Verrauschungen. Bei Bilddaten werden neben den einfachen Transformationen auch fortgeschrittene fotometrische Filter eingesetzt; diese zielen darauf ab, reale fotografische Effekte wie Bewegungsunschärfe, optisches Rauschen und Farbartefakte zu simulieren, die die Bildqualität aufgrund von Kameraartefakten und Aufnahmebedingungen beeinträchtigen können. Im Gegensatz zu Transformationen wie Spiegelung, die die räumliche Struktur eines Bildes verändern, modifizieren fotometrische Methoden den Pixelinhalt, ohne die räumliche Anordnung zu verändern, und passen die Zusammensetzung der RGB-Kanäle mithilfe etablierter Bildverarbeitungstechniken an. Zu diesen Anpassungen gehören Änderungen der Farbe, Helligkeit, Schärfe, des Kontrasts und der Sättigung eines Bildes, um Umgebungsbedingungen wie unterschiedliche Beleuchtungsstärken, Wetterbedingungen und Tageszeiten zu imitieren oder um Artefakten von bildgebenden Geräten entgegenzuwirken. Zu den gängigen Ansätzen der fotometrischen Augmentation gehören Farb-Jittering, Farbraumkonvertierung, Bildverbesserung und Verzerrungstechniken, die die Robustheit von Deep-Learning-Modellen gegenüber verschiedenen Bildvariationen verbessern [24].

2.1.1.2 Semantische Augmentierung

Die semantischen Augmentierungsverfahren transformieren die Daten unter Bezugnahme auf die semantischen Inhalte der Daten. Bei Bilddaten schneiden die einfachsten Verfahren dieser Art Objekte aus Bildern aus und setzen sie auf andere Hintergründe oder führen weitere Transformationen durch. Beispiele hierfür sind CutPaste [28], InPS [29], PA-AUG [30], CutMix [31], CutBlur [32]. Diese Methoden konnten zwar nachweislich zu besseren Trainingsergebnissen führen, allerdings sind die dadurch entstehenden Bilder nicht realistisch; sie ähneln Fotocollagen. Kim et al. [33] haben diese Methoden untersucht und nachgewiesen, dass sie, weil die Datenvielfalt zufällig und ohne Berücksichtigung der Bildstruktur einbezogen werden, auch verwirrende Trainingsmuster erzeugen und die Netzleistung negativ beeinflussen können. Dadurch, dass feste Gewichtungskoeffizienten für die Bildmischung verwendet werden, die gleichmäßig über das gesamte Bild verteilt sind, kommt es dazu, dass weniger relevante Bereiche wie der Hintergrund betont und semantisch wichtige Regionen vernachlässigt werden. Um diese Probleme zu adressieren, schlagen sie PuzzleMix [33] vor. Diese Methode passt den Mischprozess an, indem sie die signifikanten Regionen und die lokale Bildstruktur berücksichtigt. Ähnliche Methodiken, die Bildregionen auf Basis ihrer Auffälligkeit kombinieren sind in [34] und [35] beschrieben. Augmentierungsmethoden, die auf Basis der Inhalte und durch Vermischung von Daten neue Daten erstellen, werden primär im Bereich von Bilddaten verwendet. Ein Beispiel aus der Audioklassifizierung ist in [36] beschrieben, bei dem eine Strategie zur Datenaugmentierung durch Mischung

von Audiosamples vorgeschlagen wird, um die Leistung von Modellen bei der Klassifizierung von Audioszenen, der Klassifizierung von Geräuschereignissen und der Sprachverbesserung zu verbessern. Neben Methoden, die Aspekte verschiedener Datenpunkte zu einem neuen Datenpunkt kombinieren, gibt es auch Methoden, die Teile von Bildern auf Basis von semantischen Annahmen über die Bilder ändern, um neue Trainingsdaten zu erzeugen (Abbildung 2.1) [37] (eigene Vorarbeit).

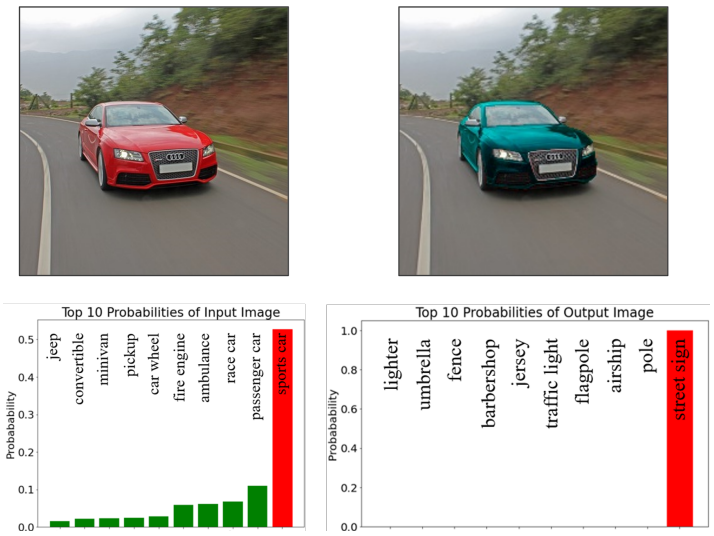

Abbildung 2.1: Das Originalbild links wurde durch die semantische Augmentierungsmethode unter der Annahme erstellt, dass die Lackfarbe des PKW keine Bedeutung für die Detektion haben sollte.

2.1.2 Featureraumaugmentierung

Die Idee der Featureraumaugmentierung besteht darin, extrahierte Featurevektoren zu transformieren, um die Darstellungsvarianz im Featureraum zu vergrößern, während das Klassenlabel auf der Eingabeebene weiter erhalten bleibt. Ziel ist es, die Vielfalt der extrahierten Features zu erhöhen, um zu verhindern, dass das zugrundeliegende Modell spezifische Konfigurationen der Eingabedaten erlernt. Auch im Featureraum lassen sich die Augmentierungen in Filter- und semantische Augmentierungen kategorisieren.

2.1.2.1 Filteraugmentierungen

Filteraugmentierungen basieren auch in der Featureebene auf Transformationen, die unabhängig des semantischen Inhalts auf die Daten angewendet werden. Jia und Zhang [38] schlagen Shakedrop vor, bei dem sie kNN-Techniken verwenden, um die Featurebeziehungen zwischen Instanzen in einem mehrdimensionalen Klassifikationskontext zu nutzen und somit die ursprünglichen Featuredarstellungen zu transformieren und die Leistung von mehrdimensionalen Klassifikationsmodellen zu verbessern. Gastaldi [39] schlägt vor, Summationsoperationen durch affine

Kombinationen zu ersetzen – eine Technik, bei der stochastische affine Transformationen im Summationsprozess angewendet werden. Diese Technik wurde verwendet, um die Leistung von ResNeXt-Modellen bei verschiedenen Computer Vision-Aufgaben zu verbessern. Shen et al. haben einen Ansatz vorgestellt, bei dem zufällige affine Operationen – speziell Translation, Skalierung und Rotation – auf gelernte Featuremaps angewendet werden, um deren Variabilität zu erhöhen [43]. Ähnlich dazu schlagen Li und Liu vor, adaptive Gaußsche Geräusche in einige tiefe CNN-Schichten einzufügen, um die Featurevielfalt zu fördern und Übertraining zu verhindern [40].

2.1.2.2 Semantische Augmentierung

Ähnlich wie im Eingabedatenraum fokussieren sich die Methoden, die den Featureraum augmentieren, auf das Verbinden von verschiedenen Featureräumen unterschiedlicher Eingangsdaten [41]. Dabei haben die Autoren ein separates Attribut-Lern-Subnetz trainiert, das basierend auf Annotationen passende Features vorschlägt, die dann mit der Attribut-Mix-Technik zu einem neuen Featurevektor zusammengefügt werden können. Faramarzi et al. kombinieren die Featurevektoren blockbasiert, was Ähnlichkeiten mit Bildcollagen aufweist, allerdings auf Featureebene [42]. In [43] ist eine Interpolation von unterschiedlichen Featureräumen auch über unterschiedlich tief liegende CNN-Layer hinweg umgesetzt. Ein grundsätzliches Problem dieser Ansätze ist, dass sie ein sehr gutes Verständnis des Featureraums erfordern, das typischerweise kaum gegeben ist. Die Methoden können zwar trotzdem angewendet werden, jedoch ignorieren die Transformationen dann jegliches Wissen über die Domänen und können unrealistische Daten erzeugen.

2.1.3 Adversariale Augmentierung

Dieses Kapitel beleuchtet die Augmentierungsverfahren, die adversariale Methoden nutzen, um zusätzliche Trainingsdaten zu erzeugen. Diese Methoden erzeugen speziell konstruierte Eingabedaten, bekannt als "Adversariale Beispiele", um die Grenzen und Schwächen von ML-Modellen aufzudecken und diese zu verbessern. Adversariale Beispiele sind Eingaben, die gezielt entworfen werden, um ein ML-Modell zu einer falschen Vorhersage oder Klassifizierung zu verleiten. Sie bieten somit Einblicke in die Vulnerabilität von KI-Systemen gegenüber manipulativen Eingriffen. Die Verwendung von adversarialen Beispielen zum Training von neuronalen Netzen wurde zuerst von Goodfellow et al. in [44] beschrieben. Inspiriert von Goodfellow et al. hat sich ein ganzes Forschungsfeld mit dem Ziel gebildet, neuronale Netze zu trainieren, die robust gegenüber adversarialen Beispielen sind. Dieses Kapitel bietet keinen umfänglichen Überblick über das ganze Forschungsfeld, sondern stellt nur ausgewählte Arbeiten vor, die Augmentierungen nutzen. Zunächst lag der Fokus auf dem genaueren Verständnis der Beispiele und wie Robustheit gegen sie definiert und messbar gemacht werden kann. So haben Madry et al. [45] mit Optimierungsmethoden ein robustes Netzwerk auf dem CIFAR [46] Datensatz trainiert, dessen Genauigkeit

dafür stark eingeschränkt war. Bei einem durch die NIPS-Konferenz ausgerufenen Wettkampf der besten Methoden hat eine Weiterentwicklung des von Goodfellow entwickelten Fast-Gradient-Sign-Method (FGSM)-Ansatzes die besten adversarialen Beispiele hervorgebracht [47]. Die robusten Netze haben allerdings weiter den Nachteil, dass deren Genauigkeit deutlich unter dem Durchschnitt der zu dieser Zeit üblichen Werte von nicht robusten Modellen anzusiedeln sind. Genau dieses Phänomen, dass es zwischen Robustheit und Genauigkeit von neuronalen Netzen eine prinzipbedingte Abwägung gibt, wurde sowohl von Tsipras et al. [48] als auch von Zhang et al. [49] theoretisch nachgewiesen.

2.2 Simulative Methoden zur Trainingsdatengenerierung

Simulative Methoden werden seit Jahrzehnten in der Automatisierungstechnik und insbesondere auch in der Fertigung eingesetzt [50]. Die Relevanz von Simulation in der Automatisierungstechnik nimmt dabei immer weiter zu und wird dies voraussichtlich auch in der Zukunft tun, wie Umfragen zeigen [51], [52]. Auch die Trainingsdatengenerierung durch Simulation wird in sehr vielen Bereichen eingesetzt. Insbesondere im Bereich des Reinforcement Learnings ist die Simulation oft die einzige Quelle von Trainingsdaten. Diese Arbeit betrachtet Reinforcement Learning nicht, daher beschränkt sich dieses Kapitel darauf, einzelne Aspekte der simulativen Generierung von Trainingsdaten außerhalb des Reinforcement Learnings zu betrachten. Für eine ausführlichere Betrachtung des Feldes der simulativen Trainingsdatengenerierung kann hier auf Nikolenko [53] verwiesen werden. Ferner beschränkt sich dieses Kapitel auf die Simulation von Trainingsdaten in den Domänen der Evaluierungsfälle dieser Arbeit.

2.2.1 Umgebungssimulationen

Für viele Anwendungen des maschinellen Lernens werden Sensordaten benötigt, die Informationen über die Umgebung erfassen, beispielsweise durch Thermometer, Radar, LiDAR oder andere Funktechnologien wie 5G. Damit solche Daten generiert werden können, müssen zumindest Ausschnitte der Umgebung simuliert werden, sodass Sensordaten als Trainingsdaten erzeugt werden können.

Eines der verbreiteten Umgebungssimulationstools ist Gazebo [54]. Gazebo ist eine essenzielle Simulationsumgebung für die Robotikforschung, die die Modellierung, Simulation und das Testen komplexer robotischer Systeme in einer virtuellen Realität ermöglicht. Ähnlich zu Gazebo ist auch MuJoCo eine verbreitete Physik-Engine für Robotik- und Umgebungssimulationen [55]. Beide Systeme sind dazu geeignet, Trainingsdatensätze für maschinelles Lernen unter sichereren, kosteneffizienten und kontrollierbaren Bedingungen zu generieren. So bieten sie Monokamera, Tiefenkamera, LIDAR, IMU, Kontaktsensor, Höhenmesser, und Magnetometer Simulationen an. Auch Sensorrauschen kann durch Gaußrauschen oder benutzerdefinierbare Rauschfunktionen implementiert werden. Neben diesen verbreiteten Simulationsframeworks existieren zahlreiche Speziallösungen für einzelne Domänen, wie Microsofts AirSim [56] für das Fliegen.

Der Simulator verbindet die Bereiche Flug- und Robotersimulation und bietet eine detaillierte Physik-Engine. Dadurch wird nicht nur die realistische Flugmodellierung verbessert, sondern auch die Integration mit Robotik-Betriebssystemen erleichtert. So wird die direkte Interaktion mit Software- und/oder Hardwaresystemen unbemannter Luftfahrzeuge (Unmanned Aerial Vehicles, UAVs) unterstützt. Im Gegensatz zu herkömmlichen Flugsimulatoren wie Microsoft Flight Simulator oder X-Plane [57] nutzt AirSim die Unreal Engine 4 für eine realitätsgetreue Visualisierung

und bietet umfassende Sensoremulationsfunktionen, einschließlich Barometer, Gyroskope, Beschleunigungsmesser, Magnetometer und GPS. Die Erweiterbarkeit von AirSim durch Unreal Engine 4-Plugins fördert die Entwicklung autonomer Fahrzeugprojekte und synthetischer Simulationsumgebungen. Diese wurden in verschiedenen Forschungserweiterungen ausgiebig genutzt. Beispielsweise wurden realistische Waldszenarien für Studien zur visuellen Wahrnehmung von UAVs erstellt [58]. Auch Erweiterungen für Aufgaben im Bereich der Erhaltung der Tierwelt wurden in Umgebungen wie der afrikanischen Savanne modelliert [59]. Im Rahmen von Projekten wurden zudem Szenarien für kritische Zwischenfälle simuliert, beispielsweise chemische, biologische oder nukleare Ereignisse, um UAVs für die Datenerfassung in Notfällen zu trainieren [60] [61]. Eine bedeutende Erweiterung von Huang et al. [62] ist ein Modell zum Verstehen natürlicher Sprache, um Drohnenbefehle in natürlicher Sprache zu verarbeiten und auszuführen.

Auch für den Bereich des autonomen Fahrens existieren mehrere Spezialsimulatoren, mit denen synthetische Trainingsdaten erzeugt werden können.

CARLA (CAR Learning to Act) [63] ist ein Open-Source-Simulator für das Fahren in der Stadt, der auf der Plattform Unreal Engine 4 entwickelt wurde. Er stellt eine umfassende Reihe von Sensoren bereit, die für autonome Fahranwendungen erforderlich sind, ähnlich wie andere Simulationen [64]. Dazu gehören RGB-Kameras mit einstellbaren Positionen, Tiefenkarten und semantische Segmentierungskarten, die in 12 semantische Klassen unterteilt sind, die für Fahrszenarien relevant sind (z. B. Straßen, Fahrbahnmarkierungen, Verkehrsschilder), sowie Bounding-Boxen für sich bewegende Objekte und Messungen der Fahrzeugposition und -orientierung. Auch der Simulator DeepDrive [65] ist als Unreal-Engine-Plugin realisiert. Das System ist mit acht RGB-Kameras ausgestattet, die nahezu in Echtzeit eine Auflösung von 512x512 bei einer Frequenz von 20 Hz liefern. Der Aufbau ist für die Entwicklung von ML-Algorithmen optimiert, indem verschiedene und dynamische Fahrbedingungen simuliert werden. Dadurch soll die Effektivität und Robustheit der Algorithmen im realen Betrieb verbessert werden. Bei Straßenszenen sind auch Passanten von großer Bedeutung, die aber bei CARLA und DeepDrive nur eine untergeordnete Rolle spielen. Dieser Nische hat sich VIVID (VIrtual environment for VIsual Deep learning) von Lai et al. [66] angenommen und hat Funktionen wie das detaillierte Skripten von Passantenfußwegen, wie es in Videospielen üblich ist, eingeführt. Generell lässt sich ein Trend feststellen, dass die Simulatoren immer mehr mit der Welt der Videospiele verschmelzen. Üblicherweise lag der Fokus bei professionellen Fahrsimulatoren für die Fahrzeugentwicklung auf der physikalisch korrekten Simulation der Fahrdynamik. Für synthetische Trainingsdaten waren allerdings schon seit den 90er-Jahren die Simulatoren, die auf Videospielen aufgebaut haben, von größerer Bedeutung, wie TORCS (The Open Racing Car Simulator) [67], CARLA [63], LGSVL [68] und Simulatoren, die auf Grand Theft Auto (GTA) aufbauen. Durch die Verwendung von synthetischen Trainingsdaten aus der GTA V Engine gelang es Yue et al. [69] ein Modell zur

Punktewolken-Segmentierung aus LIDAR-Daten signifikant zu verbessern. Auch der verbreitete PreSIL (Precise Synthetic Image and LIDAR) Datensatz basiert auf der GTA V Engine [70].

Die qualitativen Unterschiede der durch die Ansätze generierten Ergebnisse zeigen, dass zur Erreichung eines hohen Realitätsniveaus in Simulationen erhebliche Rechenressourcen erforderlich sind, insbesondere beim Rendern dynamischer und detaillierter Umgebungen, die für das Training von Modellen notwendig sind.

Die Schaffung realistischer Trainingsumgebungen erfordert nicht nur viel Rechenleistung, sondern auch viel Arbeit. Besonders bei ungewöhnlichen oder neuartigen Problemen, für die es keine vordefinierten Vorlagen oder Umgebungen gibt, ist die Gestaltung von Szenarien eine Herausforderung. Die Verwendung von Spiel-Engines wie der Unreal Engine oder Unity zur Erstellung von Simulationen führt zu einer zusätzlichen Komplexitätsebene. Obwohl diese Plattformen leistungsstark sind, erfordern sie in der Regel Expertenwissen, um die Einstellungen effektiv zu manipulieren und zu optimieren, um den gewünschten Grad an Realismus und physikalischer Genauigkeit zu erreichen. Das für die Anpassung von Spiel-Engines erforderliche Fachwissen ist nicht trivial. Es erfordert ein tiefes Verständnis sowohl der technischen Aspekte der Spielentwicklung als auch der spezifischen Anforderungen des Simulationsszenarios. Die Notwendigkeit von hoch spezialisiertem Wissen kann die Zugänglichkeit simulationsbasierter Datengenerierungstechniken einschränken und macht sie für Teams ohne direkten Zugang zu Spielentwicklungsexpertise weniger praktikabel. Simulationen mithilfe von Videospiel-Engines können in kontrollierten Umgebungen wertvolle Daten liefern. Allerdings sind sie aufgrund der hohen Anforderungen an Rechenleistung und Expertenwissen in vielen Kontexten des angewandten maschinellen Lernens eine ineffiziente Lösung für die Generierung von Trainingsdaten.

2.2.2 Simulation von Bilddaten

Die Nutzung simulierter Bilddaten für Aufgaben des Machine Learning, die damals noch vor allem durch klassische, nicht trainierte Filtermethoden der computergestützten Bildverarbeitung geprägt waren, geht der Nutzung echter Bilder sogar voraus. Bereits in den 1960er-Jahren wurden erste Versuche mit Kantendetektoren auf synthetischen, damals noch gezeichneten Bildern durchgeführt, da echte Fotografien für die Algorithmen noch zu herausfordernd waren [53]. Über die Jahrzehnte hat die Bedeutung von simulierten Bilddaten weiter zugenommen, insbesondere als wissenschaftliche Vergleichsdatensätze sind sie heute von hoher Bedeutung. Beispiele für wichtige simulierte Bild-Vergleichsdatensätze sind SceneNet [71], PartNet [72] und Cloth3D [73].

In den vergangenen Jahren hat die Generierung simulierter Bilddaten signifikante Fortschritte erlebt, maßgeblich bedingt durch die Entwicklung und Verbesserung von Werkzeugen im Bereich der Computergrafik. Speziell Spiel-Engines wie Unity und Unreal Engine, frei verfügbare 3D-Grafiksoftware wie Blender sowie die zunehmende Zugänglichkeit umfangreicher Bibliotheken

von 3D-Assets haben die Entwicklung von maßgeschneiderten Simulatoren erheblich vereinfacht [74]. Diese technologischen Fortschritte ermöglichen es Forschern und Entwicklern, realitätsnahe und hoch detaillierte simulierte Umgebungen für eine Vielzahl von Anwendungsgebieten zu erstellen. Dadurch eröffnen sich neue Möglichkeiten für die Validierung und das Training von Algorithmen des maschinellen Lernens und der künstlichen Intelligenz, insbesondere in Bereichen, in denen der Zugang zu realen Daten entweder beschränkt oder mit hohen Kosten verbunden ist. Beispiele hierfür finden sich im Bereich des Objekt-Trackings [75], [76], der Schätzung des Bildausschnitts [77], der semantischen Segmentierung [78], [79], [80] und für diese Arbeit besonders relevant die Objektdetektion [81], [82], [83], [84] (eigene Vorarbeit).

In einer Vielzahl von Anwendungsfällen haben sich synthetische Bilddaten als effektives Mittel zur Verbesserung der Leistung neuronaler Netze erwiesen, vorwiegend wenn diese Daten mit realen Daten kombiniert werden. Die zugrundeliegenden Syntheseverfahren basieren in der Regel auf Computergrafik-Rendering-Pipelines. Diese nehmen 3D-Szenendaten (z.B. Punktkoordinaten im Raum, die ein Fahrzeug definieren), Informationen über Material- und Beleuchtungseigenschaften, wie z.B. Fahrzeugfarbe und Lichtquellen, sowie Renderingparameter, wie z.B. die Verwendung von Raytracing oder Rasterisierungsalgorithmen, auf und erzeugen daraus ein Bild der Szene. Durch den Zugriff auf detaillierte Informationen über die Szene ist die Pipeline in der Lage, automatisch fehlerfreie Labels wie Bounding-Boxen für relevante Objekte, Tiefeninformationen und Segmentierungsmasken der Szene zu erzeugen.

Durch Erhöhung des Detaillierungsgrades der 3D-Szenen, z.B. durch Erhöhung der Anzahl der Punkte, die die interessierenden Objekte definieren, und durch Optimierung der Rendering-Algorithmen, ist es möglich, den visuellen Realismus der Ausgabe zu erhöhen, d. h. die visuelle Übereinstimmung der erzeugten Szene mit der realen Umgebung zu verbessern. In ähnlicher Weise kann der Realismus der Bewegungsdarstellung durch den Einsatz von 3D-Bewegungserfassungstechniken (z. B. für die Erkennung menschlicher Aktivitäten) und fortgeschrittenen Physikengines (z. B. für die Robotermanipulation) erhöht werden. Dies wurde experimentell vielfach nachgewiesen: Durch Zhang et al. mit Innenraumszenen für die Servicerobotik [85]. Wrenninge et al. [86] und Tsirikoglou et al. [87] für Straßenszenen. Planche et al. generierten Tiefenbilder für Bilder von Motoren [88]. Movshovitz-Attias et al. [77] haben am Beispiel der Blickwinkelschätzung systematisch untersucht, wie sich die Realitätstreue der Bilddaten auf die Erkennungsrate auswirkt und kamen zu dem Ergebnis, dass mit höherem Realitätsgrad des Renderings auch das Trainingsergebnis besser wird. Im Allgemeinen führt eine Erhöhung der Realitätsnähe der synthetischen Daten also zu einer verbesserten Leistung von Deep-Learning-Systemen im Einsatz. Es gibt auch Methodiken, mit denen fehlende Realitätstreue ausgeglichen werden kann, wie Domain Randomization [86] und dem Compositing-Ansatz von Guosheng et al. [89], wobei diese Methoden mit einem erheblichen Mehraufwand einhergehen und ein tiefes Verständnis über die jeweiligen Daten benötigen. Generell kann festgestellt werden, dass der Einsatz von Simulationen zur

Generierung von hochwertigen synthetischen Bilddaten sehr große technische und personelle Aufwände erfordert. Es werden Spezialisten benötigt, die Rendering-Pipelines aufbauen und einstellen sowie leistungsstarke Rechner, die sie ausführen.

2.2.3 Simulation von Funkdaten

Meist werden Funkdaten durch analytische Methoden basierend auf mathematischen Modellen simuliert; diese basieren allerdings auf vielen vereinfachenden Annahmen, die ihre Nutzung zur Abbildung realer Szenarien nach Qureshi et al. unbrauchbar machen [90]. Weiter sind sie oft unzureichend, da sie in der Regel weder spezifische Konfigurationen wie die genauen Standorte der Basisstationen (BS) berücksichtigen, noch detaillierte Parameter wie Antennenhöhe, Neigung und Sendeleistung für einzelne BS unterstützen. Ferner erfassen Modelle, die auf stochastischer Geometrie basieren, nicht die Dynamik des Netzes, die für die Leistungsbewertung entscheidend ist, wie z.B. Mobilitätsmanagement und Übertragungslatenz. Simulatoren auf Systemebene sind sowohl in der Industrie als auch im akademischen Bereich zu unverzichtbaren Werkzeugen geworden, um die Unzulänglichkeiten analytischer Modelle und die geringe Praktikabilität von Feldexperimenten zu beheben. Umfängliche Arbeiten zu den systemischen Simulatoren sind in [91] und [92] zu finden. Die systemischen Simulatoren sind dafür entwickelt, Netze auszulegen, und können auch dazu verwendet werden, Trainingsdaten zu generieren. Beispielsweise haben Masood et al. [93] mit einem Simulator Trainingsdaten generiert, um ein realistisches Dämpfungsmodell für die Netzauslegung zu trainieren.

2.3 Trainingsdatengenerierung durch generative Modelle

In diesem Kapitel ist der Stand der Wissenschaft und Technik zu mit realen Daten trainierten generativen Modellen für die Generierung von Trainingsdaten beschrieben. Da auf allen generierten Daten trainiert werden könnte, ist zunächst ein Überblick über den Stand der Forschung von trainierten generativen Modellen im Allgemeinen gegeben, danach folgen Modell-agnostischen Methoden zur Generierung von Trainingsdaten und ein Kapitel über die Modell-informierte Generierung von Trainingsdaten.

2.3.1 Generative Modelle

Deep-Learning-Modelle können in diskriminierende und generative Modelle unterteilt werden. Diskriminierende Modelle lernen die konditionale Verteilung $p(y \mid x)$ einer Datenmenge und können so beispielsweise Klassifikations- oder Regressionsaufgaben durchführen. Generative Modelle hingegen lernen die multivariate Verteilung $p(x,y)$ einer Datenmenge und können damit keine Klassifikations- und Regressionsaufgaben ausführen, sondern Daten für diese Verteilung generieren. Generative Modelle gelten als anspruchsvoller in der Realisierung und im Training [94].

Derzeit lassen sich vier populäre Arten generativer Modelle unterscheiden: der Variational Autoencoder (VAE) [95], das Generative Adversarial Network (GAN) [96], Transformer [97] und die Diffusionsmodelle [98]. In den letzten Jahren haben sich VAEs als nicht mehr konkurrenzfähig gegenüber GANs und Diffusionsmodellen gezeigt, daher werden sie im Folgenden nicht weiter besprochen [94]. Die Transformermodelle werden zum Zeitpunkt des Verfassens dieser Arbeit fast ausschließlich für die Sprachverarbeitung verwendet, was nicht Thema dieser Arbeit ist, daher werden sie ebenfalls nicht weiter besprochen. In den folgenden Kapiteln ist der Stand der Technik im Bereich der GANs und der Diffusionsmodelle beschrieben.

2.3.1.1 Generative Adversarial Network

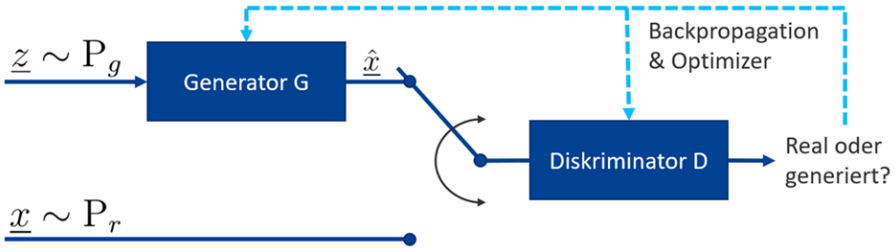

Abbildung 2.2: Grundstruktur eines Generative Adversarial Networks

Das GAN, erstmalig von Goodfellow et al. in [96] eingeführt, ist ein komplexes Modell bestehend aus zwei simultan trainierten Netzwerken: einem generativen Modell G und einem diskriminierenden Modell D. Die grundlegende Architektur eines GANs wird schematisch in Abbildung 2.2 dargestellt. Der Generator G zielt darauf ab, Daten zu erzeugen, die nicht von echten, tatsächlichen Daten unterscheidbar sind, indem er aus einer zufällig gewählten latenten Variablen z (bekannt als „Latentspacevektor“) schöpft. Der Diskriminator D evaluiert diese generierten Daten gegenüber den realen Daten x und versucht, beide zu differenzieren.

Die Lernprozesse beider Modelle erfolgen durch Backpropagation, wobei G aus den Bewertungen von D lernt, wie die generierten Daten verbessert werden müssen, um realistischer zu erscheinen. Dieser Prozess ist durch ein Min-Max-Optimierungsproblem charakterisiert, das theoretisch zum globalen „Nash-Gleichgewicht“ führt, bei dem D nicht mehr fähig ist, zwischen generierten und realen Daten zu unterscheiden [96]:

$$\min_G \max_D V(D,G) = \mathrm{E}_{\underline{x}\sim \mathrm{P}_r}[\log D(\underline{x})] + \mathrm{E}_{\underline{z}\sim \mathrm{P}_g}\left[\log\left(1 - D\left(G(\underline{z})\right)\right)\right] \qquad (1)$$

wobei E der Erwartungswert ist. Das Erreichen dieses Gleichgewichts stellt jedoch eine signifikante Herausforderung dar, da das Optimierungsproblem in einem Raum mit theoretisch unendlich vielen Dimensionen operiert. Es ist möglich, dass nur lokale Gleichgewichte erreicht werden, oder dass ein globales Gleichgewicht theoretisch nicht existiert [104]. Aufgrund dieser Schwierigkeiten wurden zahlreiche Modifikationen und Weiterentwicklungen von GANs spezifisch für verschiedene Anwendungen entwickelt. Diese Varianten und ihre Anwendungen werden im Folgenden diskutiert.

Deep Convolutional Generative Adversarial Network

In der Veröffentlichung [99] präsentieren Radford et al. das Deep Convolutional Generative Adversarial Network (DCGAN), welches die Techniken von Convolutional Neural Networks (CNNs) und GANs integriert, um die Vorteile beider Ansätze zu nutzen. Die Hauptmotivation

hinter der Entwicklung von DCGANs war die Überwindung der Instabilität beim Training größerer GANs, die mit CNNs strukturiert sind. Ein spezifisches Problem bei der Skalierung von GANs, das als "Mode Collapse" bekannt ist, tritt auf, wenn das Netzwerk beginnt, minimale oder keine Varietät in den Ausgangsdaten zu produzieren, ungeachtet der Variabilität in den Eingangsdaten.

Um die Stabilität während des Trainingsprozesses zu verbessern und die Architektur robuster gegenüber dem "Mode Collapse" zu machen, implementierten Radford et al. spezifische architektonische Richtlinien für DCGANs. Diese Leitlinien sind wie folgt definiert [99]:

- Ersetzung von Pooling-Layer: Statt der herkömmlichen Pooling-Schichten werden in DCGANs „Strided Convolutions“ eingesetzt, die dazu beitragen, die räumliche Hierarchie der Merkmale effektiver zu lernen.
- Einsatz von Batch Normalization (Batchnorm): Diese Technik wird angewandt, um die interne Kovarianzverschiebung zu reduzieren, was die Effizienz des Trainings erhöht und die Wahrscheinlichkeit des Mode Collapses minimiert.
- Entfernung von Fully Connected Hidden Layers: Durch das Weglassen dieser Schichten wird eine direktere Verbindung zwischen Eingabe- und Ausgabedaten ermöglicht, was die Netzwerkarchitektur vereinfacht, und die Stabilität verbessert.
- Verwendung spezifischer Aktivierungsfunktionen: Im Generator wird die ReLU-Aktivierungsfunktion verwendet, um eine effiziente Propagation positiver Gradienten zu gewährleisten. Der Diskriminator verwendet die LeakyReLU-Aktivierung, um auch bei negativen Eingangswerten eine Gradientenpropagation zu ermöglichen.

Diese strategischen Anpassungen in der Netzwerkstruktur von DCGANs tragen signifikant zur Stabilisierung des Trainingsprozesses bei und erweitern die Fähigkeit des Modells, sehr diverse und qualitativ hochwertige generative Ausgaben zu erzeugen. Durch diese methodischen Optimierungen setzen DCGANs einen neuen Standard in der Entwicklung effizienter und stabiler generativer Netzwerke.

Wasserstein Generative Adversarial Networks

Das Wasserstein Generative Adversarial Network (WGAN), entwickelt von Arjovsky et al. in [100], zielt darauf ab, die Qualität der generierten Daten zu erhöhen und gleichzeitig die Problematik des Mode Collapses zu vermeiden, die bei herkömmlichen GANs häufig auftritt. Das fundamentale Problem konventioneller GANs liegt in der verwendeten Metrik für die Divergenz in ihrer Kostenfunktion, speziell der Jensen-Shannon-Divergenz (JS).

Die Jensen-Shannon-Divergenz ist wie folgt definiert:

$$JS(\mathrm{P}_r, \mathrm{P}_g) = KL\left(\mathrm{P}_r \parallel \frac{\mathrm{P}_r + \mathrm{P}_g}{2}\right) + KL\left(\mathrm{P}_g \parallel \frac{\mathrm{P}_r + \mathrm{P}_g}{2}\right) \tag{2}$$

wobei

$$KL(\mathrm{P}_r \parallel \mathrm{P}_g) = \int \log\left(\frac{\mathrm{P}_r(x)}{\mathrm{P}_g(x)}\right)\mathrm{P}_r(x)\mathrm{d}\mu(x) \tag{3}$$

die Kullback-Leibler-Divergenz darstellt, und P_r sowie P_g die realen bzw. vom Netzwerk approximierten Verteilungen sind.

Die Kullback-Leibler-Divergenz, die auf logarithmischen Unterschieden basiert, ist jedoch nicht immer definiert (potenziell unendlich), was insbesondere dann ein Problem darstellt, wenn P_g nicht den Support von P_r vollständig überdeckt. Dies kann während des Trainings eines GANs zu instabilen Ergebnissen und der bereits erwähnten Problematik des Mode Collapses führen, bei dem das Netzwerk nicht die gesamte Bandbreite der Datenverteilung lernt.

Arjovsky et al. [100] schlagen daher vor, die Wasserstein-Distanz W als Alternative zu verwenden:

$$W(\mathrm{P}_r, \mathrm{P}_g) = \inf_{\gamma \in \Pi(\mathrm{P}_r, \mathrm{P}_g)} \mathrm{E}_{(x,y)\sim\gamma}[\parallel x - y \parallel] \tag{4}$$

Hierbei repräsentiert $\prod(P_r, P_g,)$ die Menge aller möglichen gemeinsamen Verteilungen $\gamma(\mathrm{x},\mathrm{y})$ mit den Randverteilungen P_r und P_g. Die Wasserstein-Distanz, die das Minimum der erwarteten Entfernung zwischen gekoppelten Punkten aus beiden Verteilungen sucht, bietet eine robustere Metrik, da sie auch dann definiert ist, wenn die Support-Mengen der Verteilungen sich nicht überschneiden.

Zur praktischen Berechnung der Wasserstein-Distanz nutzen Arjovsky et al. die Kantorovich-Rubinstein-Dualität, die sich wie folgt formuliert:

$$W(\mathrm{P}_r, \mathrm{P}_\theta) = \sup_{||f||_L \leq 1} \mathrm{E}_{x\sim\mathrm{P}_r}[f(x)] - \mathrm{E}_{x\sim\mathrm{P}_\theta}[f(x)] \tag{5}$$

wobei das Supremum über alle 1-Lipschitz-Funktionen $f: X \rightarrow \mathbb{R}$ genommen wird. Diese Formulierung führt direkt zur Verlustfunktion für den Diskriminator in WGANs:

$$L = \mathrm{E}_{\underline{z}\sim\mathrm{P}_g}\left[D\left(G\left(\underline{z}\right)\right)\right] - \mathrm{E}_{\underline{x}\sim\mathrm{P}_r}\left[D\left(\underline{x}\right)\right] \tag{6}$$

Die ursprüngliche Methode zur Sicherstellung der 1-Lipschitz-Kontinuität durch „Clipping“ der Gewichte, wie in [100] beschrieben, wurde später durch die „Gradient Penalty“-Methode in [101] von Gulrajani et al. verbessert. Diese Methode führt einen zusätzlichen Term in die Verlustfunktion ein, um die Lipschitz-Bedingung effektiver durchzusetzen:

$$L = \mathrm{E}_{\underline{z}\sim\mathrm{P}_g}\left[D\left(G\left(\underline{z}\right)\right)\right] - \mathrm{E}_{\underline{x}\sim\mathrm{P}_r}\left[D\left(\underline{x}\right)\right] + \lambda\mathrm{E}_{\underline{\tilde{x}}\sim\mathrm{P}_{\underline{\tilde{x}}}}\left[\left(\parallel \nabla_{\underline{\tilde{x}}} D\left(\underline{\tilde{x}}\right) \parallel_2 - 1\right)^2\right] \tag{7}$$

Hierbei ist λ ein Gewichtungsfaktor, der die Stärke der Strafe (Penalty) reguliert, und $\tilde{x}$ bezeichnet zufällige Stichproben, die interpolativ zwischen den Verteilungen P_g und P_r erzeugt werden.

Diese Entwicklungen ermöglichen es WGANs, ein stabiles Training zu gewährleisten und gleichzeitig qualitativ hochwertige Ergebnisse zu erzielen, indem sie die Wasserstein-Distanz nutzen, um eine effektive und theoretisch fundierte Divergenzfunktion bereitzustellen.

Conditional Generative Adversarial Networks

Ein grundlegendes Problem bei Standard-GANs ist die fehlende Kontrolle über die Spezifikation der generierten Daten $\hat{x}$. Wenn beispielsweise das Ziel ist, Bilder von Zahlen zwischen 0 und 9 zu erzeugen, kann nicht bestimmt werden, welche spezifische Zahl generiert wird. Um diese Einschränkung zu adressieren, entwickelten Mirza und Osindero das Conditional Generative Adversarial Network (cGAN), das in [102] vorgestellt wurde und eine gezielte Steuerung der generierten Daten ermöglicht.

Im Unterschied zum ursprünglichen GAN-Modell, integriert das cGAN zusätzliche bedingende Informationen (z.B. Klassenlabels), die sowohl an den Generator G als auch an den Diskriminator D übermittelt werden. Diese Modifikation erlaubt es dem Generator, Daten zu produzieren, die den spezifischen Labels entsprechen, während der Diskriminator bewertet, ob die generierten Daten nicht nur realistisch sind, sondern auch dem vorgegebenen Label adäquat entsprechen.

Die strukturelle Anpassung im cGAN wird typischerweise durch die Verkettung der Eingangsdaten mit den Labels erreicht, bevor diese in die Netzwerke eingespeist werden. Dies ermöglicht eine direkte Beeinflussung des Generierungs- und Diskriminierungsprozesses durch die bedingenden Labels. Das resultierende Minimax-Optimierungsproblem des cGAN wird formuliert als:

$$\min_G \max_D V(D,G) = \mathrm{E}_{\underline{x}\sim \mathrm{P}_r}\left[\log D\left(\underline{x}\middle|\underline{y}\right)\right] + \mathrm{E}_{\underline{z}\sim \mathrm{P}_g}\left[\log\left(1 - D\left(G\left(\underline{z}\middle|\underline{y}\right)\right)\right)\right] \qquad (8)$$

Verschiedene Studien haben unterschiedliche Methoden zur Integration der Labels untersucht, welche die Effektivität des Netzwerkes beeinflussen können. Beispielsweise:

- In [102] werden die Labels mit den Eingangsdaten vor deren Einführung in das Netzwerk verkettet.
- In [103] erfolgt die Zusammenführung der Daten und Labels erst nach mehreren separaten Schichten.
- In einem Text-zu-Bild cGAN [104], wird der Label-Vektor beim Generator an den Latentspacevector angehängt, während er beim Diskriminator zwischen den mittleren Schichten eingefügt wird.
- Eine alternative Methode wird in [105] vorgeschlagen, wo das Label durch eine spezielle Funktion am Ende des neuronalen Netzes eingebunden wird.

Perarnau et al. in [106] evaluieren die Effizienz verschiedener Einfügungspunkte des Labels innerhalb des Netzwerkes durch ein „Attribute Predictor Network“, das die Attribute y' der vom Generator erzeugten Daten bestimmt und diese mit den tatsächlichen Attributen y vergleicht. Die Ergebnisse zeigen, dass die Einführung des Labels nach der ersten Schicht die effektivste Methode darstellt, was sich in der höchsten durchschnittlichen Genauigkeit und dem besten F1-Score widerspiegelt.

2.3.1.2 Diffusionsmodelle

Ein Diffusionsmodell oder Diffusion Probabilistic Model [98] ist ein generatives Modell im Bereich des Deep Learning, das zur Datengenerierung eingesetzt wird. Diese Modelle, die ihre Ursprünge in den nicht gleichgewichtigen Thermodynamiksystemen haben, nutzen Markov-Ketten, um aus einem Diffusionsprozess eine Verteilung zu erlernen.

Das Denoising Diffusion Probabilistic Model (DDPM), eine vereinfachte Form des ursprünglichen Modells von Sohl-Dickstein et al. [98], wird hier exemplarisch erörtert. Das DDPM nutzt zwei Hauptprozesse:

Vorwärtsprozess (Diffusion): Startend mit einem Datenpunkt x_0 (z.B. ein Bild), der einer Verteilung $q(x_0)$ entstammt, wird schrittweise Rauschen hinzugefügt. Dieses Rauschen folgt einer Normalverteilung und transformiert den Datenpunkt über T Schritte in ein fast reines Rauschsignal x_T. Die Transformation wird mathematisch durch folgende Gleichungen beschrieben:

$$p_\theta(x_{0\dots T}) \coloneqq \mathrm{p}(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \;\; p_\theta(x_{t-1}|x_t) \coloneqq \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)I) \qquad (9)$$

Hierbei ist $\beta_{1\dots T}$ ein vordefinierter Varianzzeitplan.

Rückwärtsprozess (Denoising): Beginnend mit x_T, wird das Rauschen iterativ entfernt, um zum ursprünglichen Datenpunkt x_0 zurückzukehren. Dieser Prozess wird durch die Gleichung

$$p_\theta(x_{0\dots T}) \coloneqq \mathrm{p}(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \;\; p_\theta(x_{t-1}|x_t) \coloneqq \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)I) \qquad (10)$$

beschrieben, wobei $p(x_T) = \mathcal{N}(x_T; 0, I)$ und die Parameter μ_θ und Σ_θ durch ein neuronales Netz geschätzt werden.

Das Training des neuronalen Netzes erfolgt durch Minimierung des Losses zwischen dem tatsächlichen Rauschen und dem vom Netzwerk vorhergesagten Rauschen, was die Gleichung

$$L_t = E_{t,x_0,\epsilon}\left\|\epsilon - \epsilon_{\boldsymbol{\theta}}\left(\sqrt{\overline{\alpha_t}}x_0 + \sqrt{1-\overline{\alpha_t}}\epsilon, t\right)\right\|^2 \tag{11}$$

nutzt, wobei $\epsilon \sim \mathcal{N}(0, \mathrm{I})$ $\alpha_t = 1 - \beta_t$.

Das Modell wird typischerweise als ein U-Net implementiert, das verschiedene Upsampling- und Downsampling-Schichten verwendet, die durch Residual- und Self-Attention-Blöcke ergänzt werden, um die zeitlichen und räumlichen Dynamiken der Daten effektiv zu modellieren. Die Einbindung des Zeitstempels t erfolgt über Sinusoidal Positions-Embeddings, die in jedem Residualblock integriert sind, was eine genaue Rekonstruktion ermöglicht.

2.3.2 Modell-agnostische Trainingsdatengenerierung

Als Modell-agnostische Ansätze zur Trainingsdatengenerierung werden in dieser Arbeit Ansätze bezeichnet, die Daten generieren, ohne spezifisch auf die Mängel oder gelernten Features eines bestimmten Modells einzugehen. Hierbei ist das Ziel, allgemein die Diversität und Quantität der Trainingsdaten zu erhöhen, ohne sich auf die spezifische Verbesserung eines existierenden Modells zu konzentrieren.

2.3.2.1 Conditional-GAN-basierte Ansätze

In der Studie von Yang [107] wird das Potenzial des cGAN zur Generierung von Trainingsdaten für Machine-Learning-Algorithmen, spezifisch der Support Vector Machine (SVM), untersucht. Diese Untersuchung konzentriert sich auf die Analyse von Vibrationen einer Flugzeugturbine, eine Anwendung, die aufgrund der Komplexität und Variabilität der Daten eine Herausforderung darstellt. Der Einsatz des cGANs erfolgt mit dem Ziel, die Quantität und Qualität der verfügbaren Trainingsdaten für die SVM zu verbessern. Dabei adressiert das cGAN zwei Hauptanforderungen: Die generierten Daten müssen einerseits die spezifischen Stichprobengrößenanforderungen für die SVM erfüllen und andererseits das inhärente Rauschen der verschiedenen Erfassungsumgebungen effektiv glätten. Das cGAN-Modell, das in dieser Anwendung verwendet wird, besteht aus multiplen linearen Schichten sowohl im Generator als auch im Diskriminator, was es ermöglicht, komplexe Datenstrukturen zu modellieren und die charakteristischen Muster der realen Vibrationssignale zu reproduzieren. Durch die Fähigkeit des cGANs, die Verteilung der realen Daten präzise nachzuahmen, kann die SVM mit einer umfassenderen und repräsentativeren Datengrundlage trainiert werden. Dies führt zu einer signifikanten Steigerung der Klassifikationsgenauigkeit.

In dem Beitrag von Patel et al. [108] wird ein cGAN zur Generierung von Trainingsdaten verwendet. Sie setzen es ein für Automatic Modulation Classification (AMC)-Algorithmen, die in nichtkooperativen Kommunikationssystemen eingesetzt werden. Diese Systeme erfordern die Identifikation des Modulationsverfahrens ohne vorherige Kenntnisse über die verwendeten

Methoden, was eine Herausforderung darstellt, da traditionelle Techniken wie das CNN umfangreiche und vielfältige Trainingsdaten benötigen. Die Besonderheit des Ansatzes von Patel et al. liegt in der Nutzung eines cGAN, das speziell dafür konzipiert ist, realistische Modulationssignale unter Berücksichtigung verschiedener Störeinflüsse wie I/Q-Imbalance und drahtlosen Kanaleffekten zu simulieren. Das cGAN integriert diese Einflüsse direkt in den Trainingsdatengenerierungsprozess, wodurch das Training des AMC-Algorithmus effektiver gestaltet wird. Der Aufbau des cGANs umfasst einen Generator und einen Diskriminator, die hauptsächlich aus Dense Layers bestehen und LeakyReLU-Aktivierungsfunktionen nutzen. Der Generator des cGANs wandelt einen 1 x 128-dimensionalen Latenzvektor in einen 2x128 dimensionalen I/Q-Datensatz um, der dann als synthetisches Trainingsbeispiel dient. Diese generierten Datensätze erweitern effektiv den Umfang und die Diversität der Trainingsdaten für das CNN, das als Benchmark-Modell für die Modulationsklassifikation dient. Die Ergebnisse von Patel et al. zeigen, dass durch die Verwendung der cGAN-generierten Daten die Klassifikationsgenauigkeit des CNN signifikant verbessert werden kann. Dies bestätigt die Effektivität von generativen Modellen in der Erzeugung von qualitativ hochwertigen Trainingsdaten, die spezifische physikalische und technische Merkmale des Modulationsprozesses abbilden. Durch diesen Ansatz konnte nicht nur die Leistungsfähigkeit der Modulationsklassifikation erhöht, sondern auch die Robustheit des AMC-Systems unter realen, variablen Bedingungen signifikant gesteigert werden. Vergleichbar dazu haben Strelcenia und Prakoonwit [109] ebenfalls ein conditional GAN auf Kreditkartenbetrugsdatensätzen trainiert und mit den synthetischen Daten mehrere Algorithmen (ANN, Nearest-Neighbour, Random Forest, XGBoost) trainiert. Sie konnten eine Verbesserung der Klassifizierungsgenauigkeit bei allen Algorithmen durch den Einsatz der synthetischen Daten nachweisen.

2.3.2.2 Style-Transfer-basierte Ansätze

Die bisher präsentierten Arbeiten haben die synthetischen Daten von Grund auf generiert. Es existieren auch Ansätze, die bestehende Daten mithilfe von generativen Methoden in synthetische Daten umwandeln, was dem Prinzip der Augmentierung ähnelt. Beispielsweise stellen Niu et al. [110] ein Surface Defect-Generation Adversarial Network (SDGAN) vor. Dieses GAN nutzen sie, um Oberflächendefektdatensätze in industriellen Anwendungen synthetisch zu erweitern. SDGAN kann die Menge und Variabilität der Daten für das Training von Deep-Learning-Modellen zur Defekterkennung erheblich erhöhen. Hierbei werden hochwertige und vielfältige Defektbilder aus defektfreien Industriebildern generiert. Die in der Studie vorgestellten experimentellen Ergebnisse unterstreichen die Vorteile von SDGAN-erweiterten Datensätzen: So konnte eine Fehlerrate von 1,77 % bei der Erkennung von Anomalien und eine Fehlerrate von 0,74 % bei der Klassifizierung von Defekten auf gelabelten bzw. nicht gelabelten Datensätzen erreicht werden. Diese Zahlen stellen erhebliche Verbesserungen dar - 49,43 % bei der Erkennung von Anomalien und 57,47 % bei der Fehlerklassifizierung - gegenüber Modellen, die auf nicht erweiterten Datensätzen trainiert wurden. Mitri et al. [111] haben einen ähnlichen Ansatz, der die

Verwendbarkeit von StyleGAN-basierten synthetischen Datengeneratoren in einem Anwendungsfall der Automatisierungstechnik, der optischen Detektion von Schweißnähten, nachweist.

2.3.2.3 Ansätze mit Kontrolle der Datengenerierung im latenten Raum

Weiter existieren auch Ansätze, die, wie der Ansatz, der in dieser Arbeit vorgestellt ist, den latenten **z**-Vektor dazu verwenden, Einfluss auf die Generierung der Daten zu nehmen. Zhang et al. [112] verbinden das StyleGAN [113] mit differenzierbaren Grafik-Renderern. Ziel ist es, die 3D-Objektdarstellung und -Manipulationsmöglichkeiten in synthetischen Bildern zu verbessern. Die Forscher haben ein inverses neuronales Grafiknetzwerk (IGN) trainiert, das den Bildentstehungsprozess umkehrt. Diese Synergie ermöglicht die Entflechtung komplexer, hochdimensionaler Daten in besser interpretierbare und kontrollierbare 3D-Darstellungen. Die experimentellen Ergebnisse zeigen, dass dieses integrierte System ein neues Maß an Kontrollierbarkeit bei der Erzeugung von 3D-Bildern ermöglicht. Ein weiterer StyleGAN-basierter Ansatz mit dem Namen DatasetGAN [114] nutzt den hochdimensionalen latenten Raum von StyleGAN, welcher nachweislich eine latente Darstellung von Variablen wie Blickwinkel und Objektidentität enthält. Es werden eine begrenzte Anzahl von Bildern generiert, welche dann manuell annotiert werden, um als Trainingsdaten zu dienen. Diese Annotationen dienen dazu, eine Gruppe von mehrschichtigen ANNs zu trainieren, die als Style Interpreter bezeichnet werden. Die MLPs lernen, spezifische latente Merkmale mit den entsprechenden Bezeichnungen zu assoziieren. Dadurch schaffen sie ein automatisches Annotationssystem.

2.3.3 Modell-informierte Trainingsdatengenerierung

Modell-informierte Ansätze zur Trainingsdatengenerierung bezeichnen in dieser Arbeit Methoden, die speziell darauf ausgerichtet sind, Daten zu erzeugen, welche die Schwächen oder Fehlklassifikationen eines bestimmten Modells gezielt ausnutzen oder adressieren. Ziel dieser Ansätze ist es, die Leistungsfähigkeit des vorhandenen Modells durch systematische Verbesserung seiner Schwachstellen zu steigern. Dies wird erreicht, indem Daten, die vom Modell inkorrekt verarbeitet werden, identifiziert und in den Trainingsdatensatz integriert werden, um das Modell gezielt in diesen kritischen Bereichen zu trainieren. Solche Ansätze helfen nicht nur, die Robustheit des Modells gegenüber adversen Eingaben zu erhöhen, sondern optimieren auch seine Genauigkeit und Zuverlässigkeit durch erweiterte Exposition gegenüber komplexen oder marginalen Fallbeispielen, die es bisher falsch klassifiziert bzw. prädiziert hat. Das gezielte Suchen von Daten, die falsche Ausgaben hervorrufen, wird auch als Hard-Negative-Mining bezeichnet und kommt ursprünglich aus dem Bereich der Computer Vision [115]. Das Generieren von adversarialen Daten kann als Spezialfall des Hard-Negative-Mining gesehen werden, weil es nicht nur Daten generiert, die falsche Ausgaben erzeugen, sondern die Daten auch noch für den menschlichen Beobachter

nicht erkennbar anders sind. In den folgenden Kapiteln wird der Stand der Technik von Hard-Negative-Mining und Adversarialen Methoden nacheinander vorgestellt.

2.3.3.1 Hard-Negative-Mining Trainingsdatengenerierung

In [116] stellen Hughes et al. ein generatives Verfahren vor, das ein VAE verwendet, um qualitativ hochwertige Bilder zu erzeugen, die als Hard-Negativ-Samples beim Training von Klassifikationsnetzwerken dienen. Der Kern der Methodik liegt darin, mit dem VAE die latente Verteilung des z-Vektors eines GANs zu erlernen. Dadurch sind sie in der Lage, realistische, aber deutlich unterschiedliche Bilder zu erzeugen. Dieser Prozess erzeugt effektiv hart-negative Bilder, die in Aussehen und Verteilung den echten Daten sehr ähnlich sind, sich aber ausreichend unterscheiden, um die Robustheit der Klassifikationsmodelle gegenüber falsch-positiven Ergebnissen zu verbessern. Die Nützlichkeit der vorgeschlagenen Methodik wird anhand einer Fallstudie demonstriert, in der ein SAR (Synthetic Aperture Radar)-Bild mit einem optischen Bild verglichen wird. Die Autoren verwenden ein existierendes SAR-optisches Matching-Netzwerk, um die Leistungsfähigkeit ihrer Methode mit traditionellen Hart-Negativ-Mining-Techniken und einem Basismodell zu vergleichen. Die generierten SAR-Bildfelder, die sich durch eine sehr hohe Auflösung und Realitätsnähe auszeichnen, werden als harte Negative im Trainingsprozess verwendet. Die Ergebnisse der experimentellen Evaluierung zeigen, dass durch die Einbeziehung dieser generierten Hart-Negativ-Muster eine signifikante Verbesserung der Genauigkeit des SAR-optischen Matching-Netzwerkes bei gleichzeitiger Reduktion der False-Positive-Rate erzielt werden kann. Die Methode ist speziell für Bildmatching-Systeme entworfen, bei denen eine Änderung des Bildes sicher dazu führt, dass das Bild nicht mehr gematched werden darf. Die Methode ist in dieser Form nicht übertragbar für andere Problemstellungen wie die Objektdetektion.

2.3.3.2 Adversariale Trainingsdatengenerierung

Zhao et al. [117] schlagen eine neue Methode zur Erzeugung natürlicher adversarialer Beispiele vor. Dieses Framework unterscheidet sich von traditionellen Techniken zur Erzeugung von Negativbeispielen wie der FGSM, indem es im latenten Raum eines Modells und nicht direkt im Raum der Eingabedaten arbeitet. Die Methode verwendet GANs, um normalverteilte Vektoren fester Länge auf Dateninstanzen abzubilden und so eine kontinuierliche und dichte Repräsentation der Daten zu lernen. Sobald diese Abbildung erfolgt ist, werden negative Beispiele durch die Suche nach nahen Nachbarn im latenten Raum einer gegebenen Eingabeinstanz erzeugt. Bei dieser Suche werden Samples in einem immer enger werdenden Bereich um die latente Repräsentation gezogen, um Variationen zu finden, die ein Klassifikator falsch klassifiziert. Die dahinterliegende Idee ähnelt ebenfalls der Grundidee von [116] und der in dieser Arbeit vorgestellten Methodik, dass die Trainingsdaten im latenten Raum gesucht werden. Die Autoren merken allerdings an,

dass ihre Suchmethode sehr rechenzeitintensiv ist. Ob ihr Algorithmus in Anwendungsfällen wie der Objektdetektion einsetzbar ist, bleibt offen.

In [118] wird die Verwendung generativer Modelle zur künstlichen Erweiterung von Trainingsdatensätzen untersucht, um die Robustheit gegenüber adversarialen Störungen zu verbessern. Neuere Studien haben gezeigt, dass für robustes Training in der Regel größere Datensätze als für Standardklassifikationsaufgaben benötigt werden, was zu einer Lücke in der Robustheit und Genauigkeit führt, wenn nur die Originaldatensätze verwendet werden. Die Autoren schlagen vor, dass diese Lücke geschlossen werden kann, indem das Training durch zusätzliche synthetische Daten ergänzt wird, die aus den Originaldatensätzen generiert werden. Die Studie identifiziert spezifische Bedingungen, unter denen die Einbeziehung generierter Daten vorteilhaft ist, und zeigt signifikante Verbesserungen in der robusten Genauigkeit, wodurch die Lücke in der robusten Genauigkeit im Vergleich zu Modellen, die mit zusätzlichen realen Daten aus dem "80 Million Tiny Images"-Datensatz trainiert wurden, verringert wird.

Wang et al. [119] haben untersucht, ob Fortschritte bei den Diffusionsmodellen das adversariale Training weiter verbessern können. Dazu wird das zum Zeitpunkt der Anfertigung der Arbeit beste verfügbare Diffusionsmodell [120] verwendet, das sich im Vergleich zu früheren DDPM durch höhere Effizienz und bessere Bildqualität auszeichnet. Das neue Modell benötigt etwa 20 Abtastschritte und erreicht eine geringere Fréchet Inception Distance (FID), was auf eine höhere Qualität der erzeugten Bilder hinweist. Mit dem neuen Modell konnten sie synthetische Daten generieren, die die robuste Genauigkeit auf den CIFAR-Datensätzen um bis zu 4 % erhöht hat.

Rade et al. [121] haben die Genauigkeits-Robustheit-Abwägung genauer untersucht und eine Methodik entwickelt, mit der robuste Modelle eine höhere Genauigkeit erzielen können. So erreichten sie, dass die Genauigkeit auf CIFAR10 nur noch von 84,9 auf 49,08 durch den Einsatz von adversarialem Training absinkt.

In diesem Kapitel wurden bestehende Ansätze zur Trainingsdatengenerierung nach deren Generierungsmethode strukturiert und beschrieben. Im folgenden Kapitel wird bewertet, inwieweit die Ansätze die Problemstellung dieser Arbeit adressieren.

2.4 Bewertung der bestehenden Ansätze

Die in den Kapiteln 2.1 bis 2.3 vorgestellten Methodiken zur Trainingsdatengenerierung haben sehr unterschiedliche Ansätze, typische Anwendungsbereiche, Vor- und Nachteile in der Verwendung. Um die Ansätze vergleichen zu können, werden die folgenden Kriterien und Bewertungsstufen für relevant erachtet. Die Kriterien basieren auf den hergeleiteten Teilzielen dieser Arbeit.

2.4.1 Bewertungskriterien

Datenqualität

-- (Schlechte Qualität): Die erzeugten Daten sind weder realistisch noch relevant. Es gibt nur eine minimale Vielfalt, wobei viele generierte Proben identisch oder irrelevant erscheinen, was das Training von Modellen irreführen könnte.
- (Unterdurchschnittliche Qualität): Die Daten weisen einen gewissen Grad an Realismus und Relevanz auf, sind aber nicht ausreichend vielfältig. Einige Proben repräsentieren möglicherweise nicht genau die realen Variationen.
0 (Mittlere Qualität): Die Daten sind einigermaßen realistisch, relevant und weisen ein akzeptables Maß an Vielfalt auf. Die meisten Stichproben sind nützlich, auch wenn die Wiedergabetreue verbessert werden könnte.
+ (Gute Qualität): Die erzeugten Daten sind recht realistisch, relevant und vielfältig. Gelegentlich gibt es Probleme mit sehr spezifischen Szenarien oder kleinere Ungenauigkeiten in der Darstellung.
++ (Ausgezeichnete Qualität): Die Daten sind sehr realistisch, sehr relevant und weisen eine große Vielfalt auf. Sie kommen den realen Daten in jeder Hinsicht sehr nahe und bieten eine optimale Grundlage für das Training von Modellen.

Skalierbarkeit auf große Datenmengen

-- (sehr ineffizient): Die Methode ist rechenintensiv, langsam und skaliert nicht gut mit steigenden Datenanforderungen. Sie verbraucht beträchtliche Ressourcen und ist daher für großangelegte Anwendungen ungeeignet.
- (Ineffizient): Die Methode hat spürbare Einschränkungen in Bezug auf Geschwindigkeit und Skalierbarkeit und erfordert erhebliche Rechenressourcen, die eine Einschränkung darstellen könnten.
0 (Mäßig effizient): Die Methode stellt ein gutes Gleichgewicht zwischen Rechenaufwand und Geschwindigkeit her und bietet einen Mittelweg zwischen Ressourcenverbrauch und Geschwindigkeit der Datenerzeugung.

+ **(Effizient)**: Die Methode ist schnell und lässt sich gut skalieren, ohne dass eine übermäßige Zunahme der Rechenressourcen erforderlich ist. Sie ist für größere Datensätze geeignet, kann aber geringfügige Einschränkungen aufweisen.
++ **(Hocheffizient)**: Die Methode bietet eine hervorragende Skalierbarkeit und Geschwindigkeit bei minimalen Rechenkosten und ist ideal für die Generierung sehr großer Datenmengen ohne Ressourcenbeschränkungen.

Übertragbarkeit in unterschiedlichen Domänen

-- (Nicht vielseitig): Die Methode ist sehr spezifisch für einen Bereich oder eine Aufgabe und erfordert eine umfangreiche Neukonfiguration, um sie an andere Bereiche oder Anforderungen anzupassen.
- **(Geringfügig vielseitig)**: Die Methode kann mühsam an andere Bereiche oder Aufgaben angepasst werden, ist aber nicht von Natur aus flexibel. Sie erfordert oft erhebliche Änderungen oder Kompromisse bei der Leistung.
0 (Mäßig vielseitig): Die Methode lässt sich mit einigen Anpassungen an verschiedene Bereiche oder Aufgabenstellungen anpassen. Sie bietet eine annehmbare Flexibilität ohne umfangreiche Neuentwicklungen.
+ **(Vielseitig)**: Die Methode passt sich gut an verschiedene Bereiche und Aufgaben an und erfordert nur minimale Anpassungen. Sie bietet in allen Szenarien eine adäquate Leistung.
++ **(Äußerst vielseitig)**: Die Methode ist extrem anpassungsfähig und kann ohne Leistungseinbußen und mit minimalem Anpassungsaufwand nahtlos in verschiedene Bereiche und Aufgaben übergehen.

Einfluss auf die Modellgenauigkeit bei Verwendung der Trainingsdaten

-- (Verschlechtert die Leistung): Die Verwendung der generierten Daten verschlechtert die Leistung des Modells.
- **(Minimale Auswirkung)**: Die generierten Daten haben einen minimalen positiven Einfluss auf die Modellleistung. Die Verbesserungen sind kaum spürbar und möglicherweise statistisch nicht signifikant.
0 (Mäßige Auswirkung): Die Verwendung der generierten Daten führt zu einer spürbaren Verbesserung der Modellleistung, auch wenn es einige Leistungsbereiche geben kann, die nicht so stark davon profitieren.
+ **(Signifikante Auswirkung)**: Die Methode verbessert die Modellleistung in den meisten Bereichen erheblich und trägt zu einer besseren Generalisierung bei.
++ **(Hohe Wirkung)**: Die Methode verbessert die Modellleistung dramatisch, übertrifft andere Methoden deutlich und steigert die Genauigkeit, Verallgemeinerung und Robustheit erheblich.

2.4.2 Durchführung der Bewertung

In diesem Kapitel wird die Bewertung der in Kapitel 2.1 bis 2.3 vorgestellten Methoden zur Generierung von Trainingsdaten anhand der im vorherigen Abschnitt definierten Bewertungskriterien durchgeführt. Die Tabelle 2.1 enthält eine Übersicht über die einzelnen Bewertungen, die genauen Begründungen für diese Bewertungen sind in den folgenden Abschnitten ausgeführt.

2.4.2.1 Bewertung der Augmentierungsmethoden

Die Augmentierungsmethoden, die in Kapitel 2.1 erörtert sind, lassen sich in drei Unterkategorien gliedern: Eingabedaten- ([24]-[37]), Feature- ([38]-[43])) und adversariale ([22], [45]-[49]) Augmentierungen. Die Qualität der durch diese Methoden generierten Daten variiert erheblich. Einige grundlegende Techniken der Eingabedatenaugmentierung, wie das Spiegeln von Bildern, erzielen eine hohe Datenqualität, stellen jedoch innerhalb dieses Feldes eine Ausnahme dar. Häufig angewendete Verfahren wie zufälliges Zuschneiden (Cropping) oder Stauchung können, abhängig vom Originalbild, zu Ergebnissen führen, die stark von der Realität abweichen. Feature-basierte Augmentierungen neigen dazu, Bilder zu produzieren, die eher Collagen ähneln als realistisch aufgenommenen Fotografien. Adversariale Augmentierungen, obwohl sie dem menschlichen Auge realistisch erscheinen mögen, weisen auf der Pixelebene signifikante unrealistische Pixelartefakte auf.

Die Skalierbarkeit dieser Methoden ist generell als hoch bis sehr hoch einzustufen, da die zugrundeliegenden Algorithmen meistens schnell sind und effizient in Datenverarbeitungspipelines integriert werden können. Die Übertragbarkeit dieser Techniken auf andere Domänen gestaltet sich jedoch als schwierig. Die meisten Augmentierungsverfahren sind spezifisch für einen bestimmten Datentyp entwickelt und nur in begrenztem Umfang auf andere Datentypen anwendbar.

Die in Kapitel 2.1 besprochenen Veröffentlichungen zeigen, dass mit allen nicht adversarialen Augmentierungsverfahren Trainingsdaten generiert werden können, die die Modellgenauigkeit signifikant verbessern. Adversariale Verfahren führen zu einer drastischen Verschlechterung der normalen Genauigkeitsmetriken, dafür führen sie zu Verbesserungen bei Robustheitsmetriken, die in dieser Arbeit nicht betrachtet werden.

2.4.2.2 Bewertung der Simulationsmethoden

Die in Kapitel 2.2 diskutierten simulationsbasierten Methoden umfassen zum Teil seit Jahrzehnten etablierte industrielle Werkzeuge, die hochwertige Daten durch physikalische Modellierungen simulieren. Diese Daten erreichen eine Realitätsnähe, die spezifisch für den vorgesehenen Anwendungszweck der Simulation ausreichend ist, der in der Regel nicht das Generieren von Trainingsdaten umfasst. Daher mangelt es den Daten oft an dem Rauschen und den hohen

Detailgraden, die typischerweise reale Daten charakterisieren. Eine Verbesserung der Detailgenauigkeit ist prinzipiell möglich, führt jedoch zu erheblich längeren Simulationszeiten, was wiederum die Skalierbarkeit beeinträchtigt.

Die Verwendung dieser Simulationswerkzeuge ist für große Datenmengen rechenintensiv und diese sind standardmäßig nicht in Datenverarbeitungspipelines integriert. Eine Übertragbarkeit der Simulationsmethoden auf andere Domänen ist generell nicht gegeben, da die Werkzeuge immer spezifisch für eine bestimmte Domäne entwickelt werden müssen [61] [62] [67]. Die in Kapitel 2.2 diskutierten Studien belegen jedoch, dass simulativ erzeugte Trainingsdaten die Genauigkeit von Modellen in vielen Anwendungsfällen signifikant steigern können.

2.4.2.3 Bewertung der Trainingsdatengenerierung durch generative Modelle

Je nach Ansatz der in Kapitel 2.3 vorgestellten generativen Modelle schwankt die Datenqualität erheblich. Die älteren generativen Methoden, wie VAE, erzeugten Daten von merklich niedriger Qualität. Mit der Entwicklung von GANs und Diffusionsmodellen ist jedoch die Generierung von Daten hoher Qualität möglich geworden. Obwohl das Training dieser Modelle anfänglich einen erheblichen Aufwand darstellt, ermöglicht es nach Abschluss die Produktion umfangreicher Datensätze mit überschaubarem zusätzlichem Aufwand. Da diese Modelle in verbreiteten Frameworks implementiert sind, lassen sie sich auch effektiv in bestehende Datenpipelines integrieren.

Die existierenden modell-informierten Ansätze skalieren nicht auf die gleiche Weise wie modellagnostische Verfahren, da sie oft das vorhandene Modell iterativ aufrufen müssen, was den Generierungsprozess verlangsamt und die Skalierbarkeit limitiert. Im Gegensatz dazu sind generative Modelle hervorragend auf verschiedene Domänen übertragbar, da sie universell einsetzbare Methoden darstellen. Hierbei ist die spezifische Domäne oft irrelevant, vielmehr sind die Art der Problemstellung und der Datentyp (z. B. Zeitreihen, unstrukturierte Daten) von Bedeutung. Die in Kapitel 2.3 referenzierten Studien belegen, dass durch generative Modelle erzeugte Trainingsdaten in vielen Fällen die Modellgenauigkeit signifikant steigern können.

Ein besonderer Fall sind generative Methoden, die adversariale Daten erzeugen. Diese sind qualitativ überlegen im Vergleich zu den durch Augmentierung produzierten Daten, jedoch verschlechtern sie im Allgemeinen weiterhin die herkömmlichen Genauigkeitsmetriken.

Tabelle 2.1: Übersicht über die Bewertung von Ansätzen zur Generierung von Trainingsdaten

Methode	**Datenqualität**	**Skalierbarkeit**	**Übertragbarkeit**	**Einfluss auf Modellgenauigkeit**
Augmentierung				
Eingabedaten [24]-[37]	0	++	-	+
Feature [38]-[43]	0	+	--	+
Adversarial [22], [45]-[49]	0	+	-	--
Simulation				
Umgebung [50]-[66]	+	-	--	+
Bilddaten [67]-[85]	+	-	--	+
Funkdaten [86]-[89]	+	-	--	+
Generative Modelle				
Modell-agnostisch [103]-[110]	++	+	+	+
Modell-informiert [111]-[117]	++	-	+	+*

2.5 Aufzeigen des Forschungsbedarfs

In Kapitel 1.2 wurde die Problemstellung dieser Arbeit herausgearbeitet und folgende Herausforderungen identifiziert. Bedingt durch die Notwendigkeit spezialisierter Ausrüstung, deren Integration in die Produktionsumgebung und des Labelingaufwands ist das Sammeln realer Daten in der Automatisierungstechnik aufwändig (**H1**). Zudem ist oft unklar, welche spezifischen Daten bzw. Szenarien erforderlich sind, um ein Modell zu verbessern (**H2**). Viele Automatisierungssysteme operieren unter einzigartigen Bedingungen, die spezifische Daten erfordern, die in bestehenden Datensätzen oft nicht vorhanden sind (**H3**).

Aus diesen Herausforderungen sind in Kapitel 1.3 die Zielsetzungen an ein Konzept hergeleitet worden, das diesen Herausforderungen begegnet und die Problemstellung adressiert, wie die notwendigen gelabelten Datensätze erstellt werden können, damit trainierte Modelle in der Automatisierungstechnik verbessert und damit auch kosteneffizienter eingesetzt werden können. So müssen Trainingsdaten möglichst automatisch generiert werden, um logistische und finanzielle Belastungen zu minimieren und den Entwicklungsprozess der Modelle zu beschleunigen (Teilziel 1). Die Daten müssen realistisch sein und die realen Betriebsbedingungen genau widerspiegeln, um das Training von Modellen mit ausreichend guter Genauigkeit im Einsatz zu ermöglichen (Teilziel 2). Außerdem müssen die Daten gezielt erzeugt werden, um das Modell zu verbessern und Ressourcen effizient zu nutzen und dabei das Risiko von Overfitting durch irrelevante Daten zu minimieren (Teilziel 3). Zudem müssen die Generierungsmethoden flexibel und anpassungsfähig sein, um verschiedene Datentypen und Einsatzszenarien abdecken zu können, die in der Automatisierungstechnik benötigt werden (Teilziel 4).

In den folgenden Absätzen wird die Bewertung des Standes der Technik aus Kapitel 2.4.2 (Tabelle 2.1) mit den Teilzielen verglichen und so die Forschungslücke dieser Arbeit auf Basis der bewerteten Literatur hergeleitet.

Teilziel 1: Automatische Trainingsdatengenerierung
Alle in Kapitel 2 vorgestellten Methoden können prinzipiell Trainingsdaten generieren, diese haben zwar unterschiedliche Voraussetzungen an die Datentypen, für die sie anwendbar sind, aber prinzipiell ist es mit all diesen Ansätzen möglich automatisiert Trainingsdaten zu generieren. Das Teilziel 1 wird also von mehreren Methoden erfüllt.

Teilziel 2: Realitätsnähe der Daten
Die Realitätsnähe der Daten, die durch die betrachteten Methoden generiert werden, lässt sich auf unterschiedliche Weisen betrachten, einerseits die reine Qualität der Daten, wie sie von einem menschlichen Experten eingeschätzt werden kann und andererseits, wie sich die Verwendung der

Daten auf die Modellgenauigkeit auswirkt. In Kapitel 2.4 sind beide Aspekte getrennt bewertet und es zeigt sich, dass die Augmentierungsverfahren und die Methoden zur Generierung von adversarialen Daten dabei oft sehr unterschiedlich abschneiden. Da die Qualität der Daten aus der Sicht eines menschlichen Experten für den erfolgreichen Einsatz eines trainierten Modells eher eine untergeordnete Rolle spielt, im Vergleich zu dessen Genauigkeit sieht der Autor Teilziel 2 von allen Methoden erfüllt, die keine adversarialen Daten generieren.

Teilziel 3: Generierte Trainingsdaten verbessern das Modell

Die meisten untersuchten Methoden aus dem Bereich der Augmentierung gehen nicht gezielt so vor, dass die generierten Daten das Modell verbessern. Eine Ausnahme stellen die Methoden zur Generierung von adversarialen Beispielen dar, sie verwenden ein existierendes Modell für eine gezielte Generierung von Daten. Auch die simulativen Methoden gehen nicht gezielt vor. Sie können zwar dazu verwendet werden, um bekannte Lücken in Trainingsdatensätzen zu füllen, was aus Sicht des Autors aber nicht ausreichend ist, um Teilziel 3 zu erfüllen, weil das gezielte Generieren direkt Teil der Methode sein muss, damit ein hoher Automatisierungsgrad erreicht werden kann**.** Es ist auch möglich Simulatoren durch einen Model-in-the-Loop-Ansatz in ein automatisiertes Datengenerierungssystem einzubinden, wie durch den Autor in [84] (Eigene Vorarbeit) gezeigt, was allerdings zu extrem langen Rechenzeiten führt, sodass ein praktischer Einsatz solcher Systeme unpraktikabel ist. Auch die generativen Methoden werden größtenteils ungezielt eingesetzt. Ausnahmen bilden hierbei die Ansätze zum Generieren von adversarialen Daten und die Anwendung des Hard-Negative-Mining von Hughes et al. [116], die das Teilziel 3 damit erfüllen.

Teilziel 4: Übertragbarkeit für verschiedene Datenarten

Die Analyse der bestehenden Ansätze in Kapitel 2.4 (Tabelle 2.1) hat ergeben, dass nur die trainierten generativen Modelle eine Übertragbarkeit für verschiedene Datentypen ermöglichen und damit Teilziel 4 prinzipiell erfüllen. Der einzige Ansatz der damit infrage kommt alle Teilziele zu erreichen, ist das Hard-negative Mining von Hughes et al. [116], weil er trainierte generative Modelle verwendet und gezielt Daten generiert, die Ausgabefehler in einem existierenden Modell erzeugen. Allerdings funktioniert der vorgestellte Ansatz nur beim Bild-Matching, weil jede nicht erkannte Änderung bei den generierten Bildern implizit ein Fehler des existierenden Netzes darstellt. Der Ansatz lässt sich also weder auf andere Datentypen noch weitere Problemformulierungen übertragen und erfüllt deshalb Teilziel 4 nicht.

Durch den Abgleich der Bewertung des Standes der Technik mit den Teilzielen kann festgestellt werden, dass keiner der existierenden Ansätze alle Teilziele erreicht und eine Forschungslücke besteht. Diese Forschungslücke lässt sich mit der folgenden Forschungsfrage beschreiben:

Wie kann ein Ansatz aufgebaut sein, der es erlaubt, automatisiert leistungssteigernde Trainingsdaten für unterschiedliche Datentypen und Problemstellungen in der Automatisierungstechnik zu generieren?

Im nachfolgenden Kapitel 3 ist das Konzept dieser Arbeit beschrieben, das diese Frage beantwortet und die Zielsetzungen erfüllt.

3 Konzept zur leistungssteigernden Trainingsdatengenerierung durch generative Modelle

Dieses Kapitel stellt das Konzept, den Kern dieser Arbeit, vor. Zunächst ist die Motivation der Konzeptidee vorgestellt, dann wird ein Überblick über alle Teilkomponenten gegeben, die dann im letzten Abschnitt dieses Kapitels erklärt sind.

3.1 Motivation der Konzeptidee

Das vorgestellte Konzept ist inspiriert durch die Technik der Datenaugmentation. Hierbei handelt es sich um einen Prozess, bei dem bestehende Trainingsdaten transformiert und in neue Trainingsdaten umgewandelt werden. Diese Veränderungen erfolgen unter strenger Beachtung eines entscheidenden Aspektes: Nach der Durchführung der Datentransformation sollte das ursprüngliche Label entweder identisch bleiben oder zumindest berechenbar sein. Wäre dies nicht gewährleistet, könnten die neu generierten Daten das Training unweigerlich beeinträchtigen, indem sie irreführende oder gar falsche Trainingsinformationen einspeisen.

Ein prägnantes Beispiel hierfür ist ein vertikal gespiegeltes Bild: Nach der Spiegelung befinden sich die Objekte auf dem Bild an einer anderen Position. Allerdings kann die neue Position auf unkomplizierte Weise berechnet werden, indem einfach die Koordinaten ebenfalls gespiegelt werden. Diese besondere Form der Datentransformation wird als "labelerhaltende Datenmodifikation" bezeichnet, da bisher keine verbreitete Bezeichnung in der Literatur für diese Eigenschaft existiert.

Bei Bilddaten fällt die Entscheidung in der Regel leichter, welche Augmentationstechniken genutzt werden können, ohne das zugrundeliegende Label zu verändern. Der Grund hierfür ist, dass wir Menschen eine intuitive Fähigkeit besitzen, Bilder zu verstehen. Bei den meisten Daten in der Automatisierungstechnik - wie beispielsweise Zeitreihen von Maschinensensoren oder Funksignalen - existieren jedoch nur wenige Augmentationstechniken, die ohne eine Änderung der Label angewendet werden können.

Um diesem Mangel zu begegnen, schlagen wir das Konzept zur leistungssteigernden und automatisierten Trainingsdatengenerierung vor. Dieses Konzept ermöglicht es, neue Daten zu generieren, und zwar ohne das ursprüngliche Label zu verändern. Dadurch sollen neue und qualitativ hochwertige Trainingsdaten generiert werden.

3.2 Überblick über das Gesamtkonzept

Abbildung 3.1 zeigt den grundlegenden Aufbau des in dieser Arbeit vorgeschlagenen Konzepts. Im Kern handelt es sich um einen iterativen Datengenerator, der synthetische Daten zur Leistungssteigerung neuronaler Netze erzeugt. Leistungssteigernd bedeutet, dass die generierten Daten einen erhöhten Nutzen für das Training haben im Vergleich zu zufällig generierten Daten. In diesem Kapitel werden zunächst die Voraussetzungen erklärt, die benötigt werden, damit der Trainingsdatengenerator realisiert werden kann. Danach werden die Komponenten des leistungssteigernden, automatisierten Trainingsdatengenerators vorgestellt. Zuletzt werden die Ausgaben des Datengenerators besprochen.

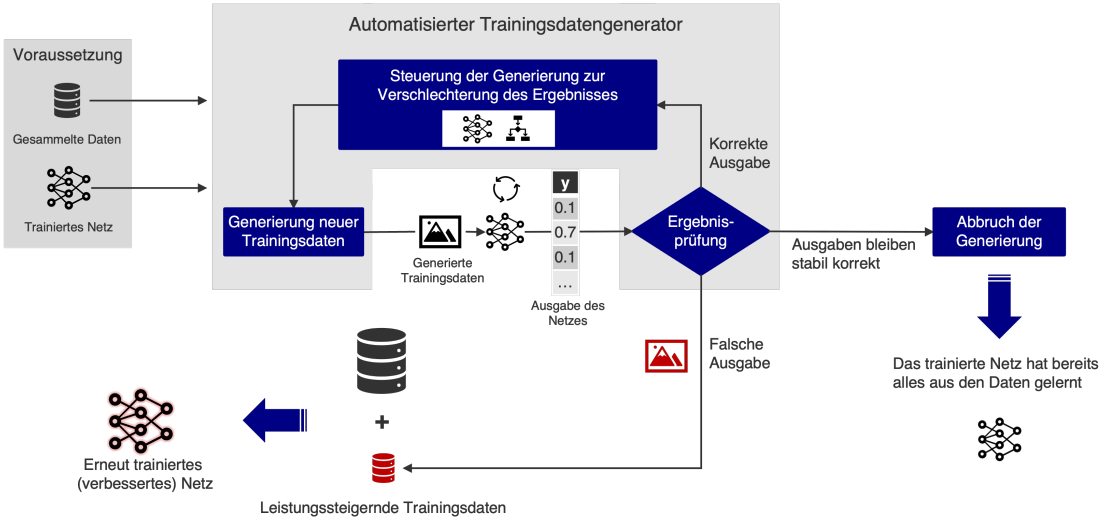

Abbildung 3.1: Das Konzept zur leistungssteigernden und automatisierten Trainingsdatengenerierung für das maschinelle Lernen im Gesamten dargestellt.

Voraussetzungen: Für die erfolgreiche Implementierung des Trainingsdatengenerators müssen bestimmte Grundvoraussetzungen erfüllt sein. Die erste Bedingung bezieht sich auf das Vorhandensein einer ausreichenden Menge an echten, qualitativ hochwertigen und korrekt gelabelten Daten. Die benötigte Menge kann hierbei nicht pauschal festgelegt werden, da sie stark von der Art der Daten, der konkreten Realisierung und insbesondere der Komplexität des Problems abhängt. Grundsätzlich lässt sich jedoch festhalten, dass die Datenmenge ausreichen muss, um ein konditioniertes, generatives neuronales Netzwerk zu trainieren, das in der Lage ist, realistische Daten für das entsprechende Szenario zu erzeugen.

Die zweite Voraussetzung ist das Vorhandensein eines bereits mit den existierenden Daten trainierten neuronalen Netzwerks. Der Datengenerator passt sich an die Güte des Netzes an. Ist das Netzwerk bereits von hoher Qualität, dann werden die vom Generator erzeugten Daten trotzdem

eine Herausforderung für dieses Netzwerk darstellen. Wenn die Netzwerkqualität bereits so hoch ist, dass sie für dessen Anwendungsfall ausreichend ist, dann werden keine weiteren synthetischen Trainingsdaten benötigt und der Trainingsdatengenerator sollte nicht realisiert werden. Bei einem Netzwerk von geringerer Qualität hingegen, könnte es sein, dass die zusätzlichen Trainingsdaten dieses erst auf ein Leistungsniveau heben, welches theoretisch auch ohne die synthetischen Daten erreichbar gewesen wäre. Für den Anwender ist es daher vorteilhafter, wenn das bereits trainierte Modell von Anfang an eine erhöhte Qualität aufweist. Dennoch ist dies für das zugrunde liegende Konzept nicht zwingend notwendig.

Leistungssteigernder, automatisierter Datengenerator: Diese Komponente ist strukturell aus drei sich ergänzenden Modulen zusammengesetzt, nämlich dem Generatormodul, dem Steuerungsmodul und dem Modul zur Ergebnisprüfung. Diese Module sind in einem zyklischen Kontext miteinander verknüpft und werden sequenziell aktiviert.

Initiiert wird der Prozess durch das Generatormodul, das die primäre Aufgabe der Datensynthese übernimmt. Die synthetisierten Daten werden anschließend an das bestehende neuronale Netzwerk weitergeleitet, was als eine essenzielle Voraussetzung des vorliegenden Konzepts fungiert.

Die generierte Ausgabe des neuronalen Netzwerks wird im Anschluss durch das Modul zur Ergebnisprüfung einer Überprüfung unterzogen. Basierend auf dieser Evaluation generiert das Steuerungsmodul neue Parameter, welche dem Generatormodul zugeführt werden.

Die genauen Mechanismen und Prinzipien des zielorientierten, automatisierten Datengenerators werden im nächsten Kapitel umfassender und detaillierter beleuchtet.

Ausgaben des Datengenerators: Der vorgestellte Datengenerator kann zwei unterschiedliche Typen von Ausgaben erzeugen. Im optimalen Fall gelingt es dem Datengenerator, neue, herausfordernde Daten für das existierende neuronale Netzwerk zu produzieren. In diesem Szenario stellen die neu generierten Daten die Ausgaben des Generators dar.

Sollte es jedoch dem Datengenerator nicht gelingen, entsprechend herausfordernde Daten zu synthetisieren, wird der Generator seinen Programmdurchlauf abbrechen. Als Konsequenz hiervon wird eine Benachrichtigung an den Nutzer ausgegeben, die darüber informiert, dass die Generierung von neuen, herausfordernden Daten nicht erfolgreich war.

3.3 Entwurf des leistungssteigernden Trainingsdatengenerators

In diesem Kapitel werden die konzeptionellen Bestandteile des Trainingsdatengenerators beschrieben. Abbildung 3.2 zeigt schematisch den Aufbau des Generators mit seinen drei Teilkomponenten. Dies sind das generative neuronale Netz, die Ergebnisprüfung und die Steuerungskomponente. Welche Funktion diese Komponenten haben und wie sie miteinander interagieren, wird im Folgenden erklärt.

3.3.1 Generatives neuronales Netz

Das generative neuronale Netz stellt die Kernkomponente zur Datensynthese innerhalb des vorgestellten Gesamtkonzepts dar. Dieses Netz wird auf Basis eines existierenden Trainingsdatensatzes - einer essenziellen Voraussetzung für dieses Konzept - darauf trainiert, Daten zu generieren. Es existiert eine breite Palette verschiedener Architekturen, die in der Lage sind, unterschiedliche Arten von Daten mit variierenden Komplexitätsgraden zu erzeugen. Der aktuelle Stand der Technik im Bereich der generativen neuronalen Netzwerke ist in Kapitel 2.3.1 behandelt.

Generative Netze basieren bei der Datensynthese auf einem latenten Zufallsvektor. Die Verwendung dieses Vektors (z) führt dazu, dass die erzeugten Daten variieren. "Latent" in diesem Zusammenhang bedeutet, dass seine Elemente keinen direkten, zuordenbaren Einfluss auf die

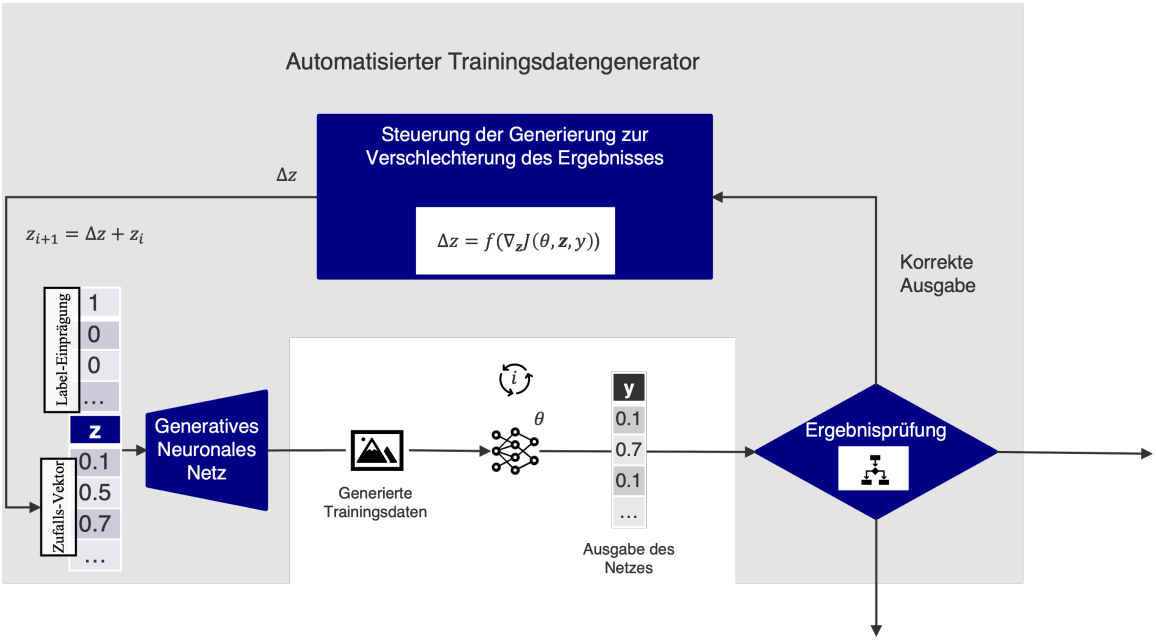

Abbildung 3.2: Aufbau des leistungssteigernden, automatisierten Trainingsdatengenerators

generierten Daten haben. Der Einfluss kann somit als verborgen oder versteckt betrachtet werden. Der Zufallsvektor kann daher vom Benutzer des Generators nicht dazu verwendet werden, die Datengenerierung direkt zu steuern.

Für die Generierung von Trainingsdaten ist jedoch eine gewisse Kontrollmöglichkeit über die generierten Daten erforderlich - insbesondere hinsichtlich des Labels, das die Daten tragen. Ohne die entsprechenden Labels sind die Trainingsdaten für das Training von überwachten Modellen ungeeignet.

Hier kommen die sogenannten konditionalen generativen neuronalen Netze ins Spiel. Sie stellen eine Unterklasse der generativen neuronalen Netze dar und ermöglichen es, das Label der zu generierenden Daten festzulegen. Während des Trainings der conditional Netze wird das gewünschte Label in dem Latentvektor des Generators eingeprägt, welches generiert werden soll. In der Inferenzphase kann das Label dann gesetzt werden, um zu kontrollieren, welches Label der generierte Datenpunkt hat.

Innerhalb des automatisierten Datengenerators werden conditional neuronale Netze verwendet, wie anhand der Label-Prägung in Abbildung 3.2 erkennbar ist. Die tatsächlich verwendete Architektur des Netzes ist für das Konzept nicht von primärer Bedeutung und muss an den jeweiligen Datentyp angepasst werden. Die generierten oder gesammelten Daten werden in das bereits existierende neuronale Netzwerk eingespeist, was eine grundlegende Voraussetzung für dieses Verfahren ist. Das Netzwerk transformiert diese Eingangsdaten, basierend auf seinen intern trainierten Parametern θ, in ein spezifisches Ausgabeformat. Dieses Ausgabeformat hängt stark von der Architektur des Netzwerks und dessen beabsichtigtem Zweck ab.

Handelt es sich bei dem Netzwerk um ein Klassifikationsmodell, so ist die Ausgabe üblicherweise ein Vektor, der Pseudo-Wahrscheinlichkeiten für jede potenzielle Kategorie enthält. Dieser Vektor weist jeder möglichen Klasse einen Wahrscheinlichkeitswert zu, wobei die Summe aller Wahrscheinlichkeiten 1 entspricht. Dies repräsentiert die Sicherheit des Modells, dass die Eingangsdaten zu den jeweiligen Klassen gehören.

Im Fall eines Regressionsnetzwerks besteht die Ausgabe aus einem Vektor von kontinuierlichen Werten. Diese Werte repräsentieren die Vorhersagen des Netzwerks für die gegebene Eingabe und sind nicht durch ein Intervall wie [0,1] begrenzt. Es gibt auch Netzwerkarchitekturen, die beide Aufgaben – Klassifikation und Regression – simultan erfüllen können. Ein solches Modell könnte etwa den Ort und die Art eines Objekts auf einem Foto gleichzeitig bestimmen.

Das hier beschriebene Konzept ist flexibel und kompatibel mit unterschiedlichen Netzwerkarchitekturen, einschließlich Klassifizierungs-, Regressions- und kombinierten Architekturen. Die diversen Ausgabeformate dieser Netzwerke werden in der anschließenden

Ergebnisüberprüfungskomponente verarbeitet und analysiert. Im Folgenden wird diese Komponente detaillierter erläutert.

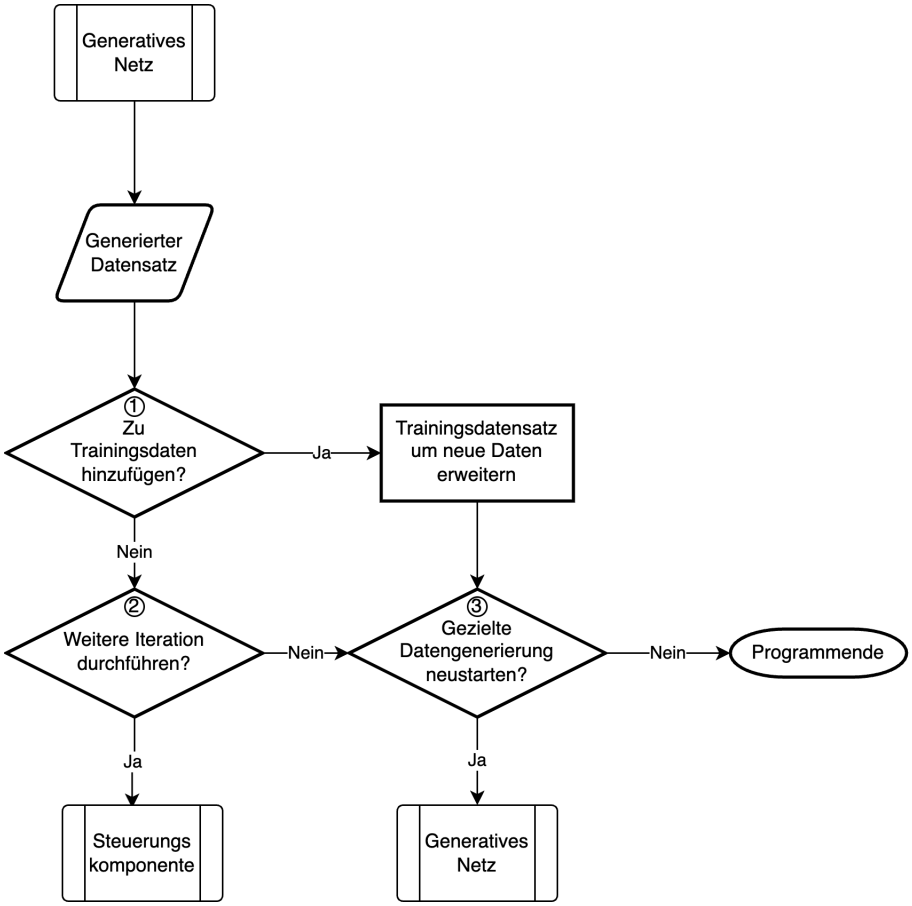

Abbildung 3.3: Ablaufdiagramm der Ergebnisüberprüfungskomponente. Kern der Komponente sind die drei nummerierten Entscheidungen, die den Programmablauf kontrollieren und beenden.

3.3.2 Ergebnisüberprüfungskomponente

Die Ergebnisprüfungskomponente erhält die Ausgabe des bestehenden neuronalen Netzes, das korrekte Label, das bei der Generierung der Eingabe des bestehenden neuronalen Netzes eingeprägt wurde und die Information, wie viele Iterationen i des gezielten Datengenerators bereits durchgeführt wurden. Sie trifft dann die Entscheidung, ob das generierte Datum zu dem neuen Datensatz der „Gezielt generierten Trainingsdaten“ hinzugefügt wird oder nicht. Wenn nicht, dann muss entschieden werden, ob eine weitere Iteration durchgeführt wird, oder die Generierung abgebrochen wird. Der Ablauf der Entscheidungsfindung ist in Abbildung 3.3 visualisiert. Die Entscheidungen sind im Folgenden nacheinander erklärt.

Entscheidung 1: Hinzufügen der Daten zum gezielt generierten Trainingsdatensatz

Die neu generierten Daten sollen dem Datensatz hinzugefügt werden, wenn mit hoher Wahrscheinlichkeit davon ausgegangen werden kann, dass sich das bestehende neuronale Netz durch ein Training mit dem neu generierten Datenpunkt verbessert. In Abbildung 3.1 ist dies abstrakt damit beschrieben, dass der Datenpunkt im bestehenden neuronalen Netz eine falsche Ausgabe hervorruft.

Bei einem Klassifikationsproblem würde das im engen Sinn umgesetzt bedeuten, dass jeder Datenpunkt hinzugefügt wird, weil die Ausgaben von neuronalen Netzen nie gänzlich korrekt sind. Label sind eindeutig vergeben – eine Person auf einem Bild ist entweder vorhanden oder nicht. Die Ausgabe eines Netzes enthält immer Unsicherheiten. Daher wird bei Klassifikationsproblemen zunächst der höchste Wert des Ausgabevektors als Vorhersage interpretiert und für den Vergleich genutzt. Allerdings ist auch das keine perfekte Lösung, wie in einem vereinfachten Beispiel von Binärklassifikation erläutert werden soll: Wenn ein Binärklassifikator Klasse A mit 51 % korrekt klassifiziert, dann würde ein einfacher Vergleich ergeben, dass der Klassifikator nichts von dem Datenpunkt lernen könnte, was falsch wäre. Der Unterschied zwischen A und B liegt nur bei 2 %, was sehr gering ist. Das neuronale Netz könnte allerdings von diesem Datenpunkt lernen, die Klassen A und B klarer auseinanderhalten zu können, was wiederum eine Verbesserung wäre. Eine Verbesserung der Vorhersagegüte wird also auch als Entscheidungskriterium herangezogen.

Im Falle eines Regressionsproblems ist die Entscheidung, welche Ausgabe korrekt ist und welche nicht, unschärfer als bei der Klassifikation, weil es keine intuitive Möglichkeit gibt, sie im Allgemeinen auf ein binäres Ja oder Nein herunterzubrechen. Ein Regressor gibt eine kontinuierliche Zahl aus, die für den jeweiligen Anwendungsfall ausreichend genau genug ist oder nicht, wobei auch die Genauigkeitsmetrik von Fall zu Fall anders gewählt werden kann. So muss die Genauigkeit und die Metrik, in der sie gemessen wird, bei jeder Realisierung des Konzepts passend für den Anwendungsfall gewählt werden. Dies stellt keinen Mehraufwand dar, weil die Genauigkeitsmetrik, die für das bestehende neuronale Netz verwendet wird, wiederverwendet werden kann und die Genauigkeit, bei der eine Ausgabe als korrekt angesehen wird aus den Zielanforderungen an das bestehende neuronale Netz und dessen Vorhersagegüte bekannt sein sollte.

Entscheidung 2: Durchführung einer weiteren Iteration

Wenn Entscheidung 1 verneint wird, muss anschließend entschieden werden, ob eine weitere Iteration durchgeführt werden soll oder nicht (Entscheidung 2). Dahinter steht die Überlegung, ob es wahrscheinlich ist, dass ein weiterer Versuch oder weitere Versuche zum Erfolg führen werden oder nicht. Ideal wäre es, wenn genau vorhergesagt werden könnte, ob weitere Versuche zum Erfolg führen. Aufgrund der hohen Dimensionalität und Nichtlinearität der Loss-Funktion ist dies jedoch nicht praktikabel.

In der Praxis ist man daher auf die Anwendung von Heuristiken angewiesen. Die einfachste und praktikabelste Heuristik ist die Festlegung einer maximalen Anzahl von Iterationen: In einigen Testläufen werden verschiedene maximale Iterationszahlen ausprobiert und dann eine ausreichend große Zahl gewählt. Diese Zahl wird dann für den Rest der Datengenerierung festgelegt. Entscheidung 2 wird in diesem Fall durch eine einfache if-Abfrage auf die aktuelle Iteration i realisiert.

Komplexere Heuristiken, die weitere Informationen aus den bereits durchgeführten Iterationen zur Entscheidungsfindung nutzen, sind ebenfalls möglich. So kann der Grad der Verschlechterung der Ausgabe des bestehenden neuronalen Netzwerkes betrachtet werden. Wenn sich die Ausgabe verschlechtert hat, aber weiterhin nicht genug, damit Entscheidung 1 mit Ja beantwortet wurde, dann ist davon auszugehen, dass eine weitere Iteration ebenfalls zu einer Verschlechterung führt. Mit dieser Annahme kann eine adaptive Heuristik realisiert werden, die so lange weiter iteriert, wie noch eine Verschlechterung erzielt wird. Um die numerische Stabilität zu gewährleisten, sollte eine Mindestverschlechterung festgelegt werden.

Die Steuerkomponente des Datengenerators verwendet den Gradienten der Loss-Funktion, um die neuen Daten gezielt so zu generieren, dass es zu einer Verschlechterung kommt, wobei nicht ausgeschlossen ist, dass es auch zu einer Verbesserung kommen kann: Abbildung 3.4 zeigt ein klassisches Lehrbeispiel eines einfachen Gradientenabstiegs, der aufgrund einer zu großen Schrittweite divergiert. Auch dieser Fall kann Teil der Heuristik für Entscheidung 2 sein. Bei einer

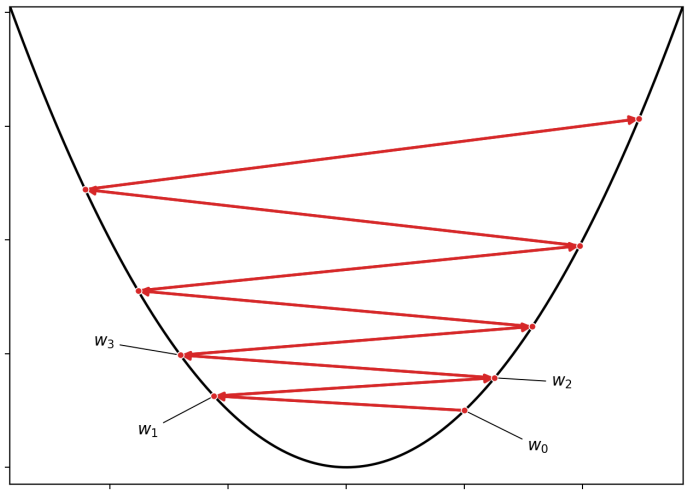

Abbildung 3.4: Beispiel eines divergierenden Gradientenabstiegs, der auf Grund der großen Schrittweite das Minimum nie finden kann.

Verbesserung kann von einer Divergenz ausgegangen werden und es sollte keine weitere Iteration durchgeführt werden. Ein Neustart mit einer kleineren Schrittweite ist der sinnvollere Weg.

Entscheidung 3: Neustart der gezielten Datengenerierung

Entscheidung 3 wird entweder getroffen, wenn ein neues Trainingsdatum erfolgreich generiert und dem neuen Trainingsdatensatz hinzugefügt wurde, oder wenn die Generierung erfolglos abgebrochen wurde. Diese Entscheidung wird in erster Linie dem Benutzer überlassen, der eine Zielanzahl neuer Daten generieren möchte. Wenn die Zielanzahl erreicht ist, wird nicht neu generiert, andernfalls wird neu generiert. Bei erfolglosem Abbruch der Iterationen kann entweder so oft weiter versucht werden, bis der Benutzer abbricht, oder es kann eine maximale Anzahl von Versuchen definiert werden, nach deren Erreichen abgebrochen wird.

3.3.3 Steuerungskomponente

Die Aufgabe dieser Komponente ist es, das generative neuronale Netzwerk durch den Vektor Δz so anzusteuern, dass es Daten generiert, die zur Verschlechterung der Ausgabe des bestehenden neuronalen Netzes führen. Als Maß für die Qualität der Ausgabe des bestehenden neuronalen Netzes wird die Loss-Funktion des bestehenden Netzes verwendet. Eine Loss-Funktion bewertet, wie gut ein Modell Vorhersagen im Vergleich zu den tatsächlichen Daten trifft. Sie quantifiziert den Unterschied zwischen den vorhergesagten und den wahren Werten.

Ziel des Trainings eines neuronalen Netzes ist es, diese Loss-Funktion zu minimieren, um durch Berechnung neuer Parameter θ die Vorhersagegenauigkeit zu verbessern. Die Steuerungskomponente hat genau das entgegengesetzte Ziel. Sie soll eine Erhöhung der Loss-Funktion bewirken. Im Gegensatz zum Training, bei dem der Parameter θ angepasst wird, soll sie $\boldsymbol{z}$ über $\Delta \boldsymbol{z}$ so verändern, dass sich die Ausgabe des bestehenden neuronalen Netzes verschlechtert. Wie beim Training eines Netzes geschieht dies ebenfalls, indem zunächst der Gradient der Loss-Funktion in Abhängigkeit von der zu optimierenden Größe, in diesem Fall $\boldsymbol{z}$, berechnet wird:

$$\nabla_z J(\theta, z, y) \tag{12}$$

Das Resultat ist ein Vektor, der misst, wie die Ausgabequalität des neuronalen Netzes auf kleine Veränderungen des latenten Vektors $\boldsymbol{z}$ reagiert. Im nächsten Schritt wird der Gradient dann durch Anwenden einer Funktion f in ein praktisch nutzbares $\Delta \boldsymbol{z}$ umgeformt:

$$\Delta z = f\big(\nabla_z J(\theta, z, y)\big) \tag{13}$$

Hier stellt sich die Frage, warum der Gradient nicht direkt als $\Delta \boldsymbol{z}$ genutzt werden kann. Der Grund liegt in der großen Komplexität der Loss-Funktionen. Es gibt keinerlei Garantien für deren

Beschaffenheit bei neuronalen Netzen. Sie enthalten sprunghafte Anstiege und sehr flache Ebenen [122]. Wenn der Gradient direkt benutzt wird, dann führt das zu numerischen Instabilitäten, die vermieden werden sollen.

Es gibt verschiedene denkbare f, die verwendet werden könnten. Zum Beispiel könnte eine Skalierung mit einem festen Skalar, wie beim Gradientenabstieg, eingesetzt werden. In Vorversuchen hat sich die Kombination aus der $sign$-Funktion und einem festen Skalar ϵ nach dem Vorbild des FGSM-Algorithmus als gut funktionierende Kombination herausgestellt [44].

$$\Delta z = \epsilon \, sign\big(\nabla_z J(\theta, z, y)\big) \tag{14}$$

Durch die Nutzung der $sign$ Funktion, wird die Information über die Größe des Gradienten nicht verwendet, nur die Richtungsinformation, was nicht optimal ist. Dafür ist die numerische Stabilität von Δz durch die feste Änderungsschrittweite ϵ sichergestellt.

Im letzten Schritt der Steuerungskomponente wird das aktualisierte $\boldsymbol{z}$ an das generative neuronale Netz weitergegeben.

3.4 Erörterung der Zielerreichung: Prüfung der Eignung des Konzepts zur Erfüllung der Teilziele

Das in diesem Kapitel vorgestellte Konzept eines automatisierten Trainingsdatengenerators für das maschinelle Lernen stellt einen ersten Lösungsansatz für das in Kapitel 1.2 vorgestellte Problem dar. Im folgenden Abschnitt wird das abstrakte Konzept mit den in Kapitel 1.3 aufgestellten Teilzielen für eine mögliche Lösung abgeglichen und verifiziert.

Teilziel 1: Automatisierte Trainingsdatengenerierung

Durch den Einsatz von generativen neuronalen Netzen werden die neuen Daten gänzlich automatisiert generiert. Es bedarf keiner weiteren manuellen Datensammlung, wenn die Voraussetzungen für das Konzept erfüllt sind, also ein gewisser Grundstock an gelabelten Trainingsdaten und das bereits trainierte neuronale Netz vorhanden sind. Die Menge an Trainingsdaten muss ausreichen, um damit ein generatives neuronales Netz trainieren zu können. Genau spezifiziert werden kann die Menge auf konzeptioneller Ebene nicht, weil sie zwischen verschiedenen Anwendungsdomänen und Implementierungen stark variiert. Auch die Label der neu generierten Daten werden automatisiert gemeinsam mit den Daten durch die Label-Einprägung generiert. Damit kann das Teilziel 1 als erfüllt betrachtet werden.

Teilziel 2: Generierte Daten müssen realistisch sein

Der Realismus von synthetischen Daten ist schwer zu messen und zu validieren. Bei Daten, die für Menschen intuitiv verständlich sind, wie Fotografien, kann zwar der Mensch als Beurteiler herangezogen werden, allerdings lassen sich Menschen leicht täuschen beziehungsweise haben ein begrenztes Wahrnehmungsspektrum, insbesondere bei geringen Fluktuationen, was das Beispiel des Pandas in Goodfellow et al. [44] eindrucksvoll gezeigt hat. Bei für Menschen nicht intuitiv verstehbaren Datenformen ist der Realismusgrad von synthetischen Daten noch schwieriger zu messen. Ein datentypenübergreifendes Maß für Realismus von Daten zu finden, liegt außerhalb des Umfangs dieser Arbeit. Daher wird die folgende Definition verwendet: Realistisch bedeutet hier, dass die generierten Daten die wesentlichen Eigenschaften und Muster der realen Daten so widerspiegeln, dass ein mit ihnen trainiertes Netz so generalisiert, dass es im Betrieb mit realen Daten zufriedenstellend funktioniert. Generative neuronale Netze sind prinzipiell in der Lage realistische Daten in diesem Sinn zu generieren, wie in Kapitel 2.3.1 ausführlich erläutert ist. Insofern kann das Teilziel 2 als erfüllt betrachtet werden.

Teilziel 3: Generierte Trainingsdaten müssen leistungssteigernd wirken

Die Steuerungskomponente nutzt die Ausgabe des bestehenden neuronalen Netzes, um das generative neuronale Netz so anzusteuern, dass die Ausgabe des bestehenden Netzes in der nächsten

Iteration verschlechtert wird. Dies geschieht, indem die Richtung des Gradienten der Loss-Funktion verwendet wird, um den latenten Vektor $\boldsymbol{z}$ gezielt anzupassen. Dadurch steigt die Wahrscheinlichkeit, dass die neu generierten Daten relevante Trainingsdaten sind im Vergleich zu einer rein zufälligen Wahl des $\boldsymbol{z}$-Vektors. Durch die gezielte Anpassung des $\boldsymbol{z}$-Vektors wird das Teilziel 3 erfüllt.

Teilziel 4: Übertragbarkeit für verschiedene Datenarten

Die Übertragbarkeit bezieht sich einerseits auf die Generierung der Daten und andererseits auf die Methode, mit der die Generierung gezielt gesteuert wird. Die Generierung wird durch generative neuronale Netze umgesetzt. Diese sind prinzipiell dazu in der Lage, verschiedene Arten von Daten zu generieren. Auch die Methodik der gezielten Ansteuerung der Generierung ist nicht speziell auf einzelne Datenarten wie beispielsweise Bilddaten begrenzt. Sie setzt in der latenten $\boldsymbol{z}$-Ebene an und ist damit wie die generativen Netze selbst unabhängig von dem speziellen Datentyp einsetzbar. Dadurch wird auch das Teilziel 4 durch das beschriebene Konzept erfüllt.

Durch die Validierung der Konzeption anhand der Teilziele wurde festgestellt, dass die Zielstellungen durch das Konzept grundsätzlich erfüllt werden können. Das Konzept eignet sich damit prinzipiell dazu, die Problemstellung zu adressieren und das Forschungsziel zu erfüllen. Im nächsten Kapitel wird das Konzept in mehreren Anwendungsfällen implementiert und evaluiert.

In diesem Kapitel wurde das zentrale Konzept der Arbeit vorgestellt, ein gezielter Trainingsdatengenerator. Das vorgeschlagene Konzept umfasst einen iterativen Datengenerator mit drei Modulen: Generatormodul, Steuerungsmodul und Ergebnisprüfungsmodul. Dieser Generator synthetisiert gezielt qualitativ hochwertige Trainingsdaten für neuronale Netze. Ein generatives neuronales Netz erzeugt diese Daten basierend auf latenten Zufallsvektoren. Die Ergebnisprüfung evaluiert, ob die neuen Daten das bestehende Netzwerk verbessern, und das Steuerungsmodul passt den latenten Vektor an, um die Loss-Funktion zu erhöhen. Die Methodik ist auf verschiedene Datentypen übertragbar und erfüllt die Anforderungen an realistische, gezielte und automatisierte Trainingsdatengenerierung. In dem folgenden Kapitel wird das Konzept anhand von drei Evaluierungsfällen realisiert und evaluiert.

4 Prototypische Realisierungen und Evaluierung des Ansatzes

In diesem Kapitel wird das in Kapitel 3 eingeführte Konzept zur gezielten und automatisierten Generierung von Trainingsdaten anhand von drei Evaluierungsszenarien prototypisch umgesetzt und bewertet. Dadurch soll die Tauglichkeit des Ansatzes zur Erfüllung der in Kapitel 1.3 definierten Ziele überprüft werden.

4.1 Vorgehen bei der Realisierung

Das in Kapitel 3 vorgestellte Konzept hat wenige direkte Einschränkungen hinsichtlich bestimmter Datentypen oder Anwendungsfälle. Lediglich die Formulierung des maschinellen Lernproblems ist auf das überwachte Lernen beschränkt, da die Ergebnisüberwachungskomponente zwingend ein Label benötigt, mit dem sie die Ausgabe des neuronalen Netzes vergleichen kann. Damit sind der Auswahl möglicher Evaluationsszenarien theoretisch kaum Grenzen gesetzt. Die Auswahl dieser drei spezifischen Anwendungsfälle zur Evaluierung des Konzepts wurde getroffen, um unterschiedliche Aspekte und Herausforderungen der synthetischen Datengenerierung im Bereich des maschinellen Lernens und der Automatisierungstechnik abzudecken. Jeder Anwendungsfall repräsentiert verschiedene Datenanforderungen und Problemstellungen, die typische Szenarien in industriellen Anwendungen widerspiegeln. Diese Auswahl erlaubt eine differenzierte Prüfung der Eignung des Konzepts zur Generierung relevanter und praxisnaher Daten sowie eine umfassende Bewertung seiner Übertragbarkeit und Flexibilität. Die spezifischen Anwendungsfälle und die jeweiligen Aspekte, die sie abdecken, werden in den folgenden Abschnitten

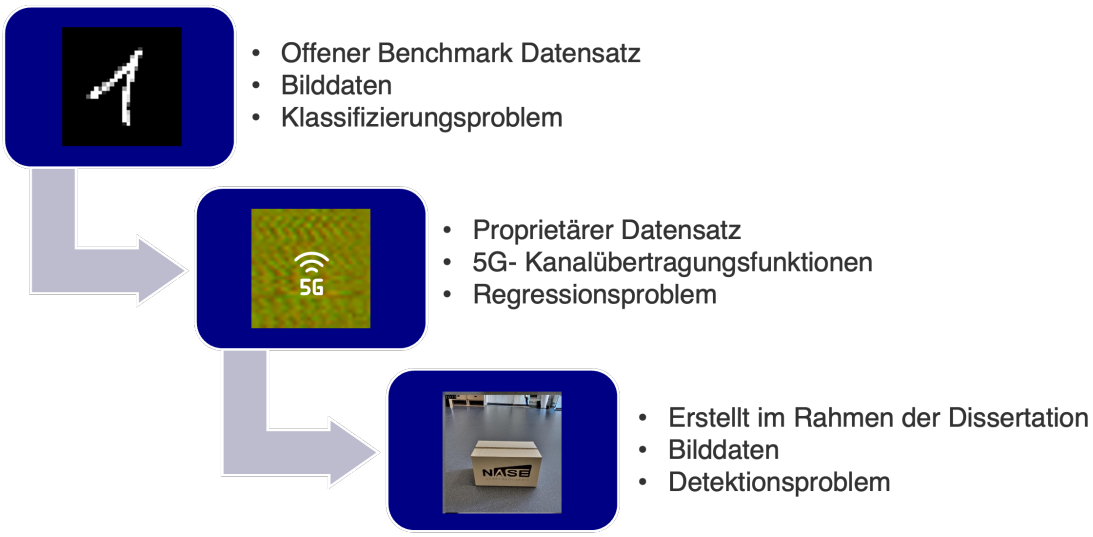

Abbildung 4.1: Übersicht der Evaluierungsszenarien in ansteigender Komplexität.

detailliert beschrieben, um eine fundierte Grundlage zur Analyse der Wirksamkeit und Grenzen der entwickelten Methodik zu schaffen. Abbildung 4.1 zeigt eine Übersicht der drei ausgewählten Anwendungsfälle, ihre Herkunft, die Art des Datensatzes und das damit verbundene Problem des maschinellen Lernens.

Der erste verwendete Datensatz ist der *Modified National Institute of Standards and Technology database* (MNIST)-Datensatz [123]. Angesichts seines Status als grundlegender Benchmark im Bereich des maschinellen Lernens bietet er eine kontrollierte Umgebung zur Bewertung der synthetischen Daten. Ziel des Anwendungsfalls ist es, festzustellen, ob das generative Modell die statistischen Eigenschaften des MNIST-Datensatzes, wie z.B. die Verteilung der Pixelintensitäten und die Varianz in den Zifferndarstellungen, replizieren und ausnutzen kann. Der erfolgreiche Einsatz mit diesem Datensatz kann die Fähigkeit des generativen Modells bestätigen, grundlegende Trainingsdaten zu erzeugen, die repräsentativ für reale Variationen sind.

Im nächsten Schritt wird der Ansatz zur Generierung von synthetischen Daten auf ein auf maschinellem Lernen basierendes 5G-Positionierungssystem angewendet. Positionierungssysteme, insbesondere solche, die auf trainierten Algorithmen basieren, benötigen umfangreiche Datensätze, die eine Vielzahl von Umgebungsvariablen umfassen. Synthetische Daten, die durch generative KI erzeugt werden, können seltene, aber kritische Szenarien abbilden, die in realen Datensätzen unterrepräsentiert sind, und dadurch die Robustheit des Ortungssystems gegenüber Ausreißern und anomalen Bedingungen erhöhen. Neben dem praktischen Nutzen zur Verbesserung des Positionierungssystems bietet dieser Anwendungsfall den Vorteil, ein Regressionsproblem zu sein und keine Bilddaten zu enthalten. Somit eignet sich der Anwendungsfall, um die Übertragbarkeit der Datengenerierungsmethode auf andere Problemformulierungen und Datenarten zu demonstrieren. Außerdem verdeutlicht er die Anwendbarkeit des Konzepts auf ein Problem der Automatisierungstechnik.

Mit dem dritten Evaluierungsszenario wird die Übertragbarkeit auf eine komplexere Problemformulierung anhand eines Objektdetektors untersucht. Bildbasierte Objektdetektoren benötigen umfangreiche Datensätze mit einem hohen Maß an Variabilität, um Overfitting zu vermeiden und die Generalisierung zu verbessern. In diesem Fall lag der Schwerpunkt auf der Fähigkeit des generativen Modells, gelabelte Bilder unter verschiedenen Bedingungen zu erzeugen. Bei verbreiteten Objektdetektionsproblemen existieren große öffentliche Datensätze, die als Trainingsgrundlage genutzt werden können. Diese enthalten, aber nur Objekte, die in den jeweiligen öffentlichen Umgebungen enthalten sind, etwa PKW, Personen oder andere Verkehrsteilnehmer. Wenn Objekte aus industriellen Umgebungen, wie sie in der Automatisierungstechnik verbreitet sind, detektiert werden sollen, existieren typischerweise keine öffentlichen Datensätze dafür, weil die industriellen Objekte einzigartig für das jeweilige Unternehmen oder den Standort sind. Aufgrund dieses Mangels an Trainingsdaten ist die Option, durch die Verwendung synthetischer Daten auf die manuelle Erhebung vieler realer Daten verzichten zu können, sehr attraktiv.

4.2 Evaluierungsfall 1: Handschrifterkennung

In diesem Kapitel wird das Konzept auf das Anwendungsbeispiel der Handschrifterkennung angewendet. Dabei handelt es sich um ein Problem der Bildklassifizierung. Im Folgenden ist zunächst der Anwendungsfall und der Datensatz beschrieben, dann ist das Konzept realisiert und abschließend evaluiert.

4.2.1 Anwendungsfall

Der MNIST-Datensatz wurde entwickelt, um als Benchmark für maschinelles Lernen und Bilderkennung zu dienen. Er besteht aus einer großen Sammlung von handgeschriebenen Ziffern, die in Trainings- und Testsets unterteilt sind. Ursprünglich erstellt in den 1990er Jahren, wurde er aus älteren Datensätzen des NIST zusammengestellt, um Forschern eine standardisierte Basis für die Entwicklung und Bewertung von Algorithmen im Bereich des maschinellen Lernens zu bieten, insbesondere für die Aufgaben der Bilderkennung und Klassifizierung. Der MNIST-Datensatz löste das Problem, dass es an einem leicht zugänglichen, umfangreichen und einheitlich formatierten Datensatz für Einsteiger und Experten in der Gemeinschaft der maschinellen Lernforschung mangelte [123].

Heute gilt MNIST als einer der klassischen Datensätze im Bereich des maschinellen Lernens und wird oft in Bildungs- und Forschungsumgebungen verwendet, insbesondere für diejenigen, die neu in diesem Bereich sind. Er dient als Ausgangspunkt für das Experimentieren mit Techniken der Bilderkennung, bevor man zu komplexeren Aufgaben übergeht. Trotz seiner Einfachheit bleibt MNIST relevant, da er als Maßstab für die Beurteilung der Grundfähigkeiten von Algorithmen im Bereich der Bilderkennung dient und oft als erste Herausforderung für neue Ansätze in diesem Bereich betrachtet wird. Er besteht aus 70.000 Bildern von handgeschriebenen Zahlen (0 bis 9). Die Bilder sind im Graustufenformat abgespeichert und haben eine Auflösung von 28x28 Pixel.

4.2.2 Realisierung

In diesem Kapitel geht es um die Realisierung des in Kapitel 3 vorgestellten Konzepts für den Evaluierungsfall 1. In Abbildung 4.2 ist die gesamte Realisierung schematisch dargestellt. Das Kapitel beschreibt die Realisierungsaspekte, nach den Komponenten des Konzepts: Generatives neuronales Netz, Ergebnisüberprüfungskomponente und Steuerungskomponente.

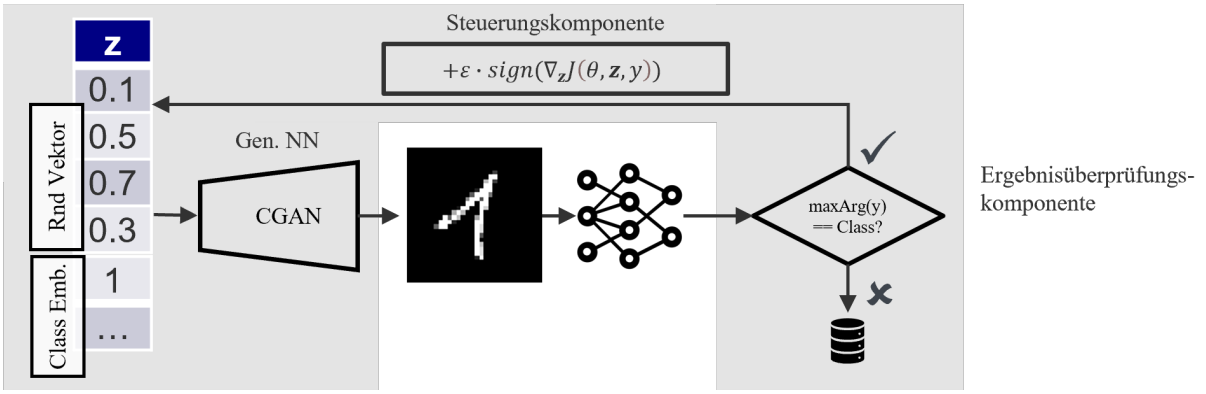

Abbildung 4.2: Übersicht über die Realisierung des Anwendungsfalls 1

4.2.2.1 Generatives neuronales Netz

Als generatives neuronales Netz wurde das CGAN nach [102] ausgewählt. Abbildung 4.3 zeigt die verwendete CGAN-Architektur. Der Eingangsvektor des Generators hat eine Länge von 110 und besteht aus dem latenten Vektor $\vec{z} \epsilon \mathbb{R}^{100}$ und dem Ausgang der Embeddings-Schicht $y \epsilon \mathbb{R}^{10}$

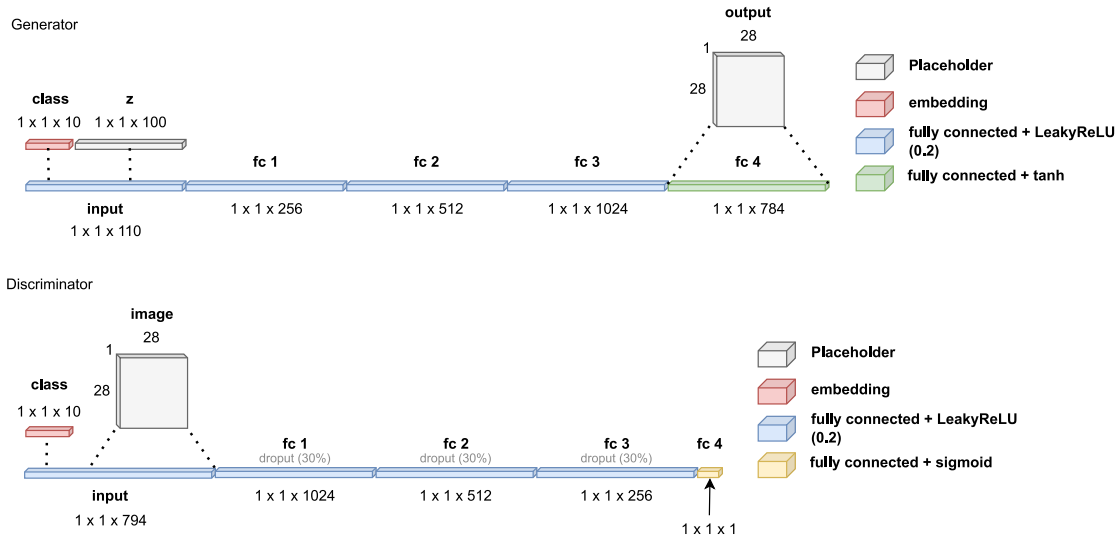

Abbildung 4.3: Die realisierte CGAN-Architektur bestehend aus Generator und Discriminator, die durch Convolution-Operationen umgesetzt werden.

($class$). Über das $\mathbb{R}^{10}$ Embedding ist das Klassenlabel One-Hot-enkodiert Teil des Modells. One-Hot-Encoding ist eine Methode, bei der kategoriale Variablen in ein binäres Format umgewandelt werden. Jede Kategorie wird durch einen Vektor repräsentiert, in dem genau ein Element den Wert 1 hat (daher "One-Hot"), während alle anderen Elemente den Wert 0 haben. Das CGAN unterscheidet sich zum gewöhnlichen GAN dadurch, dass der Aufruf des Generators von $G(\vec{z})$ zu $G(\vec{z}, \vec{y})$ wird. Das CGAN ist mit den Hyperparametern in Tabelle 4.1 trainiert.

Tabelle 4.1: Überblick über die Hyperparameter in Evaluierungsfall 1

Hyperparameter	Wert
Anzahl der Epochen des Trainings	50
Batch-Größe	32
Lernrate des Optimierers	0.0001
Dimension des latenten Vektors	100
Anzahl Trainingsiterationen des Diskriminators	5

Wie Abbildung 4.4 zeigt, konvergiert der Loss des CGANs, was darauf hindeutet, dass die Optimierung erfolgreich ist und kein „Mode Collapse“ eingetroffen ist. Die Loss-Funktion eines GANs mit Mode Collapse sieht so aus, dass einer der beiden Gegenspieler, meistens der Diskriminator, einen signifikant niedrigeren Loss hat als der jeweils andere.

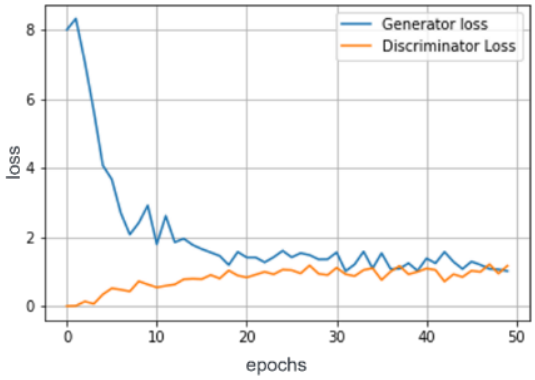

Abbildung 4.4: Verlauf des Trainingslosses von Generator und Discriminator.

4.2.2.2 Ergebnisüberprüfungskomponente

Die Ergebnisprüfungskomponente bewertet die Ausgabe des neuronalen Netzes, das auf dem MNIST-Datensatz trainiert wurde, im Hinblick auf die korrekten Labels, die während der

Generierung der Eingaben für das Netzwerk eingebettet wurden. Diese Komponente nimmt auch die Anzahl der bisher durchgeführten Iterationen des spezialisierten Datengenerators in Betracht. Auf der Grundlage dieser Daten entscheidet sie, ob das vom Generator erstellte Bild der Ziffer dem neuen Datensatz "Gezielt generierte MNIST-Trainingsdaten" hinzugefügt wird. Falls ein Bild nicht hinzugefügt wird, bestimmt die Komponente, ob der Generierungsprozess mit einer weiteren Iteration fortgesetzt oder abgebrochen wird.

Bei Klassifizierungsproblemen, wie es der MNIST-Datensatz eines ist, wird als Ergebnisklasse der höchste Wert des Ausgabevektors des Klassifikators genommen. Der Ausgabevektor wird hier als eine Pseudowahrscheinlichkeit interpretiert, die ein Maß dafür ist, wie sicher sich das neuronale Netz seiner Entscheidung ist. Die Ergebnisüberprüfungskomponente vergleicht das bekannte korrekte Klassenlabel mit der Klasse, für die das neuronale Netz die höchste Klassifizierungswahrscheinlichkeit ausgibt. Wenn das vom Datengenerator generierte Bild nicht korrekt erkannt wird, wird das Bild dem synthetischen Trainingsdatensatz hinzugefügt. Wenn das generierte Bild korrekt erkannt wird, dann wird das Bild nicht hinzugefügt, stattdessen wird eine weitere Iteration durchgeführt, bis die maximale Anzahl an Iterationen überschritten ist oder der Vorgang divergiert. Die Ergebnisüberprüfungskomponente vergleicht die Klassifizierungswahrscheinlichkeiten zwischen einzelnen Iterationen und wenn diese nicht abnimmt, sondern zunimmt, dann wird davon ausgegangen, dass der Vorgang divergiert und er wird abgebrochen.

4.2.2.3 Steuerungskomponente

Wie im Konzeptkapitel über die Steuerungskomponente 3.3.3 beschrieben, implementiert die Steuerungskomponente den FGSM-Algorithmus. Für die Berechnung des Gradienten und der $sign$-Funktion werden die eingebauten Funktionen im verwendeten PyTorch Framework [27] genutzt. Die Kostenfunktion $J(\theta, z, y)$ kann aus dem Klassifizierungsnetz übernommen werden. Es wurde der bei Klassifizierungsproblemen verbreitete Negative Log Likelihood mit der Implementierung des PyTorch Frameworks verwendet. Negative Log Likelihood Loss ist für die Klassifizierung effektiv, da es die Modellausgaben als Wahrscheinlichkeiten interpretiert und ein direktes Maß dafür liefert, wie gut das Modell die wahre Bezeichnung vorhersagt. Das Modell wird stark bestraft, wenn es falschen Vorhersagen eine hohe Wahrscheinlichkeit zuweist, was zu genaueren Vorhersagen führt. Ferner gewährleistet die Kombination mit der softmax-Funktion Recheneffizienz und numerische Stabilität.

Die Wahl von ϵ ist von großer Bedeutung für die Steuerungskomponente. Wenn ϵ zu groß gewählt ist, dann wird die Generierung fehlschlagen, weil die Optimierung divergiert, wie in Abbildung 3.4 visualisiert ist. Bei einem zu kleinen ϵ ist auch die Änderungsrate der Klassifizierung des neuronalen Netzes sehr gering und es müssen viele Iterationen durchgeführt werden, damit ein neues herausforderndes Bild erfolgreich generiert wird. Grundsätzlich gilt, je größer ϵ, desto

größer ist die Änderung des latenten Vektors des generativen Netzes. Mit großen Änderungen des latenten Eingangsvektors gehen typischerweise auch große Änderungen der generierten Daten einher. Daraus ergibt sich, dass ϵ so groß wie möglich sein sollte, ohne dabei die Stabilität der Datengenerierung zu gefährden. Es liegt nahe, die Literaturwerte für FGSM von ϵ zu übernehmen, allerdings arbeitet FGSM im Pixelraum und die Steuerungskomponente im Latentraum des synthetischen Datengenerators. Zur Größe von ϵ und dem Vergleich mit FGSM wurde für diesen Anwendungsfall ein Experiment durchgeführt.

Abbildung 4.5 zeigt, wie viele Täuschungen der jeweilige Algorithmus in Abhängigkeit von ε erfolgreich erzeugt hat.

Das Experiment ist 100-mal pro ε wiederholt, um einen Mittelwert der Anzahl darzustellen. Da dieses Experiment mit dem Generator des CGAN durchgeführt wurde, sind 100 Versuche jeweils 10 Versuche pro Klasse. Der transparente Bereich zeigt die maximale und minimale Anzahl von erfolgreich generierten Täuschungen X‘ an. Die durchgezogene Linie gibt den Modus der Anzahl an und zeigt, welche Anzahl von generierten Täuschungen am häufigsten auftritt. Der Vergleich zeigt, dass die Latentvektoranpassung mit G erst ab ε=0,75 100 Täuschungen erfolgreich generiert, während die FGSM dies bei 0,3 erreicht. Der Unterschied in ε-Richtung lässt sich mit der zusätzlichen Restriktion des CGAN, als auch der Täuschung des FGSM durch Pixelfluktuationen erklären. Diese verspätete Konvergenz zu 100 zeigt, dass G in der Lage ist, C 100-mal zu täuschen, dies allerdings eine höhere Abänderung von $\vec{z}$ durch ε erfordert als bei FGSM. Das erste ε, das eine Anzahl größer als 0 erzeugt hat, ist 0,1 und stellt somit den Minimalwert dar. Damit die Latentvektoranpassung mit dem höchsten ε eine hohe Wahrscheinlichkeit hat, alle möglichen Da-

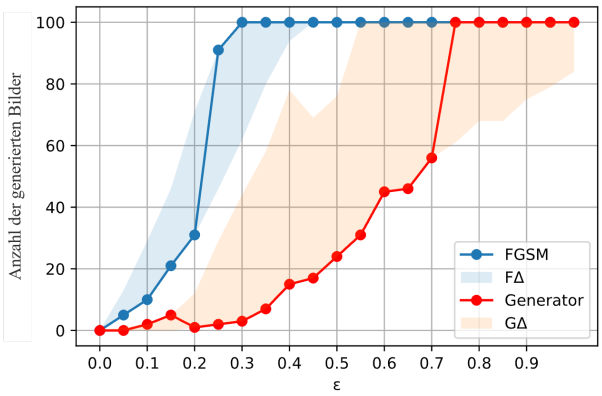

Abbildung 4.5: Experiment zum Vergleich der Abhängigkeit von ϵ bei FGSM und der Latentvektoranpassung

ten zu generieren, wird der Maximalwert von ϵ nicht als 0,75, sondern 0,8 gewählt. Der Wertebereich für ε für die Latentvektoranpassung ermittelt sich somit mit 0,1 als Mindestwert bis zu 0,8. Durch die Obergrenze bei 0,8 ist die Latentvektoranpassung begrenzt. Auf Basis dieser Ergebnisse

hat sich ein adaptiver Ansatz für die Wahl von ϵ als zielführend herausgestellt. Es wird zunächst mit $\epsilon = 0{,}1$ begonnen, wenn die Iteration nicht erfolgreich beendet wird, dann wird ϵ Schritt für Schritt erhöht.

4.2.3 Evaluierung

In diesem Abschnitt ist das im vorherigen Kapitel realisierte Konzept mithilfe von drei Experimenten evaluiert. Im ersten Experiment wird untersucht, ob der gezielte Datengenerator in der Lage ist, Bilddaten zu generieren, die in einem hochperformanten Klassifikator zu einer Klassifizierungsänderung führen. Das zweite Experiment untersucht, welche Arten von realistischen Änderungen an den Bilddaten durch den Generator erschaffen werden und wie sich diese zu den Bildausgaben der FGSM-Methode unterscheiden. Im dritten Experiment wird der Datengenerator dann eingesetzt, um Trainingsdaten zu generieren und die mit deren Einsatz verbundenen Verbesserungen sind untersucht.

4.2.3.1 Experiment 1: Latentvektoranpassung

Im Experiment 1 wird überprüft, ob der gezielte Datengenerator mit seinem Prinzip der Latentvektoranpassung in der Lage ist, Bilddaten zu generieren, die zu einer Klassifizierungsänderung führen. Dazu wurde der Generator mit $\varepsilon = 0{,}1$ und der MNIST Klasse 1 200-mal betrieben.

Abbildung 4.6 zeigt links die Originalbilder $\boldsymbol{X} = G(\vec{z}, \vec{y})$, in der Mitte die abgeänderten Daten $\boldsymbol{X'}$ und ganz rechts, wie sich die Klassifizierung geändert hat ($\vec{y} \rightarrow \vec{y}'$). Von den 200 generierten Bildern wurden die ersten 25 für die Abbildung genutzt. Es hat keine manuelle Auslese stattgefunden, um nur qualitativ hochwertige Ergebnisse zu zeigen. Die Abänderung durch die Latentvektoranpassung $L_p(\boldsymbol{X'}, \boldsymbol{X})$ ist gering, da ε den Wert 0,1 hat und somit den Minimalwert für eine erfolgreiche Abänderung hat.

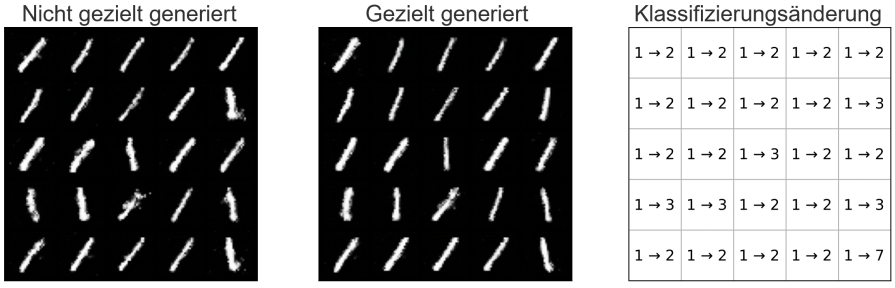

Abbildung 4.6: MNIST-Bilder, die durch den Datengenerator generiert wurden. Die gezielt generierten Daten führen zu einer Klassifizierungsänderung und steigern die Modellleistung.

Durch dieses Experiment ergeben sich drei Schlussfolgerungen:

- Der Datengenerator ist in der Lage Daten zu generieren, die zunächst korrekt erkannt werden und durch die Steuerungskomponente kann er ebenfalls gezielt Daten generieren, die nicht korrekt klassifiziert werden.
- Die generierten Bilder sehen realistisch aus. Eine detaillierte Analyse des Realitätsgrades der generierten Daten ist im nächsten Experiment durchgeführt.
- Die Änderungen, die bei den gezielt generierten Bildern zu erkennen sind, sind alle legitim in dem Sinn, dass das Bild weiterhin zweifelsfrei von einem Menschen der Klasse 1 zugeordnet werden kann. Ein Beispiel für eine illegitime Änderung wäre das Ergänzen eines Querstriches, um den Klassifikator dazu zu bringen eine 7 zu erkennen. Die Änderungen sind fast alle entweder leichte Rotationen oder Änderungen der verwendeten Strichdicke.

4.2.3.2 Experiment 2: Vergleich zwischen Latentvektoranpassung und FGSM

Die FGSM-Methodik zur Generierung von synthetischen Bilddaten hat, wie in Kapitel 2.1.3 ausführlich beschrieben, den Nachteil, dass die synthetischen Bilddaten unrealistisch sind. Ein Training mit den Daten führt zwar zu einer Resilienz gegen die Methode, nicht aber zu einer Verbesserung eines neuronalen Netzes für den Einsatz mit echten Daten. In diesem Experiment wird die Realitätsnähe der vom Datengenerator erzeugten synthetischen Daten durch den Vergleich von mit FGSM generierten adversarialen Beispielen untersucht.

Adversariale Beispiele haben zwei Nachteile. Der erste Nachteil ist, dass die Daten zwar falsch klassifiziert sind, allerdings beruht diese Falschklassifizierung auf minimalen Abänderungen der Daten (bei Bilddaten sogenannte Pixelfluktuationen) (siehe Abbildung 4.7). Der zweite Nachteil ist, dass, wenn ein Klassifikator mit den adversarialen Daten trainiert wird, sich die Genauigkeit des Klassifikators senkt, anstelle sie konstant zu halten oder sie zu verbessern [44]. Um zu vergleichen, ob die Daten durch die Latentvektoranpassung ebenfalls diese Eigenschaften aufweisen, werden diese mit den Daten der FGSM verglichen.

Dieses Experiment vergleicht die Änderung des Datenpunkts durch die Algorithmen ($\boldsymbol{X} \rightarrow \boldsymbol{X}'$), wobei die Änderung als

$$\Delta \boldsymbol{X} = sign(\boldsymbol{X}' - \boldsymbol{X}) \tag{15}$$

definiert ist. Dabei ist der Grad der Änderung irrelevant; entscheidend ist lediglich, an welcher Position der Algorithmus eine Änderung von dem Ursprungsdatum $\boldsymbol{X}$ zu dem generierten Datum $\boldsymbol{X}'$ vornimmt. Das ε in diesem Experiment steht für den Verschmutzungsprozess durch FGSM und die Latentvektoranpassung $\varepsilon = 1$. Damit der Klassifikator erfolgreich getäuscht wird, ist der Wert auf den Maximalwert gesetzt, da der Grad der Änderung irrelevant ist. Beide Algorithmen generieren 100 Täuschungen $\boldsymbol{X}'$. Anschließend wird $\boldsymbol{\Delta X}$ berechnet und auf den Wertebereich $[-1, 1]$ normiert.

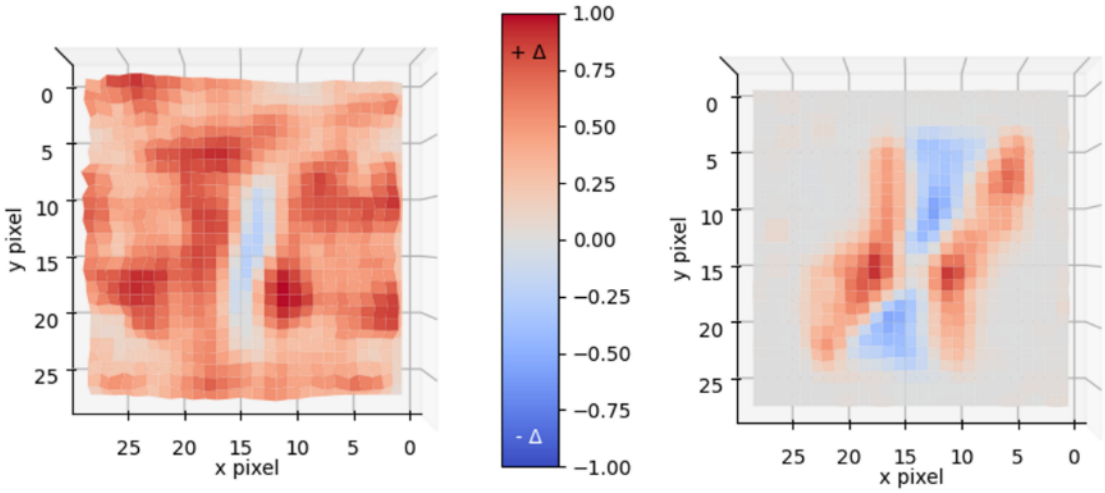

Abbildung 4.7: Position der Pixeländerung erzeugt durch den FGSM-Ansatz (links) und die Latentvektoranpassung (rechts).

Abbildung 4.7 zeigt den Unterschied für die gezielten Änderungen durch FGSM und die Latentvektoranpassung. Links ist die Pixeländerung durch die FGSM und rechts ist die Latentvektoranpassung dargestellt. Je roter ein Bereich ist, desto häufiger sind Pixel durch den Algorithmus hinzugefügt und je blauer, desto öfter ist ein Pixel bei der Änderung entfernt worden. Abbildung 4.8 zeigt eine dreidimensionale Ansicht der Graphen.

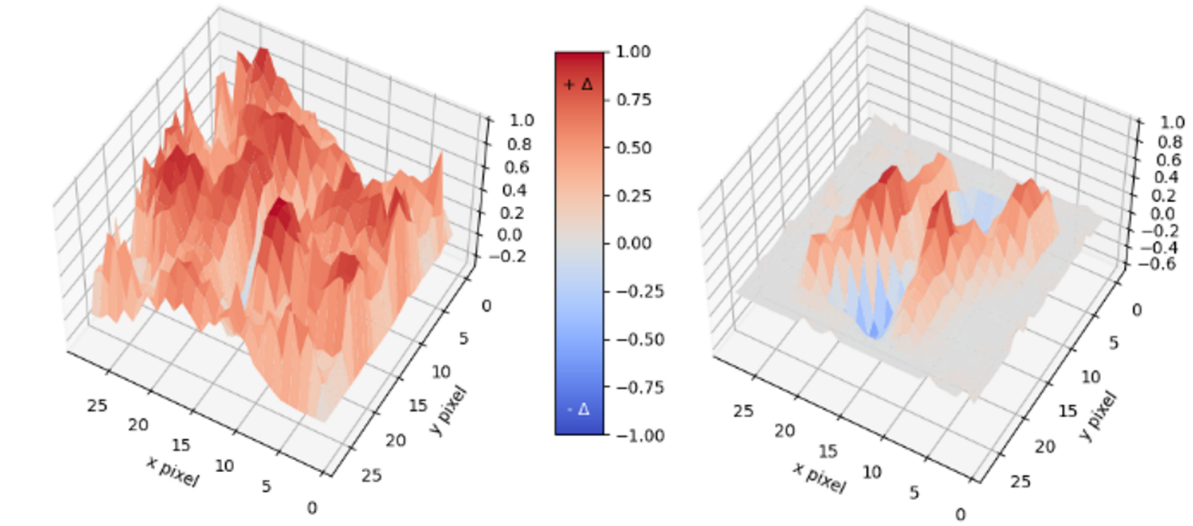

Abbildung 4.8: 3D-Ansicht auf die Pixeländerungen FGSM (links) und die Latentvektoranpassung (rechts).

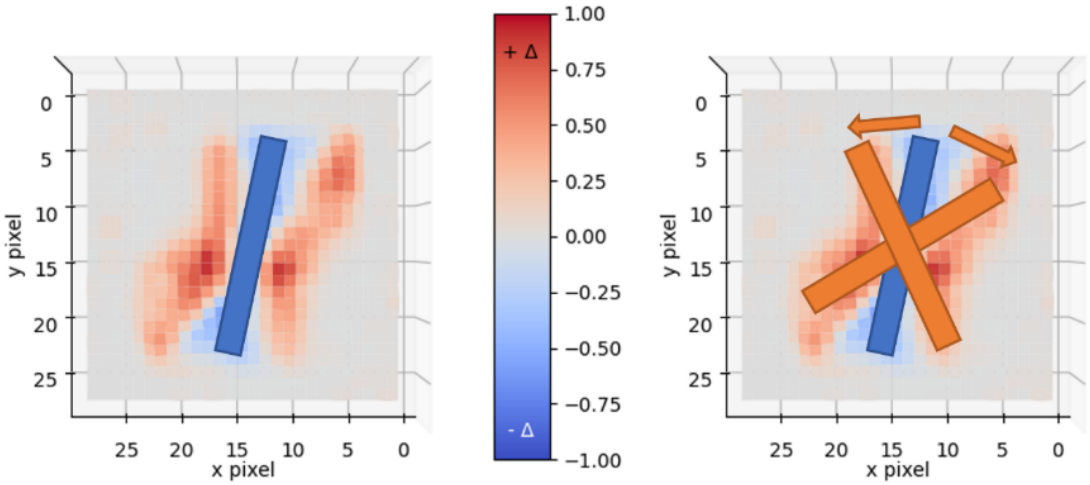

Abbildung 4.9: Transformation der 1 durch die Latentvektoranpassung

Bei der Pixeländerung durch die FGSM (links) zeigt sich, dass das komplette Bild durch den Algorithmus abgeändert ist. Bis auf das Zentrum des Bildes (Pixelbereich [16-14, 22-7]) sind zu dem Bild Pixel hinzugefügt worden. Das Zentrum ist konstant bzw. hat abgenommen, da das Bild Teil des MNIST-Datensatzes mit Wertebereich [0,1] ist. So konnte FGSM den Wert 1 nicht weiter erhöhen, sondern lediglich entfernen. Es zeigt sich, dass die Falschklassifizierung durch Pixelfluktuationen entsteht und nicht durch reale Datenänderung wie z.B. durch eine Rotation der Zahl. Die Pixeländerung durch die Latentvektoranpassung nimmt keine vollständige Bildanpassung vor. Stattdessen zeigt diese Änderung eine Transformation, die in Abbildung 4.9 visualisiert ist.

Die blaue „1“ im linken Bild wird entfernt, weswegen sie in dem Bild blau ist. Die „1“ wird allerdings nicht vollständig entfernt, sondern wird rotiert hinzugefügt (rechts). Die orangen Striche in dem rechten Bild sind Beispiele für die rotierte blaue „1“. Somit ist die Transformation eine Rotation. Anhand dieser „1“ zeigt sich, dass die Latentvektoranpassung, anders als die reine Pixelfluktuation durch FGSM, die Daten gezielt abändert und sogar selbstständig eine Transformation nachempfindet, um den Klassifikator zu täuschen. Diese Transformation ist nicht vorgegeben, sondern selbstständig erlernt.

Diese Transformation ist auch bei anderen Klassen zu sehen, wie Abbildung 4.10 zeigt. Sowohl die Klasse „6“ als auch die Klasse „9“ sind im Uhrzeigersinn rotiert, wobei auch eine leichte Rotation gegen den Uhrzeigersinn erkennbar ist. Es sind keine anderen Transformationen außer der Rotation aus der Abbildung erkennbar. Dies liegt daran, dass G die Transformation aus dem Originaldatensatz ableitet und somit Variationen der Zahlen erlernt, anstatt neue Transformationen, wie z.B. einer Spiegeltransformation, bei der die Zahlen anders aussehen als jede Zahl in dem Datensatz.

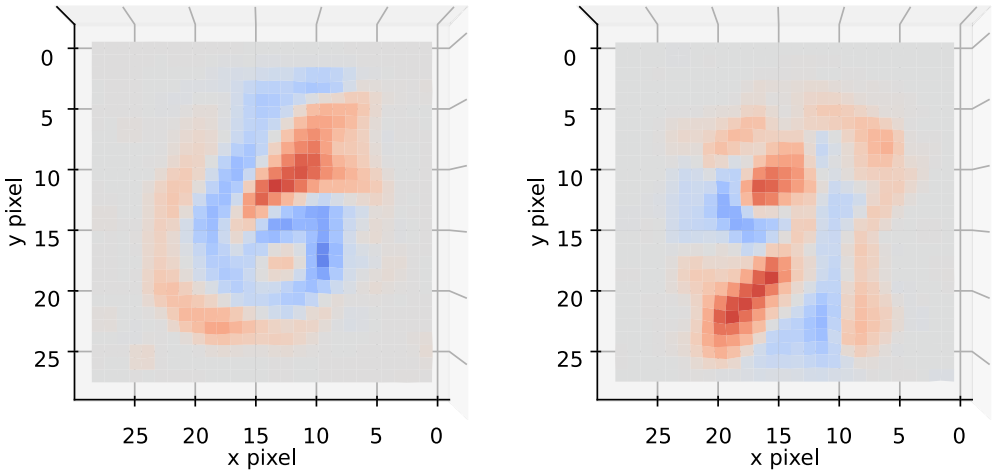

Abbildung 4.10: Transformation der Klassen "6" und "9" durch die Latentvektoranpassung

Ausgehend von dem Ergebnis dieses Experiments ergibt sich, dass die Latentvektoranpassung die Datenvarianz nicht verringert, da die Latentvektoranpassung neue Daten erzeugt, die in dem Datensatz nicht vorhanden sind. Durch die neuen Daten nimmt die Varianz zu, obwohl die Transformation aus dem Originaldatensatz abgeleitet ist. Um diese Annahme zu untersuchen, wurde ein Vergleich zwischen den Trainingsdaten und den generierten Daten durchgeführt. Dieser Vergleich zeigt, inwieweit sich die generierten Daten vom Originaldatensatz unterscheiden. Als Referenz ist in der Abbildung derselbe Vergleich zwischen den durch FGSM erzeugten Daten und dem Originaldatensatz dargestellt. Da die FGSM-Daten die Genauigkeit verringern, zeigt dieser direkte Vergleich, wie sich die Latentvektoranpassung im Vergleich dazu unterscheidet. Im folgenden Abschnitt wird der genaue Unterschied zwischen den durch FGSM erzeugten Daten und den durch Latentvektoranpassung generierten Daten dargestellt.

In der Abbildung 4.11 ist die Schnittmenge der Klasse „1“ von dem Mittelwert der Trainingsdaten und den generierten Daten dargestellt. Links ist die Schnittmenge mit den generierten Daten der FGSM und rechts die Schnittmenge mit den Daten der Latentvektoranpassung dargestellt. Während bei FGSM, bis auf einige Pixelfluktuationen, die Trainingsdaten mit den generierten Daten identisch sind, sind die Daten der Latentvektoranpassung zu den Trainingsdaten unterschiedlich. So ist die „1“ ca. zwei Pixel in y-Richtung und vier Pixel in x-Richtung größer. Zusätzlich ist die

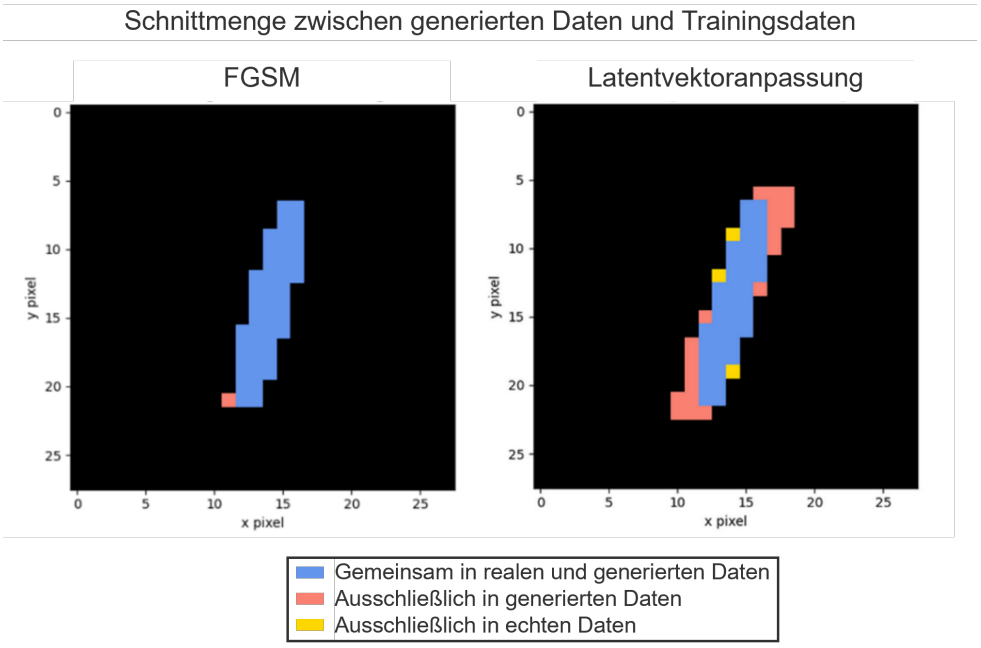

Abbildung 4.11: Unterschied zwischen Trainingsdaten und generierten Daten von FGSM (links) und Latentvektoranpassung (rechts)

„1“ durch die FGSM schmaler und zeigt 3 Pixel, die in dem Originaldatensatz auftreten und nicht in der Latentvektoranpassung auftreten.

Während der FGSM die Trainingsdaten bis auf einen Pixel repliziert, erzeugt die Latentvektoranpassung Daten, die in dem Trainingsdatensatz nicht vorhanden sind. Dadurch erhöht sich die Datenvarianz, und es werden neue Fälle abgedeckt, die im Trainingsdatensatz nicht enthalten sind. Darüber hinaus ermöglicht dies einen Einblick darin, ob der Datensatz ausgeglichen ist oder ob Eigenschaften der Klassen bei der Datensammlung vernachlässigt wurden. Die Abbildungen zeigen, dass insbesondere eine größere, rotierte „1“ beziehungsweise rotierte Ziffern ein Problem für den trainierten Klassifikator darstellen.

Durch diese Experimente ergibt sich, dass

- die generierten synthetischen Trainingsdaten zu einer Vergrößerung der Datenvarianz führen. Die synthetischen Daten enthalten neue Muster, die so nicht in den realen Daten vorkommen.
- die generierten Daten realistische Transformationen der relevanten Bereiche innerhalb der Bilder zeigen.
- der gezielte Datengenerator selbstständig die realistischen Transformationen erlernt. Er hat keine Informationen dazu erhalten, welche Transformationen legitim sind und welche nicht.

4.2.3.3 Experiment 3: Training mit gezielt leistungssteigernd generierten Daten

Im letzten Experiment dieses Anwendungsfalls wird eine große Menge synthetischer Daten gezielt generiert und mit ihnen trainiert. Um die Grenzen des Verfahrens zur gezielten Generierung von synthetischen Daten auszuloten, wird ein Vorgehen angewendet, das im Folgenden als Konvergenztraining bezeichnet wird. Dieses Konvergenztraining hat das Ziel, den Trainingsdatensatz (60.000 Daten) auf eine Größe von 120.000 zu erweitern und somit den Datensatz zu verdoppeln. Bei diesem Experiment wird untersucht, ob über den Verlauf des Trainings die Anzahl an erzeugten Täuschungen abnimmt und dadurch das Täuschen des Klassifikators C mittels des Generators G nicht mehr möglich ist. Eine Iteration des Konvergenztrainings beinhaltet:

1. Laden des Datensatzes D
2. Training des Klassifikators C mit D
3. Täuschen des Klassifikators durch die Latentvektoranpassung

4. Erweitern des Datensatzes D mit den erfolgreich generierten Täuschungen X‘.

Eine Iteration generiert dabei 70 Täuschungen pro Klasse und somit 700 neue Daten pro Iteration.

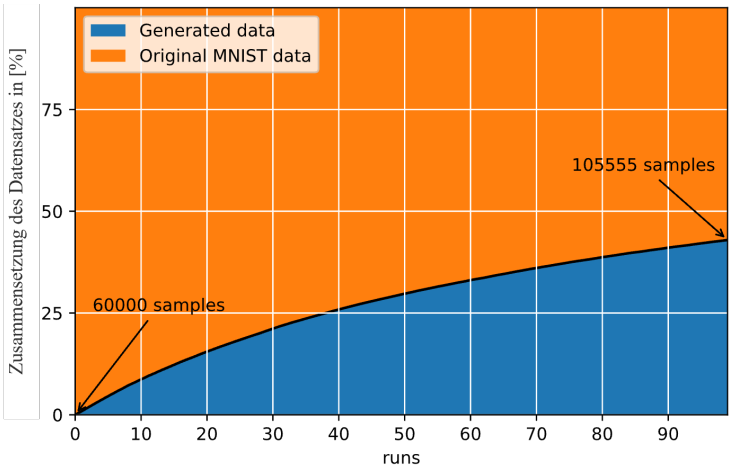

Abbildung 4.12: Verlauf der Zusammensetzung des Trainingsdatensatzes über das Konvergenztraining hinweg.

100 Iterationen generieren somit 70.000 Datenpunkte und der Trainingsdatensatz ist verdoppelt. Abbildung 4.12 zeigt über die 100 Iterationen des Konvergenztrainings die Zusammensetzung des Trainingsdatensatzes aus dem Originaldatensatz und den Täuschungen. Der Trainingsdatensatz besteht aus 60.000 Daten.

Der Abbildung 4.12 ist zu entnehmen, dass bei der ersten Iteration der Datensatz aus dem originalen MNIST-Trainingsdatensatz mit 60.000 Daten besteht. Nach 100 Iterationen (runs) ist der Datensatz bei 105.555 Daten und auf ca. 175 % der Anfangsgröße vergrößert worden. Nach der letzten Iteration fehlen 14.445 Daten bis zu den 120.000 Daten. Das bedeutet, dass nicht in jeder Iteration 700 Täuschungen erfolgreich erzeugt wurden. ε wird mit den Iterationen inkrementell erhöht, bis es den Maximalwert von 0,8 annimmt. Wenn mit ε=0,8 keine 70 Daten erzeugt werden konnten, wird ε nicht weiter erhöht, um die Möglichkeit illegitimer Täuschungen zu verringern. Das Konvergenztraining zeigt, dass das Netz durch das Training mit den generierten Daten die Transformationen, die durch die Latentvektoranpassung erzeugt werden, lernt. Ob sich das auch positiv auf die Performanz des Netzes im realen Anwendungsfall auswirkt, lässt sich daraus noch nicht schließen. Das wird im nächsten Kapitel diskutiert.

4.2.3.4 Experiment 3: Ergebnisse

In diesem Kapitel wird die Performanz des Netzes evaluiert, das durch das Konvergenztraining entstanden ist. Zunächst wird betrachtet, wie sich die Genauigkeit auf den originalen MNIST-Daten ändert. Abbildung 4.13 zeigt während des Konvergenztrainings eine Verminderung der Genauigkeit um etwa 1 % auf Trainings- und Testdatensatz. Im Gegensatz dazu hat das Training mit den durch FGSM generierten Daten zu einem Einbruch um 10 % geführt [44].

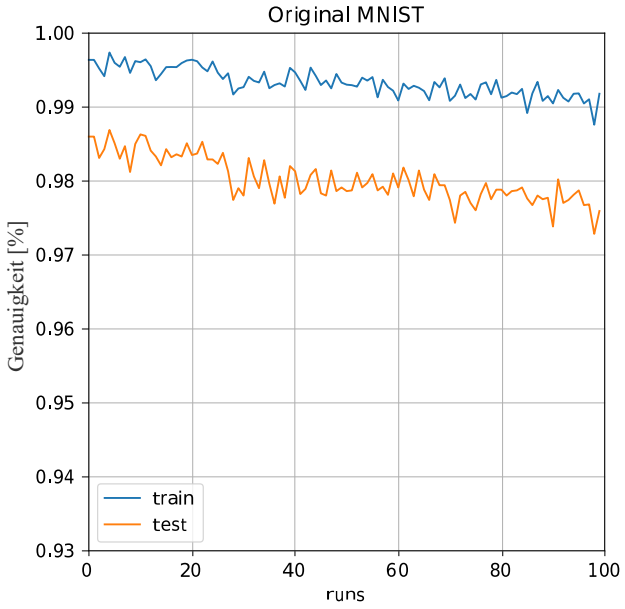

Abbildung 4.13: Verlauf der Klassifizierungsgenauigkeit des neuronalen Netzes über das Konvergenztraining hinweg.

Dieses Ergebnis wirft die Frage auf, warum das Hinzufügen der realistischen zusätzlichen Trainingsdaten zu einer leichten Verschlechterung geführt hat. Eine Erklärung dafür ist, dass die neuen Bilder zu stark von den originalen MNIST-Daten abweichen. Die Fähigkeit, diese Bilder korrekt zu klassifizieren, hilft dem Modell nicht dabei, seine MNIST-Genauigkeit zu erhöhen. Diese war bereits bei etwa 99 % und hatte damit kaum Spielraum zur Verbesserung. Um diese Hypothese zu überprüfen, wurde im Rahmen dieser Arbeit ein herausforderndererer, dem MNIST-Datensatz ähnlicher Handschriftzifferndatensatz erstellt.

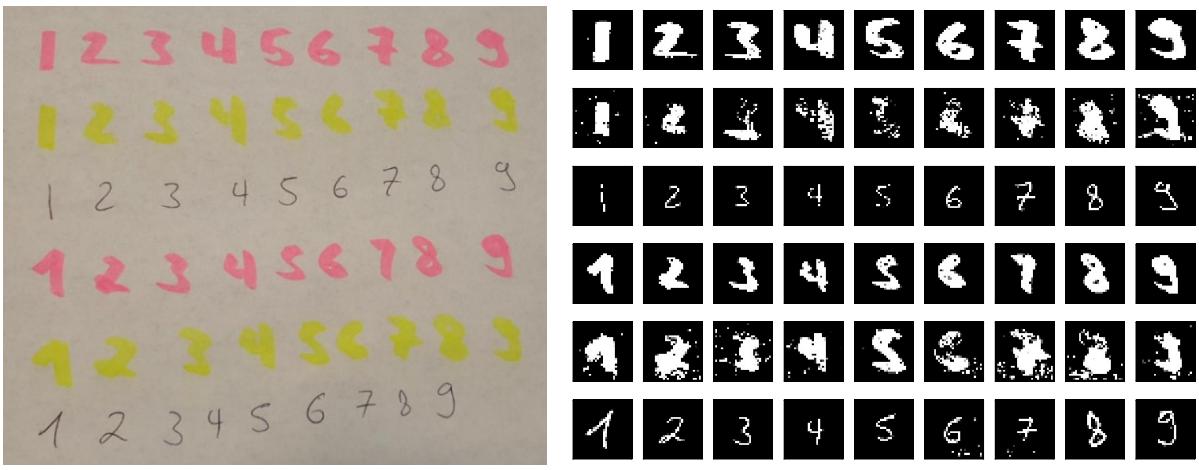

Abbildung 4.14: Neuer MNIST-ähnlicher Datensatz. Links Originalaufnahme auf Papier und rechts digitalisierte Bilder

Der neue Datensatz umfasst eine Sammlung von handgeschriebenen Zahlen von eins bis neun, die mit verschiedenen Stiften erstellt wurden. Diese unterschiedliche Stiftauswahl führt zu Variationen in der Strichstärke sowie in der Intensität des digitalen Rauschens. Das Rauschen ist bei den gelben Neon-Stiften bewusst vorhanden, um die Auswirkungen von Pixelfluktuationen zu untersuchen. Der Digitalisierungsprozess der Bilder erfolgte mittels globaler Binarisierung, wobei jedes Bild auf eine Auflösung von 100x100 Pixel beschnitten wurde. Abbildung 4.14 zeigt die Originalbilder. Die verwendeten Schwellenwerte für die Binarisierung sind in Tabelle 4.2 dokumentiert.

Tabelle 4.2: Schwellwerte der Binarisierung für die Digitalisierung

Klasse y	1	2	3	4	5	6	7	8	9
Reihe 1	155	155	155	155	155	155	155	155	155
Reihe 2	163	166	166	166	168	168	167	164	164
Reihe 3	155	155	155	155	155	155	155	155	155
Reihe 4	155	155	155	155	157	157	157	157	157
Reihe 5	162	164	164	164	164	164	164	162	161
Reihe 6	150	150	150	150	150	150	150	150	150

Abbildung 4.15 zeigt die Klassifizierungsergebnisse des Vergleichsmodells und des Modells, das mit den synthetischen Daten trainiert wurde. Der Vergleich zeigt, dass das Modell durch die synthetischen Daten nicht ausschließlich verbessert wurde. Drei Bilder werden nur durch das Vergleichsmodell korrekt klassifiziert. Dagegen stehen allerdings 12 Bilder, die nur durch das mit synthetischen Daten trainierte Modell erkannt wurden. Die Genauigkeit der Klassifizierung auf dem neu erstellten realen Testdatensatz konnte durch den Einsatz des Datengenerators von 53 % auf 70 % erhöht werden.

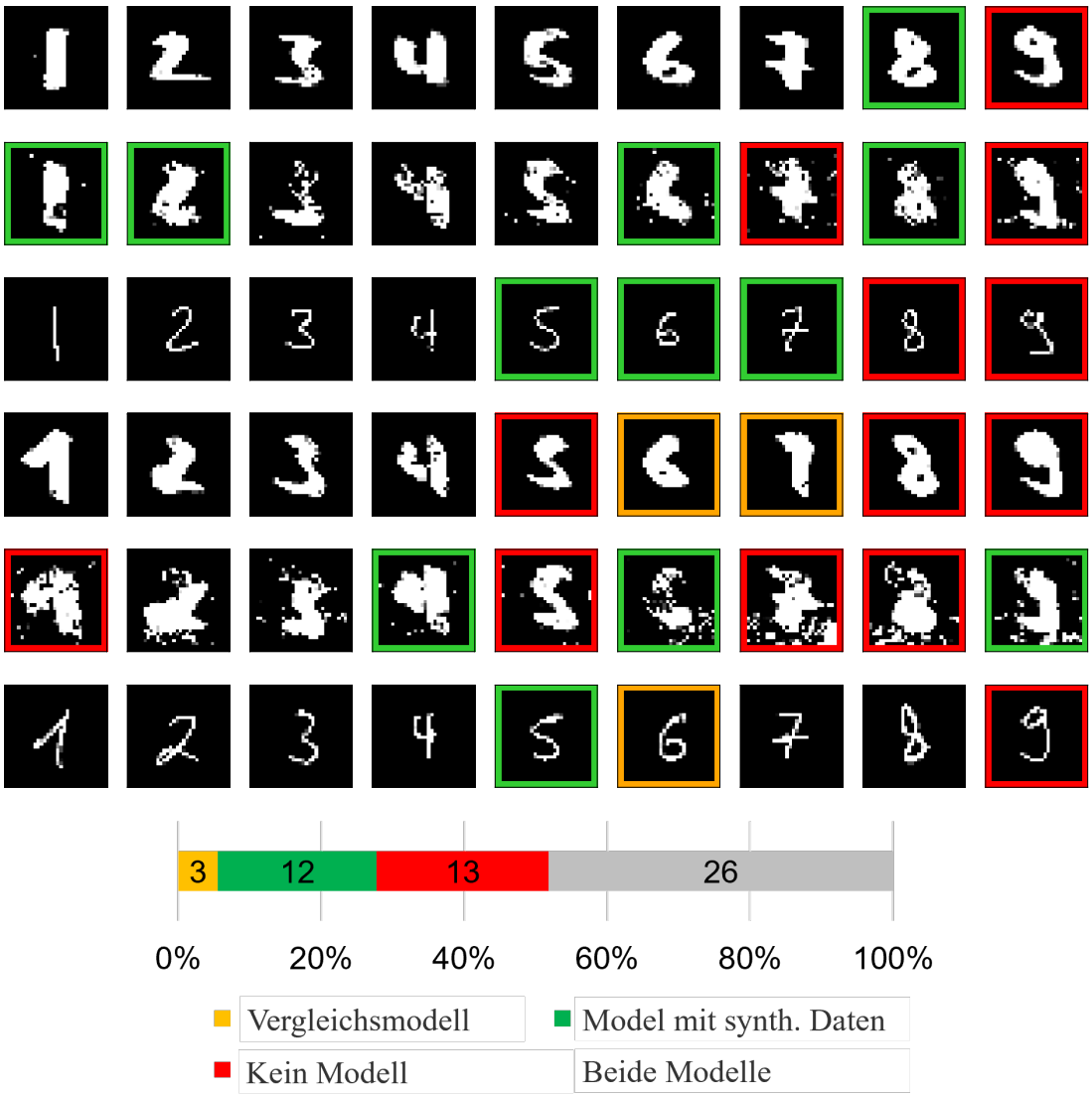

Abbildung 4.15: Klassifizierungsergebnis auf dem neu erstellten Datensatz. Die farbigen Ränder zeigen, von welchem Modell das Bild korrekt klassifiziert wurde.

Die Verbesserungen sind hauptsächlich bei Bildern mit vielen Störungen und sehr schmalen Stiftbreiten zu erkennen. Diese Erkenntnis zeigt erneut, dass der Trainingsdatengenerator es geschafft hat, Trainingsbilder zu erzeugen, die sich in realistischer Weise von den MNIST-Daten unterscheiden, ohne dass er dazu explizit programmiert wurde. Zusammenfassend konnte gezeigt

werden, dass der Trainingsdatengenerator realistische und nützliche Trainingsdaten erzeugen kann, die die Handschriftenerkennung verbessert hat.

Schlussfolgerung: *Die Experimente zeigen, dass der automatisierte Trainingsdatengenerator effektiv realistische Bilddaten erzeugen kann, die zu einer Klassifizierungsänderung führen, ohne dass diese Transformationen vorgegeben wurden. Im Vergleich zur FGSM-Methodik erhöht die Latentvektoranpassung die Datenvarianz und erzeugt realistische und leistungssteigernde Trainingsdaten, die das Modell widerstandsfähiger und genauer machen. Das Konvergenztraining, das die Trainingsdatenmenge verdoppelt, verbessert die Genauigkeit des Modells, obwohl es zu einer minimalen Verschlechterung auf den originalen MNIST-Daten kommt. Insgesamt steigert der Trainingsdatengenerator die Performanz bei der Handschriftenerkennung durch die Einführung realistischer synthetischer Daten.*

4.3 Evaluierungsfall 2: 5G-Positionsbestimmung

In diesem Kapitel wird das Konzept auf das Anwendungsbeispiel der 5G-Positionsbestimmung angewendet, ein Regressionsproblem aus der Automatisierungstechnik. Zunächst ist der Anwendungsfall und der Datensatz beschrieben, gefolgt von der Umsetzung des Konzepts. Abschließend erfolgt eine Evaluation der Ergebnisse.

4.3.1 Anwendungsfall

4.3.1.1 Beschreibung

Spezialisierte Technologien für die Positionierung in Innenräumen sind unerlässlich, da herkömmliche Außenpositionierungslösungen, wie globale Satellitennavigationssysteme (GNSS), in Innenräumen keine ausreichende Genauigkeit bieten. Systeme zur Lokalisierung in Innenräumen, häufig in Form von Echtzeit-Ortungssystemen (RTLS), stellen eine Schlüsseltechnologie für zahlreiche aktuelle Entwicklungen im Produktionssektor dar. Insbesondere die jüngsten Fortschritte bei digitalen Zwillingen in der Fertigung basieren auf der präzisen Positionierung physischer Objekte in Innenräumen, um eine exakte Synchronisierung mit ihren digitalen Gegenstücken zu gewährleisten [124]. Weitere Einsatzgebiete der Indoor-Positionierung umfassen die Verfolgung von Produkten und Werkzeugen sowie die Überwachung von Produktionsprozessen. Zudem wird sie zur Optimierung des Produktionslayouts, zur Nachverfolgung von Gabelstaplern oder FTS/AMR-Flotten und im Flottenmanagement eingesetzt [125].

Der aktuell weltweit implementierte fünfte Standard für breitbandige Mobilfunknetze, bekannt als 5G, stellt einen bedeutenden technologischen Fortschritt dar. Das Low-Band 5G, welches als direkter Nachfolger des LTE-Netzes fungiert, wird bereits von vielen Verbrauchern genutzt. Für die Industrie jedoch ist die High-Band-Version von 5G, die speziell für dicht besiedelte Gebiete und Produktionsumgebungen konzipiert ist, von größerer Bedeutung. High-Band 5G bietet bemerkenswerte Vorteile wie Gigabit-Geschwindigkeiten, minimale Latenzzeiten, die Unterstützung einer großen Anzahl von Geräten und eine niedrige Fehlerrate, was wiederum die Entwicklung neuer Anwendungen ermöglicht. Ein wesentliches Unterscheidungsmerkmal von 5G im Vergleich zu seinen Vorgängern ist seine Softwaredefinierbarkeit. Nutzer können die Netzleistungsmerkmale an ihre spezifischen Anwendungsanforderungen anpassen, was 5G zu einem Schlüsselelement in der intelligenten Fertigung macht [126].

Die Verwendung von 5G als Funktechnologie in Ortungssystemen bietet zahlreiche Vorteile. Eigenschaften wie die geringe Latenzzeit, die Fähigkeit, eine Vielzahl von Geräten zu unterstützen, und die hohe Zuverlässigkeit sind gerade für RTLS von großer Bedeutung. Zudem führt die

Nutzung derselben Hardware für Kommunikation und Ortung zu einer Verringerung der Gesamtkomplexität des Systems, was wiederum eine Kostenreduktion des Gesamtsystems ermöglicht.

Die Positionsbestimmung mittels Funktechnologien beruht wesentlich auf der Datenverarbeitung. Hierbei wird die Position eines Objekts während des Betriebs durch die Analyse der empfangenen Daten ermittelt. In den letzten Jahren hat die Industrie verstärkt auf Deep-Learning-Methoden zurückgegriffen, um bei der Datenverarbeitung hohe Leistungsfähigkeit in Bezug auf Genauigkeit, Entwicklungseffizienz und Verarbeitungsgeschwindigkeit zu erreichen [125]. Zusätzlich haben Bemühungen zur Standardisierung in den Bereichen der Datenverwaltung und -erfassung die Implementierung und Effektivität datengetriebener Lösungen deutlich verbessert [127], [128] (eigene Vorarbeiten) . Angesichts dieser Entwicklungen liegt unser Fokus auf datengetriebenen Ansätzen, insbesondere auf maschinellen Lernmethoden für die 5G-basierte Positionsbestimmung.

Die Nutzung von 5G-Technologie in Kombination mit Deep Learning für die Indoor-Ortung stellt eine Herausforderung im Bereich der datengetriebenen Problemlösung dar. Speziell die Sammlung und Auswahl relevanter, annotierter Daten ist eine komplexe Aufgabe. Die komplizierten Wechselwirkungen des Funksignals mit der Umgebung, wie etwa Reflexionen und Interferenzen durch andere Systeme in einer Produktionshalle, lassen sich nur schwer vollständig simulieren und für menschliche Experten komplex zu analysieren. Trotz dieser Herausforderungen ist der Bedarf an fortschrittlichen Ortungssystemen für Innenräume signifikant und unübersehbar. Aufgrund des hohen Aufwands, der mit der Sammlung der teuren realen Daten einhergeht, ist der Einsatz von synthetischen Daten für diesen Anwendungsfall besonders sinnvoll.

In dem hier beschriebenen zweiten Anwendungsfall werden relevante Trainingsdaten generiert, um ein Deep-Learning-System am Beispiel der 5G-Positionierung zu verbessern. Der gezielte Datengenerator wird verwendet, um Daten zu erzeugen, die vom Deep-Learning-Modell zunächst falsch positioniert werden. Der Kerngedanke dabei ist, dass, wenn das Modell Schwierigkeiten hat, einen generierten Datenpunkt korrekt zu positionieren, dieser Datenpunkt besonders wertvoll für das Training ist, um die Leistungsfähigkeit des Modells zu steigern. Durch diesen Ansatz wird sichergestellt, dass die generierten Daten direkt auf die Verbesserung der Modellgenauigkeit abzielen.

4.3.1.2 Datensatz

Im Rahmen des öffentlich geförderten Forschungsprojekts Synergieregion wurde ein 5G Industrie-Positionierungsdatensatz erstellt, der für diesen Anwendungsfall genutzt wird. Der 5G-Industrie-Positionierungsdatensatz entstand im 5G-Testfeld des Forschungscampus Arena2036 an der Universität Stuttgart. Abbildung 4.16 veranschaulicht den 5G-Positionierungsbereich von 7 m x 18 m innerhalb dieses Testfeldes. Für die Datensammlung bewegte sich der automatisiert fahrende Roboter (Automated Guided Vehicle - AGV) mit einem 5G-Sender entlang der Längsachse des Testfeldes, wobei er regelmäßig stoppte, um den Datenumfang zu vergrößern. Alle 100 Millisekunden sendete das AGV ein Sondierungsreferenzsignal (SRS) über sein 5G-Funkmodul aus.

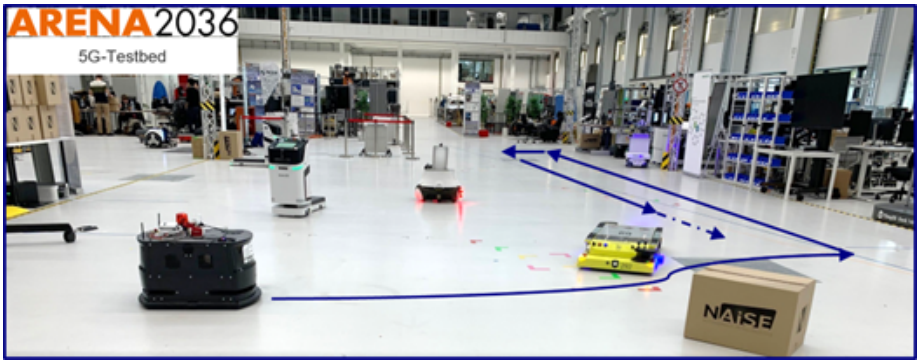

Abbildung 4.16: Fotografie eines AGV während es zur Datenerfassung durch den Shopfloor der Arena2036 fährt.

Der 5G-Positionierungsbereich ist von vier Sendern und Empfängern umgeben, die die empfangenen SRS aufzeichnen. Jeder Empfänger verfügt über acht Antennen, die in einem quadratischen Raster angeordnet sind. Die empfangenen Signale durchlaufen zunächst eine Analog-Digital-Umwandlung und anschließend eine Digital-Down-Conversion. Dies führt zu 4384 komplexen Baseband-Abtastwerten im Zeitbereich für jede Antenne. Von diesen Abtastwerten sind 288 Teil des zyklischen Präfixes, das als Schutzintervall fungiert, um die Intersymbol-Interferenz zu minimieren. Die restlichen Zeitabtastwerte werden mittels schneller Fourier-Transformation (FFT) in den Frequenzbereich überführt, was in 4096 komplexen Kanalkoeffizienten resultiert, die einzelne Subcarrier repräsentieren. Dabei wird jedoch nur jeder zweite Subcarrier von den SRS belegt, und es bleiben leere Subcarrier vor allem an den Spektralrändern, was die Anzahl der nutzbaren Kanalkoeffizienten auf 1632 reduziert. Jeder dieser Kanalkoeffizienten ist eine komplexe Zahl, bestehend aus einem Real- und einem Imaginärteil. Zur Erfassung der Position des AGV während der Datensammlung kam eine kommerzielle UWB-Innenraumortungstechnik zum Einsatz. Durch den Abgleich der Zeitstempel der UWB-Positionen mit den empfangenen 5G-Signalen konnten die gesammelten Daten entsprechend zugeordnet werden.

Für jeden Standortpunkt werden die Daten in Form von

(*Empfänger)x(Antennen)x(Kanalkoeffizienten)x(Real/Img).*

Dies führt zu einem sehr hochdimensionalen Tensor für jeden Standortpunkt in der Form von

(*4)x(8)x(1632)x(2)*

5G nutzt das orthogonale Frequenzmultiplexverfahren mit mehreren Eingängen und Ausgängen als Luftschnittstelle. Bei dieser Art der Systeme korrelieren die Koeffizienten benachbarter Subcarrier [129]. Daher wird eine Merkmalsdimensionenreduktion durch Mittelwertbildung über benachbarte Unterträger N durchgeführt. Anschließend werden die ersten beiden Dimensionen in eine Datenform von

(*32)x(32)x(2)*

transformiert, was als ein kleines quadratisches Bild mit einem fehlenden Farbkanal interpretiert werden kann. Dieses Vorgehen hat den Vorteil, dass Software-Bibliotheken und Architekturen von neuronalen Netzen, die für Bilddaten entwickelt wurden, für die Realisierung genutzt werden können.

4.3.2 Realisierung

In diesem Kapitel geht es um die Realisierung des in Kapitel 3 vorgestellten Konzepts für den Evaluierungsfall 2. In Abbildung 4.17 ist die gesamte Realisierung schematisch dargestellt. Das Kapitel beschreibt die Realisierungsaspekte, unterteilt nach den Komponenten des Konzepts: Generatives neuronales Netz, Ergebnisüberprüfungskomponente und Steuerungskomponente.

4.3.2.1 Generatives neuronales Netz

Im ersten Evaluierungsfall kam ein CGAN, das aus convolutional Schichten aufgebaut ist, zum Einsatz, um Bilder zu generieren. Diese Architektur wurde auch für die Verarbeitung der 5G-

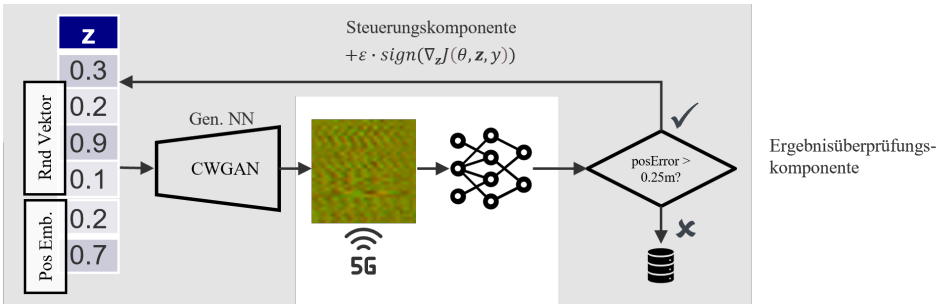

Abbildung 4.17: Übersicht der Realisierung des Trainingsdatengenerators für den 2. Anwendungsfall. Das gelbe Bild ist eine Visualisierung der vorverarbeiteten Kanalantwort der Antennen.

Daten erprobt, allerdings ohne den gewünschten Erfolg. Bei den durchgeführten Experimenten trat das Phänomen des Mode Collapse auf, was bedeutet, dass das generative Modell dazu neigte, sehr ähnliche Daten zu produzieren, anstatt eine Vielfalt an unterschiedlichen Ergebnissen zu generieren.

Um das Training von GANs zu stabilisieren und Mode Collapse zu verhindern, sind WGANs geeignet. Um eine bessere Konvergenz zu gewährleisten, wurde in diesem Evaluierungsfall eine Loss-Funktion eingesetzt, die auf der Wasserstein-Metrik basiert. Die Wasserstein-Metrik ermöglicht eine effektive Darstellung von Wahrscheinlichkeitsmaßen, was zu verbesserten Ergebnissen in der Modellierung führen kann. Für eine detaillierte Herleitung von WGANs kann auf [100] verwiesen werden. Im folgenden Abschnitt sind die konkreten Aspekte, die für die Realisierung des Evaluierungsfalls 2 verwendet wurden, zusammengefasst.

Die Wasserstein-Loss-Funktion für den Diskriminator ist:

$$\min_{D} -(\mathrm{E}_{\underline{x}\sim p_{data}}\left[D\left(\underline{x}\right)\right] - \mathrm{E}_{\underline{z}\sim p_z}[D(G(z)]) \tag{16}$$

Und für den Generator:

$$\min_{G} -(\mathrm{E}_{\underline{z}\sim p_z}[D(G(z)]) \tag{17}$$

Für das Training des Diskriminators wird der Loss berechnet, indem der Erwartungswert der vom Diskriminator für reale Bilder $E_{x\sim p_{\text{data}}}[D(x)]$ ausgegebenen Wahrscheinlichkeiten für das Label $y_i = 1$ dem Erwartungswert der für die vom Generator erzeugten synthetischen Bilder $E_{z\sim p_z}[D(G(z))]$ ausgegebenen Wahrscheinlichkeiten für das Label $y_i = -1$ gegenübergestellt wird. Beim Training des Generators hingegen wird der Loss durch den Erwartungswert der Wahrscheinlichkeiten für die Ausgaben der erzeugten Bilder $\mathrm{E}_{\underline{z}\sim p_z}[D(G(z)]$ in Bezug auf das Label $y_i = 1$ bestimmt. Durch die Verwendung dieser Loss-Funktionen wird das CGAN zum stabileren CWGAN, was in diesem Evaluierungsfall verwendet wird.

Die Architektur des Generators besteht aus 4 Schichten, in denen jeweils eine zweidimensionale transponierte Convolution mit anschließender Batch-Normalisierung durchgeführt wird. Die ersten drei Schichten verwenden die ReLU-Funktion als Aktivierungsfunktion, die letzte Schicht den Tangens hyperbolicus.

4.3.2.2 Ergebnisüberprüfungskomponente und Steuerungskomponente

Die Ergebnisprüfungskomponente bewertet die Positionierung des neuronalen Netzes, das auf dem 5G-Lokalisierungsdatensatz trainiert wurde, im Hinblick auf die Präzision der Positionierung, die während der Generierung der Eingaben für das Netzwerk in den **z**-Vektor eingebettet wurde.

Auch die Anzahl der bisher durchgeführten Iterationen des 5G-Datengenerators wird bei der Entscheidung betrachtet. Auf der Grundlage dieser Daten entscheidet die Komponente, ob das vom Generator erstellte 5G-Signal dem neuen synthetischen Datensatz hinzugefügt wird. Falls ein synthetischer 5G-Datensatz nicht hinzugefügt wird, bestimmt die Komponente, ob der Generierungsprozess mit einer weiteren Iteration fortgesetzt oder abgebrochen wird.

Das Positionierungsproblem ist ein Regressionsproblem, daher ist die Entscheidung, ob die Ausgabe des Netzes korrekt ist, weniger eindeutig als bei einem Klassifizierungsproblem. An dieser Stelle muss ein domänenspezifischer Genauigkeitsschwellwert festgelegt werden. In der Domäne der Innenraum-Positionierung für die Intralogistik gelten 50 cm als oberes Limit für Anwendungen, die auf eine präzise Position angewiesen sind und 25 cm als genau [125]. Daher wurde der Genauigkeitsschwellwert auf 25 cm gesetzt. Wenn das vom Datengenerator generierte 5G-Signal nicht präzise genug positioniert wird, wird das 5G-Signal dem synthetischen Trainingsdatensatz hinzugefügt. Wenn das generierte 5G-Signal korrekt erkannt wird, dann wird das 5G-Signal nicht hinzugefügt, stattdessen wird eine weitere Iteration durchgeführt, bis die maximale Anzahl an Iterationen überschritten ist oder der Vorgang divergiert. Die Ergebnisüberprüfungskomponente vergleicht ebenfalls die Positionierungsgenauigkeit zwischen einzelnen Iterationen und wenn diese nicht abnimmt, sondern zunimmt, dann wird davon ausgegangen, dass der Vorgang divergiert und er wird abgebrochen.

Auch für diesen Anwendungsfall wird die Steuerungskomponente wie im Konzeptkapitel 3.3.3 mit dem FGSM-Algorithmus implementiert. Als ϵ wurde zunächst der Wert 0,2 gewählt, weil er sich in Anwendungsfall 1 als guter Kompromiss zwischen einer hohen Diversität der synthetischen Daten und zuverlässigen Realitätstreue herausgestellt hat. Es wurden auch Vergleichsexperimente mit anderen Werten für ϵ durchgeführt, aber kein besserer Wert gefunden. Das unterstreicht, dass dieser Hyperparameter übertragbar ist, was daran liegt, dass er die Änderung des latenten Raumes des Generators beeinflusst und keinen direkten Zusammenhang zu der Domäne der Daten hat.

4.3.3 Evaluierung

In diesem Abschnitt ist die Evaluierung des im vorherigen Kapitel realisierten Konzepts mit dem industriellen 5G-Positionierungsdatensatz beschrieben. Dabei wird untersucht, ob die Innenraumpositionierung in einem industriellen Umfeld mit synthetischen 5G-Empfangsdaten verbessert werden kann.

4.3.3.1 Positionierungsmodell

Das Positionierungsmodell besteht aus insgesamt sieben Schichten. Es beginnt mit drei Convolutionsschichten, die aufeinanderfolgen und darauf abzielen, verschiedene Merkmale aus den Eingabebildern zu extrahieren. Diese Schichten nutzen jeweils einen 3x3-Kernel, um die räumlichen Merkmale zu erkennen und zu verstärken, wobei sie eine Schrittweite von 1 und ein Padding von 2 einsetzen, um die Dimensionalität der Daten zu erhalten. Zwischen den Schichten finden Max-Pooling Operationen statt. Nach den Convolutionsschichten folgt eine fully-connected Schicht, die die hochdimensionalen Merkmale aus den Faltungsschichten in eine kompaktere Form überführt, die für die Lokalisierung nützlich ist. Diese Schicht dient als Brücke zwischen den räumlichen Merkmalen und den Lokalisierungsmerkmalen. Ein Dropout wird nach der ersten fully-connected Schicht angewendet, um das Modell robuster zu machen, indem zufällig einige Verbindungen während des Trainings deaktiviert werden. Dies hilft, Überanpassung zu vermeiden und die Generalisierungsfähigkeit des Modells zu verbessern. Abschließend folgen zwei weitere fully-connected Schichten, die die Merkmale weiter verdichten und für die Endlokalisierung vorbereiten. Die letzte dieser Schichten produziert die endgültige Lokalisierung. Das Modell wurde mit 16566 Positionierungssamples für 70 Epochen trainiert, was auf dem Labor-PC mit einer GeForce RTX 2080 GPU und der AMD Ryzen Threadripper 2920X CPU etwa 4,5h gedauert hat.

4.3.3.2 Generierung der synthetischen Daten

In diesem Abschnitt ist beschrieben, welche synthetischen Daten für den Anwendungsfall 2 generiert werden. Im Anwendungsfall 1 wurden für jede Klasse synthetische Daten generiert. Da es sich im Gegensatz dazu bei dem 5G-Positionierungsanwendungsfall um ein Regressionsproblem handelt, existiert keine natürlich vorgegebene Menge an Koordinaten, für die es sinnvoll ist, synthetische Daten zu generieren. Daher wird der Generator in diesem Fall als spezifisches Verbesserungswerkzeug in einer bestimmten Positionierungsregion eingesetzt. Über die Genauigkeit des Positionierungsmodells auf den Testdaten lässt sich klar eine Region in dem 5G-Testbed identifizieren, in der das Positionierungsmodell sehr ungenau ist (Abbildung 4.18). Diese Region wird in ein Punktraster mit einem Abstand von 0,5 m zwischen den einzelnen Punkten unterteilt. Für jeden Punkt generiert der Datengenerator 10 neue 5G-Datenpunkte, die zu den echten Trainingsdaten hinzugefügt werden. Das Positionierungsmodell wird dann erneut mit allen Daten trainiert.

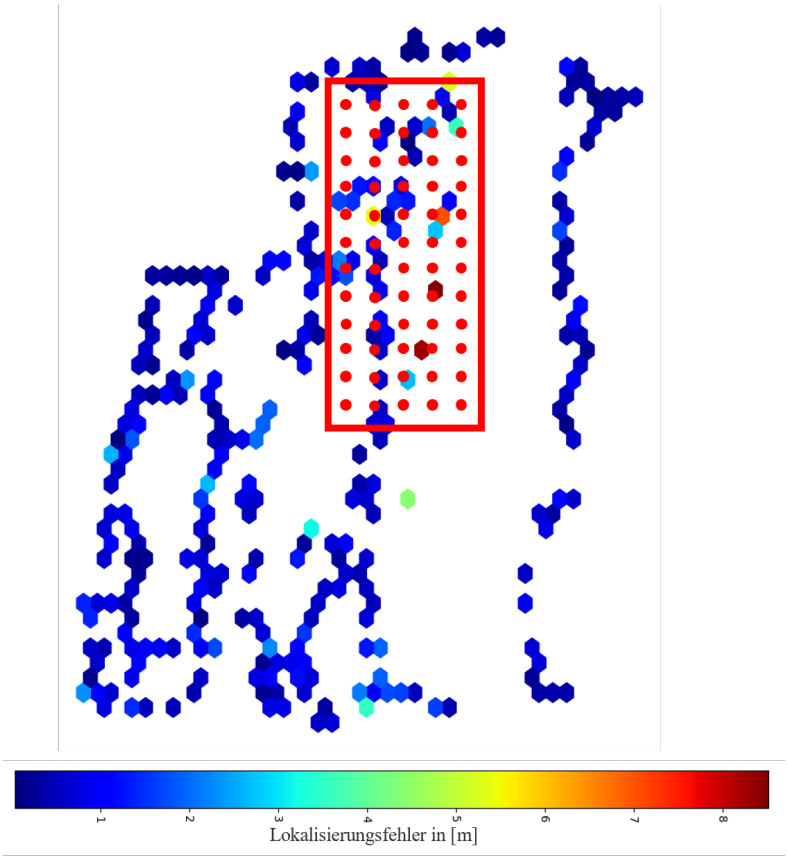

Abbildung 4.18: Histogramm der Positionierungsgenauigkeiten des Positionierungsmodells im 5G-Testbed. In dem rot umrandeten Bereich ist die Genauigkeit besonders schlecht und soll durch synthetische Daten verbessert werden.

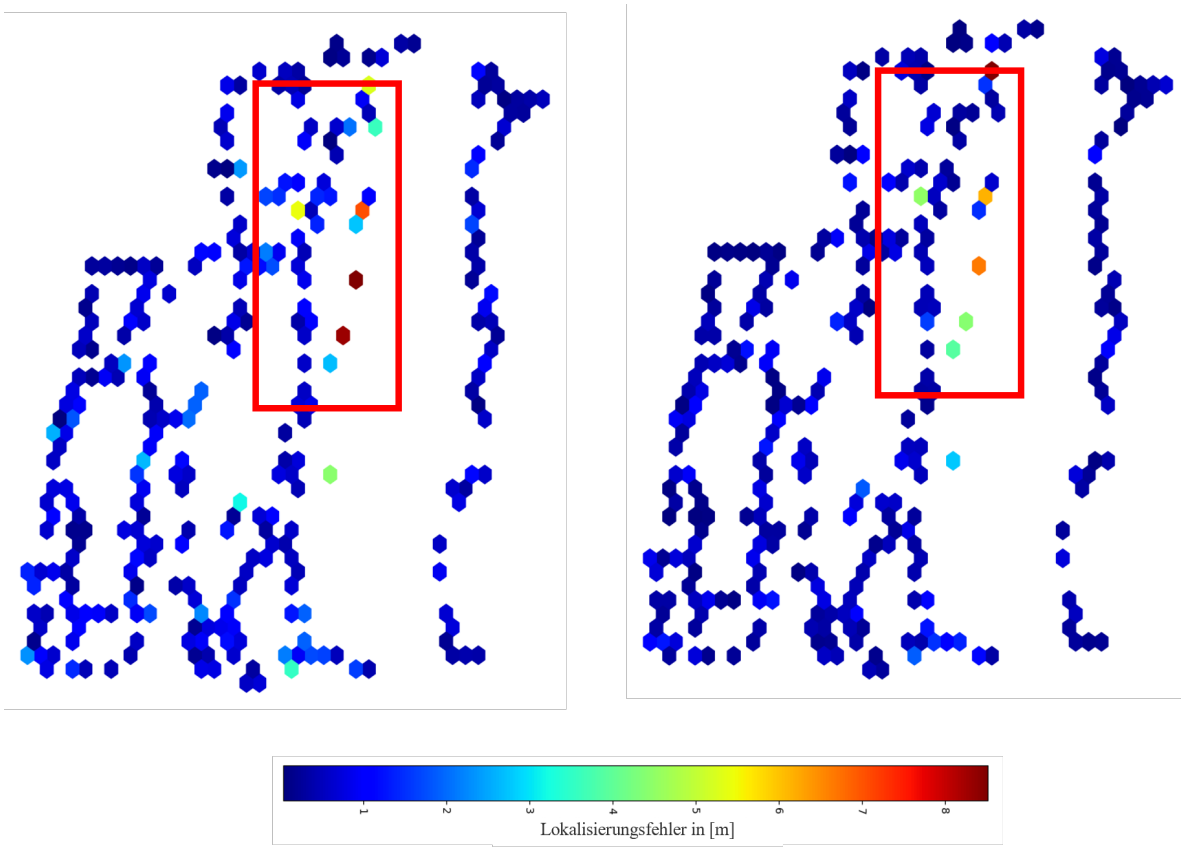

Abbildung 4.19: Histogramme der Positionsfehler. Training nur mit realen Daten (links). Training mit realen und generierten Daten (rechts).

4.3.3.3 Ergebnisse

In diesem Abschnitt werden die mit den Testdaten erzielten Positionierungsergebnisse vorgestellt und die Wirksamkeit des vorgeschlagenen Ansatzes zur Verbesserung der 5G-Positionierungsgenauigkeit durch die Verwendung synthetischer Daten bewertet. Alle gesammelten realen Daten wurden in eine typische Aufteilung von 70 % Training, 20 % Validierung und 10 % Test aufgeteilt. Das CNN ohne synthetische Daten wurde ausschließlich mit dem Trainingsdatensatz trainiert, während die Suche nach Hyperparametern zur Bestimmung seiner Architektur mit den Validierungsdaten durchgeführt wurde. Das CWGAN wurde hingegen lediglich mit den realen Trainingsdaten trainiert. Das neue CNN wurde mit den Trainingsdaten und den generierten synthetischen Daten trainiert. Abbildung 4.19 zeigt die Histogramme der Positionierungsfehler der beiden Modelle. Jedes Sechseck entspricht einem Histogramm-Bin, das mit dem Mittelwert des Positionsfehlers aller Positionen eingefärbt ist, die sich innerhalb des Sechsecks befinden. Das linke Diagramm enthält die Positionsfehler des CNN, das nur auf den real gesammelten Trainingsdaten trainiert wird, während das rechte Diagramm dem CNN entspricht, das mit den zu den Trainingsdaten hinzugefügten synthetischen Daten trainiert wurde.

Das rote Rechteck hebt die Region hervor, in der, wie im vorherigen Abschnitt beschrieben, zusätzliche synthetische Daten erstellt wurden. Dies ist auch die Region, in der im linken Diagramm sehr hohe Positionsfehler zu beobachten sind. Vergleicht man die Farbveränderungen zwischen dem linken und dem rechten Diagramm, so kann man deutlich eine Verbesserung des Positionsfehlers erkennen. Zusätzlich kann sogar eine Verbesserung außerhalb des roten Rechtecks festgestellt werden. Im unteren linken Bereich unterhalb des Rechtecks haben sich die cyanfarbenen Bins im rechten Diagramm in ein dunkleres Blau verändert. Allerdings gilt dies nicht für alle Regionen. Links im Rechteck befindet sich ein gelber Positionierungsbereich, in dem der Positionierungsfehler durch das Hinzufügen synthetischer Daten zunahm.

Abbildung 4.20 zeigt ein Histogramm aller Positionsfehler, unabhängig von deren Koordinaten. Es ist deutlich zu erkennen, dass die Bins auf der linken Seite durch die Hinzufügung synthetischer Daten zu den Trainingsdaten vergrößert wurden, was bedeutet, dass die Anzahl der Stichproben mit sehr geringen Positionsfehlern zunahm. Auch in den mittleren bis hohen Fehlerregionen ist eine gewisse Zunahme zu erkennen.

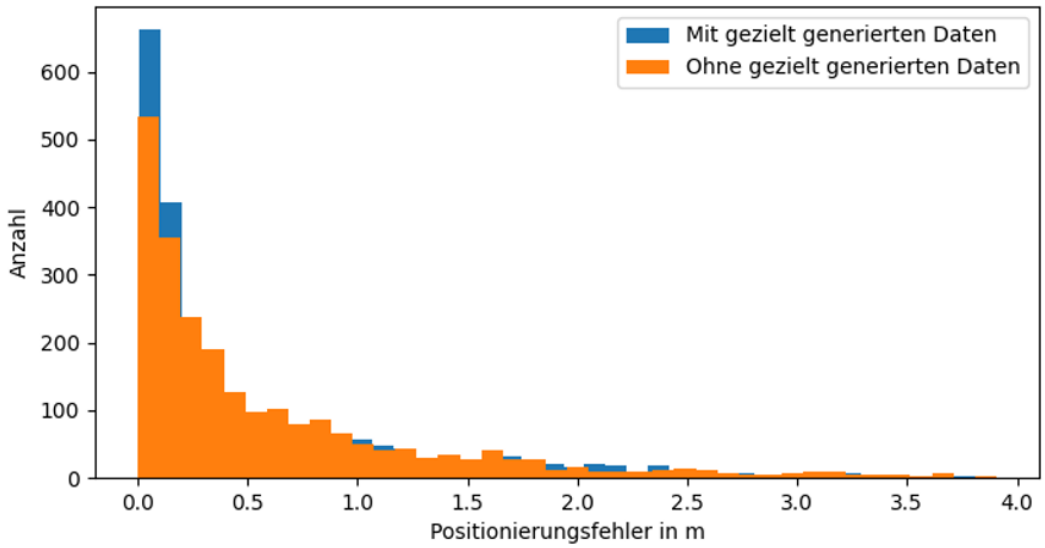

Abbildung 4.20: Histogramm aller Positionierungsfehler, unabhängig von ihren Koordinaten. Das Histogramm wird nur für Fehler bis 4,0 m dargestellt, da die Bins für höhere Fehler aufgrund ihrer geringen Höhe nicht angezeigt werden können.

Alle Positionierungsverfahren sind anfällig für Schwankungen, wobei ungenaue Ausreißer manchmal inmitten genauerer Messungen auftreten. Treten diese Ausreißer mit ausreichend geringer Häufigkeit auf, kompensieren Positionierungssysteme sie durch zusätzliche Algorithmen, die z.B. Wissen über Fahrgeschwindigkeit, Masse des Roboters und Bremsvorgänge nutzen. Abbildung 4.21 zeigt die kumulative Verteilungsfunktion der Positionierungsfehler der beiden CNNs. Die Wahrscheinlichkeit, dass ein Positionierungsfehler unter 0,5 m auftritt, ist um 5 %

höher, wenn synthetische Daten für das Training verwendet werden. Dies zeigt, dass synthetische Daten, die mit dem vorgeschlagenen Ansatz erstellt wurden, die Genauigkeit eines 5G-Positionierungs-CNN in der Praxis erhöhen können.

Dies wird zudem dadurch unterstrichen, dass sich der mittlere Positionsfehler (Median) von 0,32 m auf 0,24 m verbessert hat, was einer Verbesserung der Positionsgenauigkeit um 25 % entspricht.

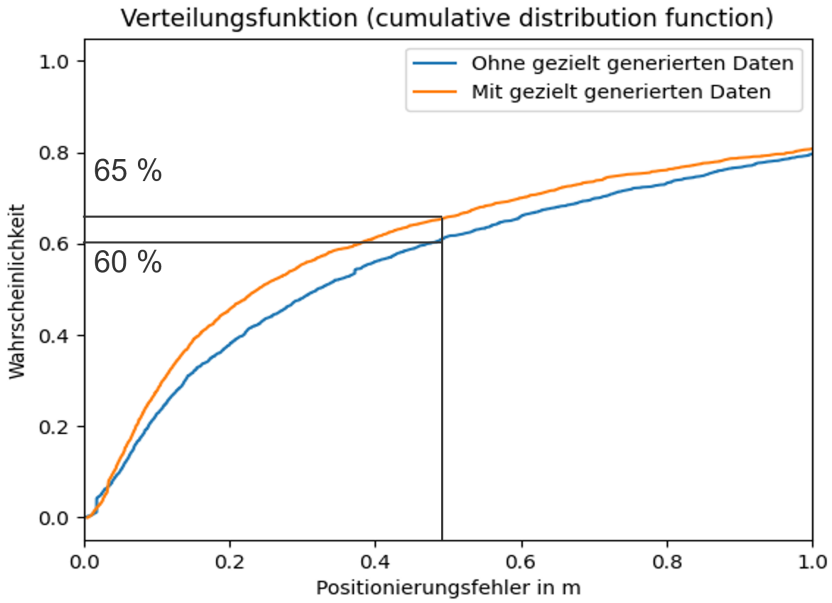

Abbildung 4.21: Verteilungsfunktion (cumulative distribution function) der Positionierungsgenauigkeiten. Die Wahrscheinlichkeit eines Positionierungsfehlers von unter 50 cm ist durch die Verwendung der generierten Daten um 5 % gestiegen.

Schlussfolgerung: *Die Ergebnisse zeigen, dass die Integration synthetischer Daten in das Training eines 5G-Positionierungs-CNNs die Genauigkeit signifikant verbessert. Dies deutet darauf hin, dass synthetische Daten eine effektive Methode zur Verbesserung der Positionsgenauigkeit in der Automatisierungstechnik darstellen können, insbesondere wenn reale Trainingsdaten begrenzt oder unvollständig sind. Die Reduktion des mittleren Positionsfehlers um 25 % bestätigt die Wirksamkeit des Ansatzes und unterstreicht das Potenzial synthetischer Daten zur Optimierung maschineller Lernmodelle.*

4.4 Evaluierungsfall 3: Industrieller Objektdetektor

In diesem Kapitel wird das Konzept auf das Anwendungsbeispiel der industriellen Objektdetektion angewendet, ein Problem der Objektdetektion aus der Automatisierungstechnik. Zunächst ist der Anwendungsfall und der Datensatz beschrieben, gefolgt von der Umsetzung des Konzepts. Abschließend erfolgt eine Evaluation der Ergebnisse.

4.4.1 Anwendungsfall

4.4.1.1 Beschreibung

Objektdetektion anhand von Bilddaten ist ein weitverbreitetes Problem, das mit maschinellem Lernen gelöst wird. In der Automatisierungstechnik wird es häufig in der Fertigung oder - wie in diesem Beispiel - in der Intralogistik eingesetzt. Die Objektdetektion beinhaltet nicht nur die Identifizierung, welche Objekte auf einem Bild vorhanden sind, wie es bei der Bildklassifizierung (Anwendungsfall 1) der Fall ist, sondern auch die genaue Lokalisierung dieser Objekte durch Umrahmung mit Bounding-Boxen. Während die Bildklassifizierung das gesamte Bild als Eingabe nimmt und ihm eine einzelne Kategorie zuordnet, geht die Objekterkennung einen Schritt weiter, indem sie jedes Objekt im Bild einzeln identifiziert und dessen Position angibt.

Abbildung 4.22: Transportbox, die durch das AGV in der Arena2036 erkannt werden soll.

Dadurch ermöglicht die Objekterkennung eine detailliertere Analyse von Bildinhalten, indem sie nicht nur das 'Was' (die Identität der Objekte), sondern auch das 'Wo' (ihre spezifische Lage im Bild) berücksichtigt. Es existieren zahlreiche vorgefertigte Datensätze, die Lösungen für gängige Problemstellungen der Objektdetektion bieten. Ein illustratives Beispiel hierfür ist der COCO-Datensatz, der Bilder mit 80 verschiedenen Objekten, darunter Menschen, Hunde und Autos, umfasst. Allerdings stoßen solche allgemeinen Datensätze in industriellen oder fabrikbezogenen Anwendungen an ihre Grenzen, da hier häufig die Identifizierung spezifischer Objekte wie

Transportkisten, Bauteile oder Roboter gefordert ist. In solchen Fällen erweisen sich die allgemeinen Datensätze als ungeeignet. Daher ist es notwendig, spezifische Daten eigenständig zu erfassen. Bei einem Mangel an Daten gestaltet sich die Implementierung mittels neuronaler Netze als besonders herausfordernd. Im Anwendungsfall 3 wird der synthetische Datengenerator eingesetzt, um Bilddaten für einen Objektdetektor in einem Produktionsumfeld zu generieren. Im Rahmen des öffentlich geförderten Forschungsprojekts Synergieregion sollte ein AGV in der Arena2036 in der Lage sein, eine spezielle Kiste zu erkennen. Es liegen keinerlei Trainingsdaten für diese Kiste vor. Ganz ohne Trainingsdaten funktioniert der in dieser Arbeit vorgestellte Ansatz nicht, deshalb wurde ein minimaler realer Datensatz erstellt.

4.4.1.2 Datensatz

Der minimale reale Datensatz besteht aus insgesamt 127 Bildaufnahmen von Transportboxen, die in der Arena2036 aufgenommen wurden. Die Bilder wurden händisch mit Bounding-Boxen gelabelt. Von den 127 Bildern werden 15 Bilder für das Training verwendet, 5 als Validierungsdatensatz und die übrigen 108 Bilder als Testbilder. Abbildung 4.23 zeigt beispielhaft 10 Trainingsbilder.

Abbildung 4.23: Bilder der Transportbox, die für das Training aufgenommen wurden.

4.4.2 Realisierung

Dieses Kapitel beschreibt die Realisierung des in Kapitel 3 vorgestellten Konzepts für den Evaluierungsfall 3. In Abbildung 4.24 ist die gesamte Realisierung schematisch dargestellt. Das Kapitel beschreibt die Realisierungsaspekte unterteilt in die Komponenten des Konzepts: Generatives neuronales Netz, Ergebnisüberprüfungskomponente und Steuerungskomponente.

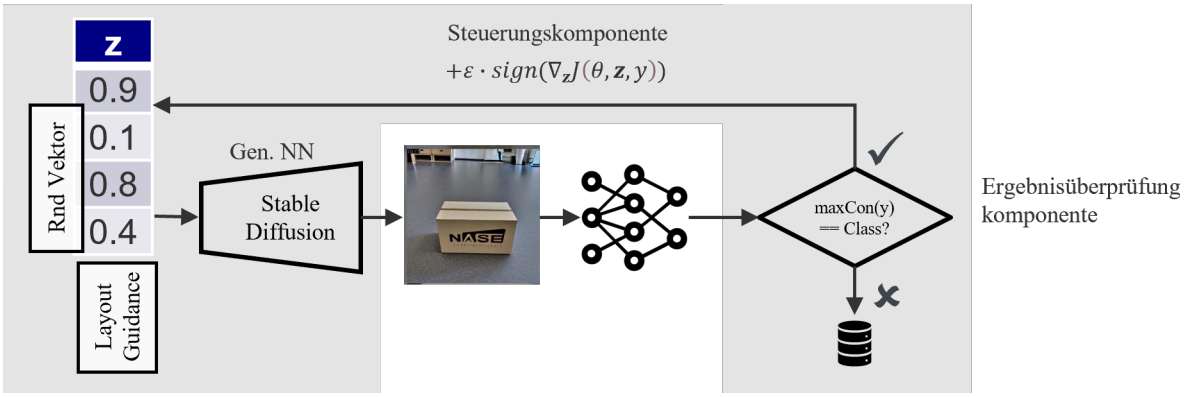

Abbildung 4.24: Übersicht über die Realisierung des Anwendungsfalls 3

4.4.2.1 Generatives neuronales Netz

In Anwendungsfall 3 wird kein generatives Netz von Grund auf neu trainiert, wie in Anwendungsfall 1 und 2, sondern ein bestehendes Foundation-Modell für den spezifischen Anwendungsfall angepasst. Als Foundation-Modell wird Stable Diffusion [130] eingesetzt, weil es zum Zeitpunkt der Anfertigung der Arbeit das beste frei verfügbare Large-Vision-Modell war. Bei Stable Diffusion handelt es sich um ein generatives Modell nach dem Prinzip des Text-zu-Bild Latent Denoising Diffusors, wie sie in Kapitel 2.3.1.2 beschrieben sind. Damit das Foundation-Modell in diesem Anwendungsfall eingesetzt werden kann, muss es auf zwei Arten erweitert werden: Das Foundation-Modell muss

1. … die Fähigkeit erlernen, genau diese Transportboxen zu generieren, obwohl diese nicht in seinen Trainingsdaten vorhanden sind.
2. … präzise ansteuerbar sein, damit festgelegt werden kann, wo genau auf dem synthetischen Bild die Transportbox generiert werden soll. In natürlichem Text im Prompt formulierte Bildkoordinaten sind nicht ausreichend genau.

Das Anlernen der dem Foundation-Modell bisher unbekannten Transportbox ist durch die Dreambooth-Methode umgesetzt. Die präzise Ansteuerbarkeit ist durch Layout Guidance umgesetzt. Die folgenden Abschnitte beschreiben die genaue Realisierung mit Dreambooth und Layout Guidance.

4.4.2.1.1 Dreambooth

Dreambooth [131] ist eine Finetuning-Methode, die es einem Diffusionsmodell ermöglicht, spezifische Objekte zu erlernen, indem diese Objekte mit einem eindeutigen Identifikator versehen werden. Im Gegensatz zum traditionellen Finetuning benötigt Dreambooth nur drei bis fünf Trainingsbeispiele des Zielobjekts, ohne dass eine textuelle Beschreibung der Bilder erforderlich ist. Das so trainierte Modell ist in der Lage, das gelernte Objekt in einer Vielzahl von Kontexten, einschließlich verschiedener Orte, Posen und Perspektiven, auf der Grundlage von Texteingaben zu reproduzieren. Die Implementierung erfolgt, indem allen Trainingsbildern das Format „a [Identifikator] [Klassenlabel]“ zugewiesen wird, wobei der [Identifikator] ein eindeutiges, dem Objekt zugeordnetes Token darstellt und der [Klassenlabel] die Objektkategorie angibt, z.B. Kiste, Auto oder Fahrrad. Bei der Wahl des Tokens ist es wichtig, ein selten verwendetes Token zu wählen, um Verwechslungen zu vermeiden. Allgemeine Begriffe wie „einzigartig“ sind ungeeignet, da sie häufig in anderen Zusammenhängen verwendet werden. Ebenso sind lange, zufällige Buchstabenfolgen nicht empfehlenswert, da der Tokenizer sie in einzelne Buchstaben oder Token zerlegen kann. Ein optimaler Identifikator besteht aus einer kurzen Zeichenfolge mit drei oder weniger Buchstaben, die in anderen Kontexten selten verwendet wird und vom Tokenizer als ein einziges Token erkannt wird.

Ein geeignetes Beispiel für einen Identifikator ist „sks“. Ruiz et al. [131] haben zwei Herausforderungen beim Fine-Tuning von Diffusionsmodellen identifiziert, die durch die Verwendung von Dreambooth gemildert werden können. Die erste Herausforderung ist das potenzielle Vergessen der sprachlichen Bedeutung durch das Diffusionsmodell während des Fine-Tunings. Die zweite Herausforderung ist die eingeschränkte Vielfalt oder Variabilität der generierten Bilder. Dreambooth adressiert diese Probleme durch einen klassenspezifischen Prior Preservation Loss. Die Überwachung des Fine-Tunings erfolgt direkt durch das Diffusionsmodell, indem vor dem Fine-Tuning Bilder erzeugt werden, die nur mit dem Klassenlabel (z.B. „a [Klassenlabel]“) versehen sind. Während des Fine-Tunings wird das Modell sowohl mit einem Beispiel des spezifischen Objekts als auch mit einem Beispiel der allgemeinen Klasse trainiert. Dabei wird das Diffusionsmodell in jedem Trainingsschritt zweimal angewendet: einmal auf ein verrauschtes Beispiel des Objekts (Rekonstruktion) und einmal auf ein verrauschtes Beispiel der Klasse (Prior Preservation). Für die Texteingabe wird „a [Bezeichner] [Klassenbezeichnung]“ für die Rekonstruktion und „a [Klassenbezeichnung]“ für die Prior Preservation verwendet.

Der gesamte Loss ergibt sich dann aus dem Loss der Rekonstruktion und aus dem Loss der Prior Preservation (siehe [131]):

$$\mathrm{L}_{\mathrm{Dreambooth}} \coloneqq L_{reconstruction} + \lambda L_{prior\ preservation} \tag{18}$$

Der jeweilige Loss wird nach Gleichung (18) aus der Differenz zwischen dem vorhergesagten Rauschen des Diffusionsmodells und dem tatsächlichen Rauschen berechnet. λ gibt die

Gewichtung für die Prior Preservation an. Anders als bei anderen Transfer Learning Verfahren wird bei Dreambooth das vollständige U-Net trainiert. Also werden keine einzelnen Layer eingefroren. Die verwendeten Hyperparameter sind: 10^{-6} für die Learning Rate, $\lambda = 1$ und 1000 Trainingsschritte. Für Dreambooth werden 5 Bilder aus dem Trainingsdatensatz verwendet.

4.4.2.1.2 Layout Guidance

Layout Guidance [132] ist eine Technik, die darauf abzielt, Diffusionsmodelle so zu beeinflussen, dass sie Objekte an bestimmten, vordefinierten Positionen innerhalb eines Bildes erzeugen. Diese Steuerung wird durch die gezielte Manipulation der Cross-Attention-Layer im U-Netz des Diffusionsmodells erreicht. Die Methode verwendet Eingaben wie Bounding-Boxen, um den Diffusionsprozess präzise zu steuern. Ein entscheidender Vorteil der Layout Guidance besteht darin, dass das Modell nicht neu trainiert werden muss, sondern lediglich in den Sampling-Prozess des bestehenden Diffusionsmodells eingegriffen wird. Dies ermöglicht eine sofortige Anwendung der Methode ohne die Notwendigkeit einer langwierigen oder ressourcenintensiven Trainingsphase. Im Folgenden wird näher beschrieben, wie Layout Guidance funktioniert und wie es implementiert wird, um die gewünschten Ergebnisse zu erzielen.

Die Cross-Attention-Schichten in Diffusionsmodellen können so angepasst werden, dass ein Token y_i spezifisch mit einer Bounding Box B assoziiert wird. Chen et al. beschreiben dazu zwei Methoden: Forward Guidance und Backward Guidance. Bei Forward Guidance wird eine Fensterfunktion g_u^{γ} verwendet, die innerhalb der Bounding Box konstante Werte und außerhalb der Bounding Box Werte von 0 annimmt, mit einem fließenden Übergang an den Rändern. Diese Fensterfunktion beeinflusst dann die Cross-Attention Maps, um die Aufmerksamkeit des Modells auf den Bereich innerhalb der Bounding Box nach dieser Gleichung zu lenken:

$$A_{ui}^{(\gamma)} \leftarrow (1-\lambda)A_{ui}^{(\gamma)} + \lambda g_u^{(\gamma)} \sum_v A_{vi}^{(\gamma)} \tag{19}$$

Dabei werden nur die Cross-Attention Maps $A_{ui}^{(\gamma)}$ des zu steuernden Tokens y_i verändert. λ kontrolliert, wie stark das Objekt gesteuert werden soll. Die Cross-Attention Map $A^{(\gamma)}$ ordnet ein Token einem räumlichen Ort zu. Die Summe am Ende der Gleichung stellt sicher, dass sich die Gesamtverteilung nicht ändert.

Backward Guidance manipuliert die Cross-Attention-Layer $A_{ui}^{(\gamma)}$ durch folgende Energiefunktion:

$$E\left(A^{(\gamma)}, B, i\right) = \left(1 - \frac{\sum_{u \in B} A_{ui}^{(\gamma)}}{\sum_u A_{ui}^{(\gamma)}}\right)^2 \tag{20}$$

Nun wird für jedes Cross-Attention-Layer der Gradient der Energiefunktion E durch die Backpropagation berechnet, um den Rauschvektor z_t nach folgender Gleichung zu aktualisieren:

$$z_t \leftarrow z_t - \sigma_t \eta \nabla_{z_t} \sum_{\gamma} E\left(A^{(\gamma)}, B, i\right) \tag{21}$$

Hierbei ist η ein Faktor, um die Stärke der Guidance zu kontrollieren, und σ_t ist abhängig vom Varianz-Schedule nach $\sigma_t = \sqrt{(1-\alpha_t)/\alpha_t}$. Um Bilder während des Samplings zu generieren, wird nach jedem Sampling-Schritt ein Update des Rauschvektors durch Berechnung des Gradienten durchgeführt. Der Ansatz der Backward Guidance ist in Abbildung 4.25 dargestellt.

Für die Realisierung wird Backward Guidance verwendet, allerdings wird sie, um Rechenzeit zu sparen, nur in den Upsample- und Midsample-Blöcken des U-Nets implementiert.

4.4.2.2 Ergebnisüberprüfungskomponente und Steuerungskomponente

Die Ergebnisprüfungskomponente bewertet die Erkennung der Transportkiste durch das neuronale Netz, das auf dem Datensatz der Transportkiste trainiert wurde, hinsichtlich der Erkennungsgenauigkeit, die während der Generierung der Eingaben für das Netz durch Layout Guidance in das Foundation-Modell eingebettet wurde. Es wird die Vorhersage des Objektdetektors ermittelt, die die größte Übereinstimmung mit der gegebenen Bounding Box und der gegebenen Klasse aufweist. Aus der ermittelten Vorhersage wird dann die Confidence berechnet.

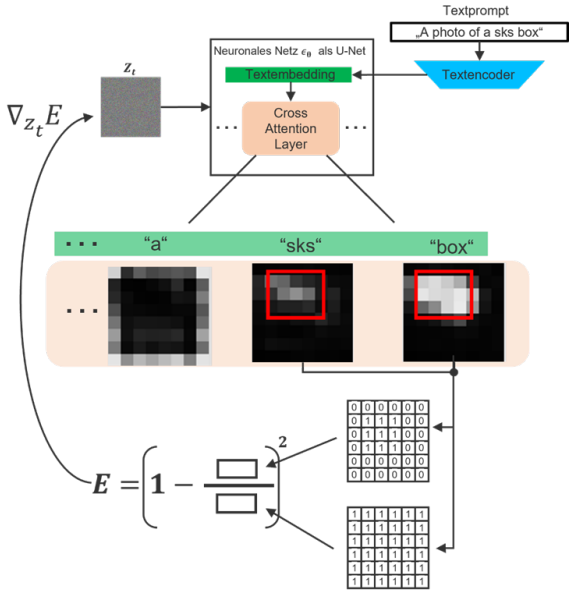

Abbildung 4.25: Backward Guidance der Layout Guidance

Für den Loss gilt schließlich $L(\text{pred}, \text{target}) = 1 - \text{conf}$. Dieser Loss wurde gewählt, damit sich der FGSM-Algorithmus der Steuerungskomponente auf die Transportbox und nicht auf andere Objekte konzentriert.

Die Steuerungskomponente ist wieder mit dem FGSM-Algorithmus umgesetzt, allerdings unterscheidet sich dessen Einsatz von den anderen Anwendungsfällen. Bei den Anwendungsfällen 1 und 2 gab es jeweils nur einen **z**-Vektor, der in FGSM verwendet werden konnte. Stable Diffusion unterscheidet sich in diesem Punkt von den bisher verwendeten generativen Modellen. Es hat einen **z**-Vektor nach dem finalen VAE-Dekoder, der das Bild aus dem latenten Raum synthetisiert und für jeden durchgeführten Diffusionsschritt einen weiteren **z**-Vektor (Abbildung 4.26). Es stellt sich also die Frage, welcher **z**-Vektor durch FGSM angegriffen werden sollte. Für die Anwendung von FGSM muss der Gradient von **z** für jeden Angriff über das gesamte Modell bis zu **z** berechnet werden. Dadurch ergibt sich in der Praxis keine andere Möglichkeit als nur den **z**-Vektor nach dem finalen VAE-Dekoder mit FGSM anzugreifen. Die verwendete Grafikkarte hat 24 GB Grafikspeicher und ist nicht in der Lage die Gradienten für einen der tiefer liegenden **z**-Vektoren zu berechnen.

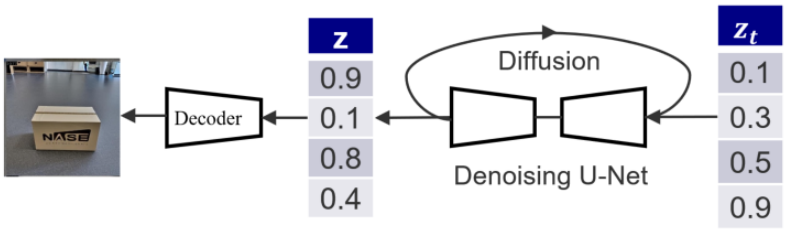

Abbildung 4.26: Vereinfachte Darstellung des verwendeten Foundation-Modells Stable Diffusion mit dem Fokus auf die latenten z-Vektoren.

4.4.3 Evaluierung

In diesem Abschnitt ist die Evaluierung des im vorherigen Kapitel realisierten Konzepts mit dem industriellen Transportkistendatensatz beschrieben. Dabei wird untersucht, ob die Objektdetektionsgenauigkeit in einem industriellen Umfeld mit extrem wenigen vorhandenen realen Daten durch synthetische Bilddaten verbessert werden kann.

4.4.3.1 Objektdetektor

Als Objektdetektionsmodell wird die YOLOv5 Architektur verwendet [133]. YOLOv5 (You Only Look Once Version 5) ist eine fortschrittliche Version der YOLO-Familie von Algorithmen für Objekterkennung, die für ihre Geschwindigkeit und Genauigkeit bekannt ist. YOLOv5 zeichnet sich durch seine Fähigkeit aus, Objekte in Bildern und Videos in Echtzeit zu erkennen und zu klassifizieren. Es verwendet ein einzelnes neuronales Netzwerk, das direkt auf das gesamte Bild angewendet wird, wodurch es die Objekte und deren Klassifikationen in einem Durchgang vorhersagt. Dieser Ansatz ermöglicht eine außerordentlich schnelle Verarbeitungsgeschwindigkeit, was YOLOv5 besonders geeignet für Anwendungen macht, bei denen es auf hohe Leistung und geringe Latenz ankommt, wie etwa in der Videoüberwachung, im autonomen Fahren und in verschiedenen Formen der automatisierten Inspektion. Im Vergleich zu früheren Versionen bietet YOLOv5 Verbesserungen in Bezug auf Architektur, Training und Implementierung, wodurch die Effizienz und Genauigkeit weiter gesteigert werden.

Für das Finetuning des YOLOv5-Modells kommt die gleichnamige Bibliothek zum Einsatz, die ein Trainingsskript beinhaltet. Dieses erlaubt die Festlegung der Epochenanzahl und der zu nutzenden Bildmenge. Die restlichen 10 Bilder des Trainingsdatensatzes dienen als Trainingsmaterial für den Vergleichs-Boxdetektor 1. Nachdem die synthetischen Daten generiert wurden, erfolgt das Finetuning von Boxdetektor 2, wobei die 10 Trainingsbilder und die neu erstellten Bilder verwendet werden. Für das Training der Boxdetektoren sind jeweils 300 Epochen vorgesehen.

4.4.3.2 Qualitative Bewertung der synthetischen Bilddaten

Um die Generierung von Daten zu ermöglichen, die Transportboxen möglichst genau abbilden, ist es erforderlich, dass das Diffusionsmodell die Charakteristika der Transportboxen effektiv abstrahiert. Dies lässt sich beispielsweise an der Genauigkeit erkennen, mit der die generierten Bilder die Vorgaben verschiedener Prompts umsetzen. Ein illustratives Beispiel hierfür wäre „A sks box next to the Eiffel Tower“. Eine hohe Übereinstimmung des generierten Bildes mit dem vorgegebenen Prompt deutet darauf hin, dass das Diffusionsmodell die Eigenschaften der Transportboxen zutreffend abstrahiert hat. Ein weiteres wichtiges Kriterium ist die Effektivität der Layout-Guidance, die durch die Verwendung unterschiedlicher Bounding-Boxen demonstriert wird. Die

Präzision, mit der die Transportboxen in der Nähe der vorgegebenen Positionen generiert werden, spiegelt die Eignung des Diffusionsmodells für die Datengenerierung wider.

Um die Genauigkeit zu bewerten, könnte der CLIP-Score, der die Ähnlichkeit zwischen Bild und Texteingabe misst, herangezogen werden [134]. Eine Herausforderung besteht jedoch darin, dass der Text-Encoder des Diffusionsmodells keine spezifischen Kenntnisse über Transportboxen hat. Folglich kann der CLIP-Score zwar das Vorhandensein von Boxen im Bild bestätigen, jedoch nicht zwischen Transportboxen und anderen Boxarten unterscheiden. Zudem beschränken sich solche Metriken darauf, ausschließlich die im Prompt explizit genannten Aspekte zu bewerten, wodurch der Stil der generierten Bilder außer Acht gelassen wird. Eine manuelle, qualitative Analyse der Bilder kann diese Einschränkungen umgehen, was den Vorzug einer solchen Methode für die Bewertung unterstreicht.

Für die Qualität der Bounding-Boxen könnte die Erstellung manueller Bounding-Boxen für die generierten Bilder in Betracht gezogen werden, um daraufhin den Jaccard-Index (auch bekannt als Intersection over Union oder IoU) zu berechnen. Angesichts der Menge an generierten Bildern erweist sich dieser Ansatz jedoch als zeitintensiv, da für jedes Bild individuelle Bounding-Boxen angelegt werden müssten. Daher wird auch die Bewertung des zweiten Kriteriums durch eine qualitative, manuelle Analyse erfolgen.

Abbildung 4.27: Beispiele der synthetischen Daten, die mit Dreambooth und Layout Guidance durch Stable Diffusion generiert wurden.

Mangels objektiver, automatisiert berechenbarer Qualitätskriterien, wurde auch keine umfängliche Hyperparametersuche durchgeführt. Stattdessen wurden einzelne Experimente durchgeführt, um eine geeignete Lernrate und Anzahl an Trainingsschritten für Dreambooth zu bestimmen. Die Experimente wurden subjektiv durch den Durchführenden bewertet. Die besten Ergebnisse

wurden mit einer Learning Rate von $2 * 10^{-6}$ und 800 Trainingsepochen erreicht. Die Ergebnisse sind in Abbildung 4.27 zu sehen. Die Ergebnisbilder zeigen, dass der Stil der Transportbox in den meisten Bildern gut getroffen wurde. Die Hintergründe sehen sehr realistisch aus und zeigen Umgebungen ähnlich der Arena2036. Auch die verschiedenen Beklebungen auf den Kisten sehen sehr realistisch aus. Die Beschriftung ist fast immer falsch, was bei Stable Diffusion als Foundation-Modell zu erwarten ist. Die Bounding-Boxen sind fast bei keinem Beispiel exakt korrekt, allerdings durchaus meistens näherungsweise korrekt. Oft scheinen sie etwas zu klein zu sein. Zusammenfassend kann der Realismus der generierten Bilder als sehr hoch bewertet werden, wobei die Änderung der Beschriftung das größte Manko darstellt. Die Qualität der Label ist wechselhaft, aber deutlich schlechter als ein manuell erstellter hochwertiger Datensatz.

Die bisher in diesem Anwendungsfall gezeigten Bilder sind ausschließlich Bilder gewesen, die bisher nicht durch die gezielte Steuerung der Generierung angepasst wurden. Im nächsten Abschnitt wird die Qualität der gezielt generierten Daten untersucht. Für die Steuerungskomponente wird der ϵ -Wert auf 0,5 gesetzt und eine Maximalanzahl von 10 Iterationen verwendet. Abbildung 4.28 zeigt sechs Bildpaare. Die linken Bilder sind mit zufälligem z-Vektor generiert. Die rechten Bilder sind die entsprechenden, gezielt generierten Bilder. Die Unterschiede zwischen den linken und rechten Bildern reichen von minimalen Veränderungen der Klebebänder bis zu Perspektivwechseln und dem kompletten Austausch des Hintergrunds. Die Änderungen sind alle für die Domäne plausibel und der Realitätsgrad der Bilder ändert sich durch die gezielte Generierung nicht. Die abgebildeten Beispiele haben auch zu einer Veränderung der Detektionsgenauigkeit des Detektormodells geführt. Es existieren auch Beispiele, bei denen die Steuerungskomponente fehlgeschlagen hat und sich die Genauigkeit des Detektors nicht verändert hat. Diese Beispiele werden durch die Entscheidungskomponente identifiziert und nicht in den synthetischen Trainingsdatensatz hinzugefügt. Im nächsten Kapitel wird das Training mit den synthetischen Daten evaluiert.

Abbildung 4.28: Synthetische Bildpaare. Das jeweils linke Bild zeigt ein Bild, das mit zufälligem z-Vektor generiert wurde. Das rechte Bild zeigt das darauf basierende, durch die Steuerungskomponente angepasste, Bild.

4.4.3.3 Quantitative Bewertung des Objektdetektors durch die Verwendung von synthetischen Trainingsdaten

Um den gezielten Datengenerator quantitativ zu evaluieren, werden, wie in 4.4.3.1 beschrieben, zwei Objektdetektoren trainiert. Ein Detektor wird nur mit den echten Daten trainiert, der andere mit den echten und den gezielt generierten Daten. Die generierten Bilder werden immer an der gleichen Position, also mit der gleichen Bounding Box generiert. Die weiteren Parameter der Datengenerierung sind

- Ab einem Detektionsschwellwert von 0,7 wird eine Box als erkannt gewertet.
- Es werden maximal 10 Iterationen des Generators durchgeführt.
- Das FGSM-Epsilon wurde auf 0,05 gesetzt.
- Es wurden 100 synthetische Bilder generiert.

Für die Evaluierung wurde bei der Generierung neben den Bildern, die den Objektdetektor täuschen, auch immer die originalen Bilder gespeichert. Diese originalen Bilder wurden ohne den Einfluss des FGSM-Algorithmus erzeugt, was einer ziellosen Datengenerierung entspricht. Diese

ziellose Datengenerierung ist rein zufällig. Mit den zufällig generierten Bildern und den realen Bildern wurde ein dritter Objektdetektor trainiert. Für den Vergleich wird die im Bereich der Objektdetektion verbreitete Metrik des mAP gewählt.

Die mittlere durchschnittliche Präzision (mean Average Precision, mAP) ist eine Metrik zur Bewertung der Leistung von Objekterkennungsmodellen, die sowohl die Präzision (den Anteil wahrer positiver Erkennungen im Verhältnis zur Gesamtzahl positiver Vorhersagen) als auch den Recall (den Anteil wahrer positiver Erkennungen im Verhältnis zur Gesamtzahl tatsächlicher positiver Vorhersagen) über verschiedene Schwellenwerte hinweg quantifiziert. Sie ist wichtig, weil sie eine einzige Zahl liefert, die die Gesamteffektivität des Modells bei der Unterscheidung und korrekten Lokalisierung verschiedener Objekte innerhalb eines Bildes erfasst, wobei sowohl die Genauigkeit der Bounding-Boxen als auch die Zuverlässigkeit der Vorhersagen berücksichtigt werden. Die mAP wird berechnet, indem zunächst die durchschnittliche Präzision (AP) für jede Klasse berechnet wird. Dazu wird für jede Klasse eine Präzisions-Recall-Kurve über einen Bereich von Vertrauensschwellenwerten aufgetragen und die Fläche unter dieser Kurve berechnet. Der mAP-Score ist dann der Mittelwert dieser AP-Werte über alle Klassen hinweg und bietet ein ausgewogenes Maß, das sowohl die Präzision als auch den Recall berücksichtigt und somit den Vergleich zwischen verschiedenen Modellen oder Konfigurationen erleichtert.

Die Angabe "mAP50" bezieht sich auf den mAP, der bei einem IoU-Schwellenwert (Intersection over Union) von 0,50 berechnet wird, was bedeutet, dass eine vorhergesagte Bounding-Box als True Positive angesehen wird, wenn sie einen IoU-Wert von 50 % oder mehr mit der Bounding-Box des Labels hat. Dieser Schwellenwert ist relativ fehlertolerant und macht mAP50 zu einem etwas optimistischen Maß für die Modellleistung. Im Gegensatz dazu ist "mAP50:95" eine strengere und umfassendere Metrik, bei der der mAP über mehrere IoU-Schwellenwerte gemittelt wird, in der Regel von 0,50 bis 0,95 in Schritten von 0,05. Diese Metrik liefert eine detailliertere Bewertung der Leistung des Modells über eine Reihe von Strengheitsstufen und bietet eine nuancierte Sicht auf seine Präzisions- und Wiedererkennungsfähigkeiten, insbesondere in Szenarien, die eine hohe Lokalisierungsgenauigkeit erfordern.

Tabelle 4.3 zeigt den Vergleich der drei trainierten Modelle mit dem mAP50-score. Die Ergebnisse zeigen eine sehr starke Verbesserung durch Einsatz von synthetischen Daten. Speziell bei höheren Detektionsschwellwerten (z.B. 0,90) sind die erreichten Verbesserungen sehr groß. Durch die Verwendung der generierten Daten hat sich die Genauigkeit weiter verbessert. Auch bei der Betrachtung der mAP50:95 Metrik (Tabelle 4.4) zeigt sich der gleiche Trend.

Tabelle 4.3: Vergleich der Objektdetektoren mit dem mAP50-Score

Detektionsschwellwert	0.05	0.25	0.40	0.50	0.60	0.70	0.80	0.90	0.95
Reale Daten	0.92	0.92	0.89	0.89	0.86	0.82	0.70	0.51	0.00
Reale Daten + zufällige synthetische Daten	0.94	0.90	0.89	0.89	0.87	0.86	0.83	0.73	0.58
Reale Daten + gezielt generierte Daten	0.96	0.93	0.92	0.92	0.91	0.89	0.86	0.79	0.59

Tabelle 4.4: Vergleich der Objektdetektoren mit dem mAP50:95-Score

Detektionsschwellwert	0.05	0.25	0.40	0.50	0.60	0.70	0.80	0.90	0.95
Reale Daten	0.69	0.71	0.70	0.70	0.69	0.66	0.58	0.48	0.00
Reale Daten + zufällige synthetische Daten	0.69	0.68	0.67	0.67	0.67	0.66	0.65	0.60	0.51
Reale Daten + gezielt generierte Daten	0.74	0.72	0.72	0.72	0.71	0.70	0.69	0.65	0.54

Schlussfolgerung: *Die Evaluierung des entwickelten Konzepts zeigt, dass der Einsatz synthetischer Bilddaten die Genauigkeit der Objektdetektion in einem industriellen Umfeld signifikant verbessert. Das YOLOv5-Modell, das zur Objekterkennung verwendet wird, profitiert deutlich von der zusätzlichen synthetischen Datenmenge, die speziell generiert wurde, um die Charakteristika der Transportboxen abzubilden.*

Im nachfolgenden Kapitel werden die einzelnen gewonnenen Erkenntnisse aus der Realisierung der Anwendungsfälle zusammengetragen und mit den Zielen der Arbeit abgeglichen.

4.5 Diskussion der Evaluierungsergebnisse

Im vorherigen Kapitel wurde die Implementierung des Konzepts eines zielorientierten Trainingsdatengenerators in drei unterschiedlichen Anwendungsszenarien vorgestellt und deren Mehrwert für die jeweiligen Einsatzgebiete bewertet. Dieses Kapitel zielt darauf ab, die Erfüllung der in Kapitel 1.3 formulierten Zielsetzung dieser Arbeit zu evaluieren. Dabei wird untersucht, inwieweit die Anwendungsszenarien im Kontext des Design-Science-Research-Ansatzes demonstrieren, dass die entwickelte Lösung den in Kapitel 1.3 definierten Zielen gerecht wird. Auf Basis dieser Bewertung wird die Einsetzbarkeit der entwickelten Methodik in der Automatisierungstechnik diskutiert.

4.5.1 Bewertung der Evaluierungsergebnisse

Teilziel 1: Automatische Trainingsdatengenerierung

In den Evaluierungsfällen 1 und 2 wird die Generierung von Daten durch den Einsatz von GANs automatisiert. Diese Netzwerke bedürfen einer initialen Trainingsphase, nach deren Abschluss die Datengenerierung vollautomatisch erfolgt. Im Gegensatz zu Methoden, die domänenspezifisches Fachwissen und oft manuelle Anpassungen erfordern – wie beispielsweise Simulationen – ermöglichen GANs eine automatisierte Datengenerierung.

Evaluierungsfall 3 greift auf ein bestehendes Foundation-Modell zurück, das durch spezifische Anpassungen modifiziert wird, um die benötigten Daten ebenfalls automatisiert zu generieren.

Somit erfolgt in allen drei Evaluierungsfällen die Datenerzeugung automatisch, was das Teilziel 1 erfüllt. Wie hoch der initiale Aufwand bei der Erstellung der generativen Modelle ist, ist jedoch von den einzelnen Evaluierungsfällen abhängig.

Teilziel 2: Realitätsnähe der Daten

Die Evaluierungsfälle 1 und 3 basieren auf für Menschen interpretierbaren Bilddaten, was eine direkte Beurteilung des Realitätsgrades der generierten Bilder ermöglicht. Speziell im Evaluierungsfall 1 wurde ein detaillierter Vergleich der durch die FGSM-Methode induzierten Pixelveränderungen vorgenommen. Dabei wurde unterschieden zwischen Veränderungen auf der Pixelebene und im latenten Raum des Generators, wie in dieser Arbeit vorgeschlagen. Es wurde deutlich, dass der Einsatz von FGSM im latenten Raum realistische Modifikationen in den Bildern hervorruft. Ähnlich lassen sich im Evaluierungsfall 3 domänenspezifische und erkennbare Anpassungen in den Bildern feststellen.

Im Gegensatz dazu basiert Evaluierungsfall 2 auf 5G-Empfangsdaten, die für Menschen ohne Weiteres nicht interpretierbar sind. Eine Beurteilung des Realitätsgrades allein durch Betrachtung

dieser Daten ist somit nicht möglich. Neben der qualitativen Analyse erlaubt auch eine quantitative Untersuchung Rückschlüsse auf den Realitätsgrad der generierten Daten, indem deren Einfluss auf die Modellleistung beim Training betrachtet wird. Eine Verschlechterung der Modellleistung bei Verwendung unrealistischer Daten und eine Verbesserung bei Einsatz realistischer Daten weisen darauf hin, dass generierte Daten einen hinreichenden Realitätsgrad besitzen müssen, damit das Modell relevante Erkenntnisse über die tatsächliche Domäne gewinnen kann. Die Verbesserung der Modellgüte durch den Einsatz gezielt generierter Daten wurde für alle drei Evaluierungsfälle nachgewiesen.

Somit erfüllen die Evaluierungsfälle das Teilziel 2, wobei für die Fälle 1 und 3 sowohl qualitative als auch quantitative Nachweise erbracht wurden, während für den Fall 2 ausschließlich quantitative Belege vorliegen.

Teilziel 3: Generierte Trainingsdaten verbessern das Modell

In allen drei Evaluierungsfällen führte die Anwendung der FGSM-Methode auf den latenten **z**-Vektor der jeweiligen Datengeneratoren dazu, dass die erzeugten Daten gezielt Schwachstellen des bestehenden Modells ausnutzen. Dies macht sie besonders relevant für weitere Trainingsprozesse. Diese Ergebnisse bestätigen die Erfüllung des Teilziels 3. Ferner wurde die Leistungssteigerung der Trainingsdaten für die Evaluierungsfälle 1 und 3 detailliert in den entsprechenden Kapiteln analysiert.

Teilziel 4: Übertragbarkeit für verschiedene Datenarten

Das entwickelte Konzept wurde erfolgreich auf drei unterschiedliche Datentypen angewendet: Bilddaten für Klassifikationsaufgaben, 5G-Empfangsdaten für Positionierungsanwendungen und Bilddaten für die Objekterkennung.

Trotz der Vielfalt in den Anwendungsgebieten blieb die grundlegende Architektur der Realisierungen sowie der Einsatz der FGSM-Methode konstant. Notwendige Anpassungen wurden spezifisch für die Implementierung der Datengeneratoren und die Entscheidungslogik je nach Datentyp vorgenommen. Die erfolgreiche Evaluierung über diese verschiedenen Datenarten hinweg bestätigt die Erfüllung des Teilziels 4.

Fazit der Bewertung:

Die Evaluierung des entwickelten Konzepts eines zielorientierten Trainingsdatengenerators über drei diversifizierte Anwendungsszenarien hinweg bekräftigt dessen Eignung, die definierten Zielsetzungen dieser Arbeit zu erfüllen. Dies steht im Einklang mit den Prinzipien des Design Science Research. Durch den Einsatz von Generative Adversarial Networks und modifizierten Foundation-Modellen wird eine automatisierte und realistische Datengenerierung realisiert, die spezifische Schwachstellen bestehender Modelle adressiert und somit für das Training von besonderem Wert ist. Die erfolgreiche Anpassung dieses Konzepts an verschiedene Datentypen – einschließlich Bilddaten für Klassifikation und Objektdetektion sowie 5G-Empfangsdaten für Positionierungsaufgaben – belegt die Übertragbarkeit und Vielseitigkeit der Methode. Diese Ergebnisse bestätigen nicht nur die Effektivität des Ansatzes, sondern auch dessen Potenzial für weiterführende Anwendungen und Forschung im Bereich des maschinellen Lernens in der Automatisierungstechnik.

4.5.2 Anwendbarkeit des Verfahrens in der Automatisierungstechnik

Die Evaluierungsergebnisse zeigen, dass das entwickelte Verfahren zur automatisierten, leistungssteigernden Generierung synthetischer Trainingsdaten die in dieser Arbeit angestrebten Ziele erreicht. Damit stellt die Generierung leistungssteigernder Trainingsdaten einen vielversprechenden Ansatz für den Einsatz in der industriellen Automatisierung dar. In diesem Kontext sind jedoch einige Aspekte zu diskutieren, die sowohl Potenziale als auch Einschränkungen des Verfahrens beleuchten.

Erstens hebt die Möglichkeit zur Automatisierung der Datengenerierung das Verfahren deutlich von herkömmlichen Methoden ab, die in der Regel manuelle Anpassungen und erhebliches domänenspezifisches Fachwissen erfordern. Diese Automatisierung senkt potenziell die Kosten und den Aufwand, die mit der Erstellung und Annotation umfangreicher Datensätze verbunden sind. Dadurch eignet sich das Verfahren besonders für Anwendungen, in denen schnelle und flexible Anpassungen an neue Szenarien oder Änderungen in den Produktionsbedingungen erforderlich sind. Für hochdynamische Produktionsumgebungen kann dies einen entscheidenden Vorteil darstellen.

Ein weiterer bedeutender Aspekt ist die Fähigkeit des Verfahrens, realitätsnahe Daten zu generieren, die das Modell gezielt in Bereichen trainieren, die für die industrielle Anwendung besonders wichtig sind. In Anwendungen, in denen spezifische Szenarien wie seltene Fehlerfälle oder extreme Umgebungsbedingungen berücksichtigt werden müssen, könnte das Verfahren dazu beitragen, die Robustheit und Generalisierungsfähigkeit der Modelle zu erhöhen. So zeigt die Generierung gezielter Datenpunkte im Evaluierungsfall der Objektdetektion, dass das Verfahren imstande ist, spezifische Eigenschaften industrieller Umgebungen abzubilden, wie sie in allgemeinen

Datensätzen nicht vorhanden sind. Dies ermöglicht eine bessere Anpassung an spezielle industrielle Anwendungen, wie z.B. die Erkennung von maschinenspezifischen Teilen oder Werkstücken.

Allerdings bringt das Verfahren auch Herausforderungen mit sich, die für den industriellen Einsatz berücksichtigt werden müssen. Insbesondere die initiale Trainingsphase der verwendeten GANs oder Foundation-Modelle erfordert beträchtliche Rechenressourcen und ist von der Qualität der vorhandenen Ausgangsdaten abhängig. In Umgebungen mit stark begrenzter Datenverfügbarkeit oder spezifischen Anforderungen an Datensicherheit könnte dieser Aspekt einschränkend wirken. Zudem hängt der Erfolg der Datengenerierung von der Verfügbarkeit geeigneter Zielmetriken ab, die für eine leistungssteigernde Generierung notwendig sind. Die Implementierung dieser Metriken erfordert fundiertes Fachwissen und könnte in stark variierenden Automatisierungsszenarien zu zusätzlichen Anpassungsaufwänden führen.

Zusammenfassend zeigt die Evaluierung, dass das entwickelte Verfahren vielversprechende Anwendungen in der industriellen Automatisierung bietet, insbesondere in Szenarien mit hohem Bedarf an flexiblen, gezielt generierten Trainingsdaten. Um jedoch die Übertragbarkeit auf verschiedene industrielle Anwendungsbereiche weiter zu stärken, ist eine Verfeinerung des Verfahrens notwendig, insbesondere im Hinblick auf die Effizienz der Trainingsphase und die Anpassungsfähigkeit an verschiedene Datenumgebungen. Dies könnte durch die Integration weiterer, domänenspezifischer Optimierungsschritte oder durch die Entwicklung effizienterer Trainingsstrategien erreicht werden, die den Einsatz des Verfahrens auch in ressourcenbeschränkten Umgebungen ermöglichen.

5 Schlussbetrachtung

Dieses Kapitel fasst zunächst die Ergebnisse dieser Arbeit zusammen und gibt dann einen Ausblick auf mögliche weitere Forschungsaktivitäten.

5.1 Zusammenfassung der Ergebnisse

Die Akquisition einer adäquaten Menge an gelabelten Trainingsdaten stellt ein signifikantes Problem in der industriellen Automatisierung dar, insbesondere für das Training leistungsfähiger Deep-Learning-Modelle für den Feldeinsatz. Oftmals erfordert die Datensammlung eine Unterbrechung der Produktionsprozesse, was den Betrieb erheblich stören kann. Zudem herrscht häufig Unklarheit darüber, welche Daten zur Verbesserung der Modellleistung tatsächlich benötigt werden. Eine weitere Schwierigkeit ist der kostspielige und meist manuelle Prozess des Daten-Labelings durch menschliche Mitarbeiter.

Basierend auf diesen Herausforderungen wurden spezifische Ziele für ein Konzept formuliert, das die Frage adressiert, wie gelabelte Datensätze effizient und kosteneffektiv generiert werden können. Dieses Konzept muss die automatische Generierung von Trainingsdaten ermöglichen, um logistische und finanzielle Belastungen zu reduzieren und den Modellentwicklungsprozess zu beschleunigen. Es ist essenziell, dass die generierten Daten die realen Betriebsbedingungen präzise widerspiegeln, um eine hohe Modellgenauigkeit im praktischen Einsatz sicherzustellen. Ferner ist eine leistungssteigernde Datengenerierung erforderlich, die speziell darauf ausgerichtet ist, das Modell effektiv zu verbessern und dabei das Risiko von Overfitting durch irrelevante Daten zu minimieren. Schließlich müssen die Methoden zur Datengenerierung eine hohe Flexibilität und Anpassungsfähigkeit aufweisen, um die diversen Datentypen und Einsatzszenarien, die in der Automatisierungstechnik benötigt werden, abdecken zu können.

Die Untersuchung des Stands der Wissenschaft ergab, dass es zwar viele Methoden gibt, die es prinzipiell ermöglichen, synthetische Trainingsdaten zu generieren, allerdings existieren keine Ansätze, die dies gezielt leistungssteigernd ermöglichen und zugleich auf verschiedene Datentypen und Problemformulierungen übertragbar sind.

Das in dieser Arbeit vorgeschlagene neuartige Konzept eines iterativen Datengenerators, der speziell für das leistungssteigernde Training von neuronalen Netzwerken konzipiert ist, adressiert die zuvor beschriebenen Problemstellungen direkt. Dies bedeutet, dass die generierten Daten einen erhöhten Nutzen für das Training aufweisen, indem sie gezielt komplexe Szenarien enthalten, die für das Modell herausfordernder sind als zufällig generierte Daten.

Für die Implementierung des Datengenerators sind zwei primäre Voraussetzungen notwendig: Erstens, eine ausreichende Menge an qualitativ hochwertigen, korrekt gelabelten echten Daten,

die groß genug sein muss, um ein conditional generatives neuronales Netzwerk zu trainieren. Zweitens, ein bereits existierendes, vortrainiertes neuronales Netzwerk, dessen Qualität die Effektivität der generierten Daten beeinflusst, weil die generierten Daten passend zu diesem Netzwerk synthetisiert werden. Der Generator selbst besteht aus drei Hauptmodulen: dem Generatormodul, das die Daten synthetisiert, dem Steuerungsmodul, das auf Basis der Netzwerkausgaben neue Generierungsparameter setzt, und dem Ergebnisprüfmodul, das die Qualität der Netzwerkausgaben überprüft und die Parameteranpassungen steuert. Diese Module arbeiten in einem zyklischen und sequenziellen Prozess zusammen, um kontinuierlich Daten zu generieren, die das neuronale Netzwerk effektiv trainieren.

Das vorgeschlagene Konzept wurde an drei unterschiedlichen Anwendungsfällen, die für die Automatisierungstechnik wichtig sind, umgesetzt und evaluiert.

Der erste Anwendungsfall ist die Handschrifterkennung mit Hilfe des MNIST-Datensatzes. Es wurde gezeigt, dass die implementierte Methodik in der Lage ist, gezielt realistische Bilddaten zu erzeugen und die Klassifikationsgenauigkeit eines Modells zu verbessern, wenn es mit diesen Daten trainiert wird. Konkret wurde die Klassifizierung um 17 % verbessert. Im Rahmen dieses Anwendungsfalls konnte demonstriert werden, dass der Generator in der Lage ist, selbstständig realistische neue Bilddaten zu generieren, die für einen menschlichen Beobachter erklärbar herausfordernd sind.

Der zweite Anwendungsfall ist die datengetriebene 5G-Positionierung, bei der ein trainiertes Modell einen 5G-Sender anhand des bei den Empfängern angekommenen Signals möglichst genau positioniert. Der reale Datensatz stammt aus dem industriellen Umfeld der Arena2036. Dabei wurde der Generator als Conditional Wasserstein GAN realisiert, das wie auch das GAN des ersten Anwendungsfalls im latenten Raum über den FGSM-Algorithmus angesteuert wird. Die Ergebnisse zeigen, dass die Genauigkeit und die Generalisierung des 5G-Positionierungssystems durch die Verwendung der generierten Daten verbessert werden konnten. Eine Verbesserung des Medians der Positionierungsgenauigkeit um 25 % konnte erreicht werden.

Bei dem dritten Anwendungsfall handelt es sich um die optische Objektdetektion von Objekten in einer Industrieumgebung, die nicht in verfügbaren Datensätzen vorkommen. Der Generator wurde durch die Anpassung eines frei verfügbaren Diffusionsmodells realisiert und zeigt, dass der Ansatz auch auf ganz unterschiedliche Arten von generativen Modellen übertragbar ist. Der mAP50-Score des Bilddetektors konnte durch den Einsatz der synthetischen Daten bei einer Entscheidungsschwelle von 0,8 um 0,16 angehoben werden, was die praktische Einsetzbarkeit des Detektors beträchtlich erhöht. Auch in diesem Anwendungsfall zeigte sich, dass der Generator eigenständig herausfordernde Bildvariationen identifizieren kann. So erkannte er beispielsweise, dass bestimmte Klebebandkonstellationen auf der Kiste den bestehenden Detektor überfordern, und erzeugte daraufhin gezielt vermehrt entsprechende Variationen.

Zusammenfassend bietet der in dieser Arbeit vorgestellte Datengenerator einen vielversprechenden Lösungsansatz für das Problem der Datenbeschaffung in der industriellen Automatisierung. Durch die gezielt leistungssteigernde und effiziente Generierung von Trainingsdaten können Deep-Learning-Modelle schneller und mit höherer Genauigkeit entwickelt werden, was den praktischen Einsatz dieser Technologien erheblich fördert. Der Ansatz zeigt auch die Bedeutung der Initialqualität von Trainingsdaten und vortrainierten Netzwerken, die für den Erfolg der generierten Daten entscheidend sind.

5.2 Ausblick auf weiterführende Forschungsaktivitäten

Zukünftige Forschungsaktivitäten im Bereich der automatisierten Datengenerierung für industrielle Automatisierungssysteme sollten sich auf die Weiterentwicklung von Methoden konzentrieren, die sowohl die Effizienz der Modelltrainingsprozesse verbessern als auch deren Anwendbarkeit in verschiedenen industriellen Szenarien erweitern. Die folgenden Forschungsansätze bieten vielversprechende Möglichkeiten zur weiteren Optimierung und Praxisintegration:

Eine wesentliche Grundlage zukünftiger Arbeiten könnte in der Entwicklung objektiver Metriken zur Bewertung der Realitätsnähe synthetischer Daten liegen. Derzeit verfügbare Metriken sind oft spezifisch auf Bilddaten beschränkt, was die Übertragbarkeit auf andere Datenarten, die in der Automatisierungstechnik relevant sind, erschwert. Ein umfassender Ansatz zur Bewertung der Realitätsnähe könnte die Anwendbarkeit synthetischer Daten erheblich erweitern, indem realitätsnahe Datenquellen für verschiedene industrielle Anwendungsfälle geschaffen werden. Diese Metriken würden nicht nur die Qualität der generierten Daten evaluieren, sondern könnten auch langfristig die Robustheit und Effizienz industrieller KI-Modelle sicherstellen.

Darüber hinaus bietet die Integration von Domänenwissen in den Generierungsprozess Potenzial, um die Qualität und Praxistauglichkeit der Daten weiter zu erhöhen. Obwohl das vorgestellte Verfahren ohne explizites Domänenwissen auskommt, könnte die gezielte Einbindung von Fachwissen die Relevanz und Genauigkeit der generierten Daten erhöhen. Forschungen in diesem Bereich könnten untersuchen, wie sich Domänenwissen systematisch und skalierbar in die Generierung von Daten integrieren lässt, um spezifische Herausforderungen industrieller Anwendungen zu adressieren und die Generalisierungsfähigkeit der Modelle zu fördern.

Eine interessante Perspektive ergibt sich aus der automatisierten Auswahl und dem Training generativer Methoden, insbesondere durch den Einsatz und die Adaption von Foundation-Modellen. Diese vortrainierten Modelle, wie bereits im dritten Anwendungsfall demonstriert, bieten vielversprechende Möglichkeiten für die flexible und effiziente Datenerzeugung. Zukünftige Arbeiten könnten untersuchen, wie ein modularer Ansatz in Form eines Modell-Hubs verschiedene generative Methoden basierend auf den spezifischen Anforderungen und Datenarten eines Projekts automatisch auswählt und anpasst. Die wachsende Verfügbarkeit vortrainierter Modelle aus verschiedenen Anwendungsbereichen bietet hier ein großes Potenzial für skalierbare und anpassbare Lösungen.

Langfristig könnte die Entwicklung eines Trainingsdatengenerierungsframeworks einen wertvollen Beitrag zur Praxisintegration leisten. Nachdem objektive Metriken zur Realitätsbewertung und optimierte generative Methoden etabliert sind, könnte ein Low-Code- oder No-Code-Framework die Anwendung und Erzeugung synthetischer Daten vereinfachen. Ein solches Framework würde es Anwendern in der Industrie ermöglichen, Trainingsdatensätze effizient zu generieren und zu

verwalten, ohne umfassende Programmierkenntnisse zu benötigen. Die Einführung eines solchen Tools könnte die Hürden für den Einsatz synthetischer Daten in der Automatisierungstechnik senken und die Reichweite innovativer ML-Methoden in industriellen Anwendungen erweitern. In diesem Zuge würde sich auch eine Integration in den digitalen Zwilling anbieten, für den sich aktuell Methoden zur Klassifizierung der Relevanz von externen Datenquellen in der Entwicklung befinden [135]. Die Kombination aus relevanten externen Datenquellen mit dem Ansatz dieser Arbeit könnte wertvolle Synergien für den Praxiseinsatz hervorbringen.

Durch diese Forschungsansätze könnten nicht nur technologische Fortschritte erzielt werden, sondern auch praxisorientierte Methoden geschaffen werden, die eine nahtlose Integration und Anwendung synthetischer Daten in industriellen Umgebungen fördern. Dies könnte letztlich die Entwicklung effizienterer und anpassungsfähigerer Deep-Learning-Modelle in der Automatisierungstechnik unterstützen und den Weg für neue Innovationen in der industriellen KI-Nutzung ebnen.

6 Literaturverzeichnis

[1] M. Weyrich, *Industrielle Automatisierungs- und Informationstechnik*. Berlin, Heidelberg: Springer Berlin, 2023. doi: 10.1007/978-3-662-56355-7.

[2] S. E. Whang, Y. Roh, H. Song, and J.-G. Lee, "Data collection and quality challenges in deep learning: a data-centric AI perspective," *The VLDB Journal*, vol. 32, no. 4, pp. 791–813, 2023, doi: 10.1007/s00778-022-00775-9.

[3] Y. Roh, G. Heo, and S. E. Whang, "A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1328–1347, 2021, doi: 10.1109/TKDE.2019.2946162.

[4] Daochen Zha, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, and Xia Hu, "Data-centric AI: Perspectives and Challenges," in *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, pp. 945–948. doi: 10.1137/1.9781611977653.ch106.

[5] N. Gupta *et al.*, *Data Quality Toolkit: Automatic assessment of data quality and remediation for machine learning datasets*. 2021.

[6] M. Shojafar, M. Mukherjee, V. Piuri, and J. Abawajy, "Guest Editorial: Security and Privacy of Federated Learning Solutions for Industrial IoT Applications," *IEEE Trans. Ind. Inf.*, vol. 18, no. 5, pp. 3519–3521, 2022, doi: 10.1109/tii.2021.3128972.

[7] Y. Jiang, B. Drescher, and G. Yuan, "A GAN-based multi-sensor data augmentation technique for CNC machine tool wear prediction," *IEEE Access*, 2023.

[8] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?" Explaining the predictions of any classifier," presented at the Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1135–1144.

[9] Y. Lu *et al.*, *Machine Learning for Synthetic Data Generation: A Review*. 2024.

[10] M. Wagenstetter *et al.*, "Synthetically Generated Images for Industrial Anomaly Detection," presented at the 2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, 2024, pp. 1–8. doi: 10.1109/etfa61755.2024.10710816.

[11] H. Vietz, A. Löcklin, H. B. H. Ammar, and M. Weyrich, "Deep learning-based 5G indoor positioning in a manufacturing environment," presented at the 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), 2022, pp. 1–4.

[12] N. Gupta *et al.*, "Data quality for machine learning tasks," presented at the Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, 2021, pp. 4040–4041.

[13] M. Hamood, A. Albaseer, M. Abdallah, and A. Al-Fuqaha, "Clustered and multi-tasked federated distillation for heterogeneous and resource constrained industrial IoT applications," *IEEE Internet of Things Magazine*, vol. 6, no. 2, pp. 64–69, 2023.

[14] S. Kaada, S. A. Hamideche, C. Daems, and M. L. Alberi Morel, "Classification with Synthetic Radio Data for Real-life Environment Sensing," presented at the 2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring), 2023, pp. 1–7. doi: 10.1109/VTC2023-Spring57618.2023.10200643.

[15] M. Dix *et al.*, "Measuring the Robustness of ML Models Against Data Quality Issues in Industrial Time Series Data," presented at the 2023 IEEE 21st International Conference on Industrial Informatics (INDIN), IEEE, 2023, pp. 1–8. doi: 10.1109/indin51400.2023.10218129.

[16] L. Li and M. Spratling, *Data Augmentation Alone Can Improve Adversarial Training*. 2023.

[17] M. Cogswell, F. Ahmed, R. Girshick, L. Zitnick, and D. Batra, *Reducing Overfitting in Deep Networks by Decorrelating Representations*. 2016.

[18] X. Ying, "An Overview of Overfitting and its Solutions," *J. Phys.: Conf. Ser.*, vol. 1168, p. 022022, 2019, doi: 10.1088/1742-6596/1168/2/022022.

[19] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS quarterly*, pp. 75–105, 2004.

[20] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007, doi: 10.2753/MIS0742-1222240302.

[21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*. MIT Press, 2016.

[23] Alhassan Mumuni and Fuseini Mumuni, "Data augmentation: A comprehensive survey of modern approaches," *Array*, vol. 16, p. 100258, 2022, doi: 10.1016/j.array.2022.100258.

[24] Kiran Maharana, Surajit Mondal, and Bhushankumar Nemade, "A review: Data pre-processing and data augmentation techniques," *Global Transitions Proceedings*, vol. 3, no. 1, pp. 91–99, 2022, doi: 10.1016/j.gltp.2022.04.020.

[25] S.-A. Rebuffi, S. Gowal, D. A. Calian, F. Stimberg, O. Wiles, and T. A. Mann, "Data Augmentation Can Improve Robustness," presented at the Advances in Neural Information Processing Systems, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, Eds., Curran Associates, Inc, 2021, pp. 29935–29948. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/fb4c48608ce8825b558ccf07169a3421-Paper.pdf

[26] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019, doi: 10.1186/s40537-019-0197-0.

[27] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc, 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[28] C.-L. Li, K. Sohn, J. Yoon, and T. Pfister, "Cutpaste: Self-supervised learning for anomaly detection and localization," presented at the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 9664–9674.

[29] L. Zhang, S. Huang, and W. Liu, "Intra-class part swapping for fine-grained image classification," presented at the Proceedings of the IEEE/CVF winter conference on applications of computer vision, 2021, pp. 3209–3218.

[30] J. Choi, Y. Song, and N. Kwak, "Part-aware data augmentation for 3d object detection in point cloud," presented at the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 3391–3397.

[31] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," presented at the Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 6023–6032.

[32] J. Yoo, N. Ahn, and K.-A. Sohn, "Rethinking data augmentation for image super-resolution: A comprehensive analysis and a new strategy," presented at the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 8375–8384.

[33] J.-H. Kim, W. Choo, and H. O. Song, "Puzzle mix: Exploiting saliency and local statistics for optimal mixup," presented at the International Conference on Machine Learning, 2020, pp. 5275–5285.

[34] A. F. M. Uddin, M. Monira, W. Shin, T. Chung, S.-H. Bae, and others, "Saliencymix: A saliency guided data augmentation strategy for better regularization," *arXiv preprint arXiv:2006.01791*, 2020.

[35] A. Dabouei, S. Soleymani, F. Taherkhani, and N. M. Nasrabadi, “Supermix: Supervising the mixing data augmentation,” presented at the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 13794–13803.

[36] G. Kim, D. K. Han, and H. Ko, “Specmix: A mixed sample data augmentation method for training with time-frequency domain features,” *arXiv preprint arXiv:2108.03020*, 2021.

[37] H. Vietz, T. Rauch, A. Löcklin, N. Jazdi, and M. Weyrich, “A Methodology to Identify Cognition Gaps in Visual Recognition Applications Based on Convolutional Neural Networks,” presented at the 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), 2021, pp. 2045–2050.

[38] B.-B. Jia and M.-L. Zhang, “Multi-dimensional classification via kNN feature augmentation,” *Pattern Recognition*, vol. 106, p. 107423, 2020.

[39] X. Gastaldi, “Shake-shake regularization,” *arXiv preprint arXiv:1705.07485*, 2017.

[40] Y. Li and F. Liu, “Whiteout: Gaussian adaptive noise regularization in deep neural networks,” *arXiv preprint arXiv:1612.01490*, 2016.

[41] H. Li, X. Zhang, Q. Tian, and H. Xiong, “Attribute mix: Semantic data augmentation for fine grained recognition,” presented at the 2020 IEEE International Conference on Visual Communications and Image Processing (VCIP), 2020, pp. 243–246.

[42] M. Faramarzi, M. Amini, A. Badrinaaraayanan, V. Verma, and S. Chandar, “Patchup: A feature-space block-level regularization technique for convolutional neural networks,” presented at the Proceedings of the AAAI Conference on Artificial Intelligence, 2022, pp. 589–597.

[43] X. Li, Y. Dai, Y. Ge, J. Liu, Y. Shan, and L.-Y. Duan, “Uncertainty modeling for out-of-distribution generalization,” *arXiv preprint arXiv:2202.03958*, 2022.

[44] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy, “Explaining and Harnessing Adversarial Examples,” presented at the International Conference on Learning Representations, 2015.

[45] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards Deep Learning Models Resistant to Adversarial Attacks,” 2017.

[46] A. Krizhevsky, G. Hinton, and others, “Learning multiple layers of features from tiny images,” 2009.

[47] Alexey Kurakin *et al.*, “Adversarial Attacks and Defences Competition,” 2018.

[48] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry, “Robustness May Be at Odds with Accuracy,” 2019.

[49] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. E. Ghaoui, and M. Jordan, “Theoretically Principled Trade-off between Robustness and Accuracy,” presented at the Proceedings of the 36th International Conference on Machine Learning, K. Chaudhuri and R. Salakhutdinov, Eds., in Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 7472–7482. [Online]. Available: https://proceedings.mlr.press/v97/zhang19p.html

[50] C. Harle, M. Barth, and A. Fay, “Operation-parallel adaptation of a co-simulation for discrete manufacturing plants,” *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8. doi: 10.1109/ETFA45728.2021.9613719.

[51] L. Bruckner, M. Oppelt, L. Urbas, and M. Barth, “The current and future use of simulation in discrete and process industries,” *Siemens AG*, 2020.

[52] M. Oppelt, L. Bruckner, R. Rosen, M. Barth, L. Urbas, and J. Jäkel, “Die aktuelle und zukünftige Nutzung von Simulation,” *atp*, vol. 63, no. 04, pp. 64–73, 2021, doi: 10.17560/atp.v63i04.2533.

[53] S. I. Nikolenko, *Synthetic data for deep learning*, vol. 174. Springer, 2021.

[54] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference*. doi: 10.1109/iros.2004.1389727.

[55] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” presented at the 2012 IEEE/RSJ international conference on intelligent robots and systems, 2012, pp. 5026–5033.

[56] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “Airsim: High-fidelity visual and physical simulation for autonomous vehicles,” presented at the Field and Service Robotics: Results of the 11th International Conference, 2018, pp. 621–635.

[57] P. Solovev *et al.*, “Learning State Representations in Complex Systems with Multimodal Data,” Nov. 2018, [Online]. Available: http://arxiv.org/pdf/1811.11067v3

[58] L. Chen, W. Wang, and J. Zhu, “Learning Transferable UAV for Forest Visual Perception,” p. 7, Oct. 2018.

[59] E. Bondi *et al.*, “Airsim-w: A simulation environment for wildlife conservation with uavs,” presented at the Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies, 2018, pp. 1–12.

[60] I. Ullah *et al.*, “A virtual testbed for critical incident investigation with autonomous remote aerial vehicle surveying, artificial intelligence, and decision support,” presented at the ECML PKDD 2018 Workshops: Nemesis 2018, UrbReas 2018, SoGood 2018, IWAISe 2018, and Green Data Mining 2018, Dublin, Ireland, September 10-14, 2018, Proceedings 18, 2019, pp. 216–221.

[61] D. L. Smyth *et al.*, “A Virtual Environment with Multi-Robot Navigation, Analytics, and Decision Support for Critical Incident Investigation,” Dec. 2018, [Online]. Available: http://arxiv.org/pdf/1806.04497v1

[62] B. Huang, D. Bayazit, D. Ullman, N. Gopalan, and S. Tellex, “Flight, camera, action! using natural language and mixed reality to control a drone,” presented at the 2019 International conference on robotics and automation (ICRA), 2019, pp. 6949–6956.

[63] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” presented at the Conference on robot learning, 2017, pp. 1–16.

[64] 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?,” p. 8, Jun. 2016.

[65] Craig Quiter and Maik Ernst, *deepdrive/deepdrive: 2.0*. (2018). Zenodo. doi: 10.5281/ZENODO.1248997.

[66] K.-T. Lai, C.-C. Lin, C.-Y. Kang, M.-E. Liao, and M.-S. Chen, “Vivid: Virtual environment for visual deep learning,” presented at the Proceedings of the 26th ACM international conference on Multimedia, 2018, pp. 1356–1359.

[67] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, “Torcs, the open racing car simulator,” *Software available at http://torcs. sourceforge. net*, vol. 4, no. 6, p. 2, 2000.

[68] G. Rong *et al.*, “LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving,” *arXiv preprint arXiv:2005.03778*, 2020.

[69] X. Yue, B. Wu, S. A. Seshia, K. Keutzer, and A. L. Sangiovanni-Vincentelli, “A lidar point cloud generator: from a virtual world to autonomous driving,” presented at the Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval, 2018, pp. 458–464.

[70] B. Hurl, K. Czarnecki, and S. Waslander, “Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception,” presented at the 2019 IEEE Intelligent Vehicles Symposium (IV), 2019, pp. 2522–2529.

[71] A. Handa, V. Patraucean, S. Stent, and R. Cipolla, “SceneNet: An annotated model generator for indoor scene understanding,” presented at the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden: IEEE, 2016, pp. 5737–5743. doi: 10.1109/ICRA.2016.7487797.

[72] K. Mo *et al.*, “PartNet: A Large-scale Benchmark for Fine-grained and Hierarchical Part-level 3D Object Understanding,” Jun. 2018, [Online]. Available: http://arxiv.org/pdf/1812.02713v1

[73] H. Bertiche, M. Madadi, and S. Escalera, “CLOTH3D: Clothed 3D Humans,” May 2019, [Online]. Available: http://arxiv.org/pdf/1912.02792v2

[74] C. M. Melo, A. Torralba, L. Guibas, J. DiCarlo, R. Chellappa, and J. Hodgins, “Next-generation deep learning based on simulators and synthetic data,” *Trends in cognitive sciences*, vol. 26, no. 2, pp. 174–187, 2022, doi: 10.1016/j.tics.2021.11.008.

[75] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, “VirtualWorlds as Proxy for Multi-object Tracking Analysis,” presented at the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA: IEEE, 2016, pp. 4340–4349. doi: 10.1109/CVPR.2016.470.

[76] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, “Sim4CV: A Photo-Realistic Simulator for Computer Vision Applications,” *Int J Comput Vis*, vol. 126, no. 9, pp. 902–919, 2018, doi: 10.1007/s11263-018-1073-7.

[77] Y. Movshovitz-Attias, T. Kanade, and Y. Sheikh, “How useful is photo-realistic rendering for visual learning?,” Mar. 2016, [Online]. Available: http://arxiv.org/pdf/1603.08152v2

[78] 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,” presented at the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA: IEEE, 2016, pp. 3234–3243. doi: 10.1109/CVPR.2016.352.

[79] A. Shafaei, J. Little, and M. Schmidt, “Play and Learn: Using Video Games to Train Computer Vision Models,” presented at the Proceedings of the British Machine Vision Conference 2016, R. C. Wilson, E. R. Hancock, W. A. P. Smith, N. E. Pears, and A. G. Bors, Eds., York, UK: British Machine Vision Association, 2016, p. 26.1-26.13. doi: 10.5244/C.30.26.

[80] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, “Playing for Data: Ground Truth from Computer Games,” in *Computer Vision – ECCV 2016*, vol. 9906, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., in Lecture Notes in Computer Science, vol. 9906. , Cham: Springer International Publishing, 2016, pp. 102–118. doi: 10.1007/978-3-319-46475-6_7.

[81] J. Tremblay *et al.*, “Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization,” presented at the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW 2018. Salt Lake City, Utah, USA, 18-22 June 2018, Salt Lake City, UT: IEEE, 2018, pp. 1082–10828. doi: 10.1109/CVPRW.2018.00143.

[82] A. Prakash *et al.*, “Structured Domain Randomization: Bridging the Reality Gap by Context-Aware Synthetic Data,” presented at the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada: IEEE, 2019, pp. 7249–7255. doi: 10.1109/ICRA.2019.8794443.

[83] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, “Augmented Reality Meets Computer Vision,” *Efficient Data Generation for Urban Driving Scenes*, 2017.

[84] H. Vietz, T. Rauch, and M. Weyrich, “Synthetic Training Data Generation for Convolutional Neural Networks in Vision Applications,” presented at the 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), 2022, pp. 1–6. doi: 10.1109/ETFA52439.2022.9921534.

[85] Y. Zhang *et al.*, “Physically-Based Rendering for Indoor Scene Understanding Using Convolutional Neural Networks,” Dec. 2016, [Online]. Available: http://arxiv.org/pdf/1612.07429

[86] M. Wrenninge and J. Unger, “Synscapes: A Photorealistic Synthetic Dataset for Street Scene Parsing,” Oct. 2018, [Online]. Available: http://arxiv.org/pdf/1810.08705

[87] A. Tsirikoglou, J. Kronander, M. Wrenninge, and J. Unger, *Procedural Modeling and Physically Based Rendering for Synthetic Data Generation in Automotive Applications*. 2017.

[88] B. Planche *et al.*, “DepthSynth: Real-Time Realistic Synthetic Data Generation from CAD Models for 2.5D Recognition,” presented at the 2017 International Conference on 3D Vision (3DV), Qingdao: IEEE, 2017, pp. 1–10. doi: 10.1109/3DV.2017.00011.

[89] H. Guosheng, P. Xiaojiang, Y. Yongxin, T. M. Hospedales, and J. Verbeek, “Frankenstein: Learning Deep Face Representations Using Small Data,” *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, vol. 27, no. 1, pp. 293–303, 2018, doi: 10.1109/TIP.2017.2756450.

[90] H. N. Qureshi *et al.*, "Toward Addressing Training Data Scarcity Challenge in Emerging Radio Access Networks: A Survey and Framework," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 3, pp. 1954–1990, 2023, doi: 10.1109/COMST.2023.3271419.

[91] C. Bouras, A. Gkamas, G. Diles, and Z. Andreas, "A comparative study of 4G and 5G network simulators," *International Journal on Advances in Networks and Services*, vol. 13, no. 1, 2020.

[92] P. K. Gkonis, P. T. Trakadas, and D. I. Kaklamani, "A Comprehensive Study on Simulation Techniques for 5G Networks: State of the Art Results, Analysis, and Future Challenges," *Electronics*, vol. 9, no. 3, p. 468, 2020, doi: 10.3390/electronics9030468.

[93] U. Masood, H. Farooq, A. Imran, and A. Abu-Dayya, "Interpretable AI-Based Large-Scale 3D Pathloss Prediction Model for Enabling Emerging Self-Driving Networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 7, pp. 3967–3984, 2023, doi: 10.1109/TMC.2022.3147191.

[94] D. Foster, *Generative deep learning*. O'Reilly Media, Inc, 2022.

[95] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," Dec. 2013, [Online]. Available: http://arxiv.org/pdf/1312.6114v11

[96] I. Goodfellow *et al.*, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[97] A. Vaswani *et al.*, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[98] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," presented at the International conference on machine learning, 2015, pp. 2256–2265.

[99] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," Nov. 2015, [Online]. Available: http://arxiv.org/pdf/1511.06434v2

[100] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," presented at the International conference on machine learning, 2017, pp. 214–223.

[101] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.

[102] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," Jun. 2014, [Online]. Available: http://arxiv.org/pdf/1411.1784v1

[103] K. Aggarwal, M. Kirchmeyer, P. Yadav, S. S. Keerthi, and P. Gallinari, "Benchmarking Regression Methods: A comparison with CGAN," May 2019, [Online]. Available: http://arxiv.org/pdf/1905.12868v5

[104] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," presented at the International conference on machine learning, 2016, pp. 1060–1069.

[105] T. Miyato and M. Koyama, "cGANs with Projection Discriminator," Feb. 2018, [Online]. Available: http://arxiv.org/pdf/1802.05637v2

[106] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez, "Invertible Conditional GANs for image editing," Nov. 2016, [Online]. Available: http://arxiv.org/pdf/1611.06355v1

[107] L. Yang, "Conditional generative adversarial networks (cgan) for abnormal vibration of aero engine analysis," presented at the 2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT, 2020, pp. 724–728.

[108] M. Patel, X. Wang, and S. Mao, "Data augmentation with conditional GAN for automatic modulation classification," presented at the Proceedings of the 2nd ACM Workshop on wireless security and machine learning, 2020, pp. 31–36.

[109] E. Strelcenia and S. Prakoonwit, "GAN-based data augmentation for credit card fraud detection," presented at the 2022 IEEE International Conference on Big Data (Big Data), 2022, pp. 6812–6814.

[110] S. Niu, B. Li, X. Wang, and H. Lin, "Defect Image Sample Generation With GAN for Improving Defect Recognition," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1611–1622, 2020, doi: 10.1109/TASE.2020.2967415.

[111] Omar De Mitri, Andreas Frommknecht, Marco F. Huber, Felix Müller-Graf, and Cosimo Distante, "Synthetic data generation for improvement of machine learning-based optical quality control: a practical approach in the welding context," presented at the Multimodal Sensing and Artificial Intelligence: Technologies and Applications III, Ettore Stella, Francesco Soldovieri, Dariusz Ceglarek, and Qian Kemao, Eds., SPIE, 2023, p. 126210Y. doi: 10.1117/12.2682047.

[112] Y. Zhang *et al.*, "Image GANs meet Differentiable Rendering for Inverse Graphics and Interpretable 3D Neural Rendering," Oct. 2020, [Online]. Available: http://arxiv.org/pdf/2010.09125v2

[113] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," presented at the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 4401–4410.

[114] Y. Zhang *et al.*, "Datasetgan: Efficient labeled data factory with minimal human effort," presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 10145–10155.

[115] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," presented at the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005, pp. 886–893 vol. 1. doi: 10.1109/CVPR.2005.177.

[116] L. H. Hughes, M. Schmitt, and X. X. Zhu, "Mining Hard Negative Samples for SAR-Optical Image Matching Using Generative Adversarial Networks," *Remote Sensing*, vol. 10, no. 10, 2018, doi: 10.3390/rs10101552.

[117] Zhengli Zhao, Dheeru Dua, and Sameer Singh, "Generating Natural Adversarial Examples," 2018.

[118] S. Gowal, S.-A. Rebuffi, O. Wiles, F. Stimberg, D. A. Calian, and T. A. Mann, "Improving Robustness using Generated Data," presented at the Advances in Neural Information Processing Systems, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, Eds., Curran Associates, Inc, 2021, pp. 4218–4233. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/21ca6d0cf2f25c4dbb35d8dc0b679c3f-Paper.pdf

[119] Zekai Wang, Tianyu Pang, Chao Du, Min Lin, Weiwei Liu, and Shuicheng Yan, "Better Diffusion Models Further Improve Adversarial Training," 2023.

[120] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 26565–26577, 2022.

[121] R. Rade and S.-M. Moosavi-Dezfooli, "Helper-based adversarial training: Reducing excessive margin to achieve a better accuracy vs. robustness trade-off," presented at the ICML 2021 Workshop on Adversarial Machine Learning, 2021.

[122] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[123] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[124] D. Braun, W. Schloegl, and M. Weyrich, "Automated data-driven creation of the Digital Twin of a brownfield plant," presented at the 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2021, pp. 1–7.

[125] A. Löcklin *et al.*, "Tailored digitization with real-time locating systems: Ultra-wideband RTLS for production and logistics," *atp*, vol. 63, no. 03, pp. 76–83, 2021.

[126] D. Mourtzis, J. Angelopoulos, and N. Panopoulos, "Smart Manufacturing and Tactile Internet Based on 5G in Industry 4.0: Challenges, Applications and New Trends," *Electronics*, vol. 10, no. 24, 2021, doi: 10.3390/electronics10243175.

[127] B. Maschler, H. Vietz, H. Tercan, C. Bitter, T. Meisen, and M. Weyrich, “Insights and Example Use Cases on Industrial Transfer Learning,” *Procedia CIRP*, vol. 107, pp. 511–516, 2022.

[128] H. Vietz, H. B. H. Ammar, S. Baum, N. Jazdi, and M. Weyrich, “5G Indoor Positioning for Manufacturing using Convolutional Neural Networks,” *Procedia CIRP*, vol. 120, pp. 1191–1196, 2023.

[129] J. Choi and R. W. Heath, “Interpolation based transmit beamforming for MIMO-OFDM with limited feedback,” *IEEE Transactions on Signal Processing*, vol. 53, no. 11, pp. 4125–4135, 2005.

[130] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” presented at the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 10684–10695.

[131] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, “Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 22500–22510.

[132] M. Chen, I. Laina, and A. Vedaldi, “Training-free layout control with cross-attention guidance,” presented at the Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2024, pp. 5343–5353.

[133] Ultralytics, “YOLOv5: A state-of-the-art real-time object detection system,” 2021.

[134] J. Hessel, A. Holtzman, M. Forbes, R. Le Bras, and Y. Choi, “Clipscore: A reference-free evaluation metric for image captioning,” *arXiv preprint arXiv:2104.08718*, 2021.

[135] G. Hildebrandt, P. Habiger, D. Dittler, M. Barth, R. Drath, and M. Weyrich, “A Methodology for classifying Data relevance to utilize external Data Sources in the Digital Twin,” *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–4. doi: 10.1109/ETFA52439.2022.9921717.