

**BACHELORARBEIT**

# Contrastive Learning mit Stable Diffusion-basierter Datenaugmentation

Verbesserung der Bildklassifikation  
durch synthetische Daten

---

vorgelegt am 16. September 2024  
Paul Hofmann

Erstprüferin: Prof. Dr. Larissa Putzar  
Zweitprüfer: Prof. Dr. Jan Neuhöfer

---

**HOCHSCHULE FÜR ANGEWANDTE  
WISSENSCHAFTEN HAMBURG**  
Department Medientechnik  
Finkenau 35  
22081 Hamburg

**FRAUNHOFER-INSTITUT FÜR  
PRODUKTIONSANLAGEN UND  
KONSTRUKTIONSTECHNIK IPK**  
Pascalstraße 8–9  
10587 Berlin

## **Zusammenfassung**

Der Arbeit beginnt mit einer kurzen Beschreibung ihrer zentralen Inhalte, in der die Thematik und die wesentlichen Resultate skizziert werden. Diese Beschreibung muss sowohl in deutscher als auch in englischer Sprache vorliegen und sollte eine Länge von etwa 150 bis 250 Wörtern haben. Beide Versionen zusammen sollten nicht mehr als eine Seite umfassen. Die Zusammenfassung dient u. a. der inhaltlichen Verortung im Bibliothekskatalog.

## **Abstract**

The thesis begins with a brief summary of its main contents, outlining the subject matter and the essential findings. This summary must be provided in German and in English and should range from 150 to 250 words in length. Both versions combined should not comprise more than one page. Among other things, the abstract is used for library classification.

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>III</b>
<b>Tabellenverzeichnis</b>	<b>IV</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Zielsetzung . . . . .	1
1.3 Aufbau der Arbeit . . . . .	2
<b>2 Theoretische Grundlagen</b>	<b>3</b>
2.1 Maschinelles Lernen . . . . .	3
2.1.1 Überwachtes und unüberwachtes Lernen . . . . .	4
2.1.2 Deep Learning . . . . .	5
2.1.3 Neuronale Netze . . . . .	6
2.1.4 Overfitting . . . . .	8
2.1.5 Datenaugmentation . . . . .	9
2.1.6 Out-of-Distribution Daten . . . . .	10
2.2 Contrastive Learning . . . . .	10
2.2.1 Unsupervised Contrastive Learning . . . . .	11
2.2.2 Supervised Contrastive Learning . . . . .	12
2.3 Synthetische Daten . . . . .	13
2.3.1 Variational Autoencoder . . . . .	14
2.3.2 Generative Adversarial Networks . . . . .	16
2.3.3 Diffusionsmodelle . . . . .	17
2.3.4 DA-Fusion . . . . .	19
2.4 Klassifikation von Gebrauchsgegenständen für die Recyclingwirtschaft . . . . .	21
2.4.1 Herausforderungen bei der Generierung synthetischer Daten . . . . .	23
2.4.2 Synthetische Daten als negativ-Beispiele im Contrastive Learning . . . . .	24
2.4.3 Integration von DA-Fusion und Supervised Contrastive Learning . . . . .	24
<b>3 Methodisches Vorgehen</b>	<b>25</b>
3.1 MVIP-Datensatz . . . . .	25
3.1.1 Teildatensatz . . . . .	25

3.1.2	Vorverarbeitung . . . . .	27
3.2	Implementierung . . . . .	27
3.2.1	DA-Fusion . . . . .	27
3.2.2	Supervised Contrastive Learning . . . . .	28
3.3	Versuchsaufbau . . . . .	29
3.3.1	Synthetische Datengenerierung . . . . .	29
3.3.2	Trainings- und Testdurchläufe . . . . .	30
3.4	Evaluationsmethoden und Metriken . . . . .	31
<b>4</b>	<b>Ergebnisse</b>	<b>32</b>
4.1	Die generierten synthetischen Daten . . . . .	32
4.1.1	In-Distribution . . . . .	32
4.1.2	Near Out-of-Distribution . . . . .	32
4.2	Trainings- und Testergebnisse . . . . .	32
4.2.1	Contrastive Pre-Training . . . . .	32
4.2.2	Lineare Klassifikation . . . . .	34
4.3	Vergleich der Ergebnisse . . . . .	34
4.3.1	Mit und ohne In-Distribution-Augmentationen . . . . .	35
4.3.2	Mit und ohne Near Out-of-Distribution-Augmentationen . . . . .	35
<b>5</b>	<b>Diskussion</b>	<b>36</b>
5.1	Eignung von DA-Fusion für die synthetische Datengenerierung . . . . .	36
5.2	Wirksamkeit von Near Out-of-Distribution-Augmentationen im Supervised Contrastive Learning . . . . .	36
<b>6</b>	<b>Fazit</b>	<b>37</b>
6.1	Zusammenfassung der wichtigsten Erkenntnisse . . . . .	37
6.2	Beantwortung der Forschungsfragen . . . . .	37
6.3	Ausblick und potenzielle Weiterentwicklungen . . . . .	37
<b>Literatur</b>		<b>38</b>
<b>Anhang</b>		<b>41</b>

# Abbildungsverzeichnis

2.1	Ein einfaches Venn-Diagramm, das die Beziehung zwischen KI, ML und DL veranschaulicht . . . . .	5
2.2	Abbildung des McCulloch-Pitts-Modells eines Neurons. . . . .	6
2.3	Architektur eines Convolutional Neural Networks. . . . .	8
2.4	Einige Beispiele für unterschiedliche Datenaugmentationstechniken und Kombinationen, die in (Chen et al., 2020) verwendet wurden. . . . .	9
2.5	Überblick über die GAN-Struktur. . . . .	17
2.6	Darstellung der Vorwärts- und Rückwärtsdiffusion in einem Diffusionsmodell.	18
2.7	Darstellung des Vorwärts- und Rückwärtsdiffusion in Stable Diffusion: Die Diffusionsprozesse werden auf die latenten Darstellungen der Bilder ange- wendet (rechts), welche vorher mit einem VAE erstellt wurden (links). . . . .	20
2.8	Vergleich zwischen semantischen Augmentationen aus Baseline-Methode und DA-Fusion. . . . .	21
2.9	Überblick über den Prozess zur Datenaugmentation mit DA-Fusion. . . . .	21
2.10	Integration des in EIBA entwickelten KI-gestützten Systems in den Prozess der Identifikation, Inspektion und Sortierung von Altteilen. . . . .	22
2.11	Die Mensch-Maschine-Schnittstelle des Systems. . . . .	23
3.1	Beispielbilder aus dem MVIP-Datensatz, die auf die Region of Interest (ROI) zugeschnitten wurden. . . . .	26
4.1	Vergrößerte Ausschnitte der synthetischen In-Distribution-Daten. . . . .	33
4.2	Beispieltext . . . . .	33
4.3	Beispieltext 1 . . . . .	34
4.4	Beispieltext 2 . . . . .	34
4.5	Beispieltext 3 . . . . .	35

# Tabellenverzeichnis

3.1	Auswahl der 20 Klassen aus der Oberklasse „CarComponent“ für den MVIP-Teildatensatz. . . . .	26
4.1	Testergebnisse der linearen Klassifikation. . . . .	35

# 1 Einleitung

...

## 1.1 Motivation

...

## 1.2 Zielsetzung

...

**Forschungsfrage 1:** Kann DA-Fusion für den EIBA-Datensatz synthetische Augmentationen erzeugen, die die Generalisierungsfähigkeit im Supervised Contrastive Learning verbessern?

Durch Beantwortung dieser Frage soll festgestellt werden, ob sich DA-Fusion grundsätzlich eignet, um die Herausforderungen der synthetischen Datengenerierung in Anwendungsfällen wie dem EIBA-Datensatz zu bewältigen (genaueres zum Datensatz in Abschnitt 3.1). Dazu wird DA-Fusion auf „normale“ Weise verwendet, d.h. es werden synthetische In-Distribution Daten generiert, die die Repräsentationen der realen Daten verbessern sollen. Es wird untersucht, ob die Verwendung der Augmentationen im Supervised Contrastive Learning dazu beiträgt, die Leistung des Modells für zuvor ungesehene Daten zu verbessern.

...

**Forschungsfrage 2:** Trägt die Verwendung von Out-of-Distribution (OOD) Augmentationen im Supervised Contrastive Learning dazu bei, die Robustheit des Modells gegenüber OOD-Daten zu erhöhen und die Repräsentationen von In-Distribution-Daten zu verbessern?

Im Rahmen dieser Frage wird untersucht, ob Near OOD-Augmentationen –also synthetische Daten, welche aus den echten Objekten abgeleitet sind, diese aber nicht akkurat darstellen müssen –im Supervised Contrastive Learning einen Mehrwert bieten, indem sie die Repräsentationen der In-Distribution-Daten verbessern und die Robustheit gegenüber OOD-Daten erhöhen. Dazu wird eine neue Negative Sampling-Strategie für das Supervised

Contrastive Learning verwendet, die es ermöglicht, für jeden Anchor genau die Near OOD-Augmentationen als negativ-Beispiele heranzuziehen, die aus einem Beispiel der Anchor-Klasse generiert wurden. Es wird untersucht, ob so die Generalisierungsfähigkeit und die Robustheit gegenüber OOD-Daten noch weiter gesteigert werden kann.

...

### **1.3 Aufbau der Arbeit**

...

## 2 Theoretische Grundlagen

Das folgende Kapitel gibt eine Einführung in die theoretischen Grundlagen, die für das Verständnis der Arbeit notwendig sind. Dabei werden vor allem zentrale Konzepte und Methoden des maschinellen Lernens, der synthetischen Datengenerierung, sowie des Contrastive Learning behandelt. Zuletzt wird der Anwendungsfall vorgestellt, den die Arbeit behandelt, und ein eigener Ansatz zur Integration von DA-Fusion und Supervised Contrastive Learning definiert.

### 2.1 Maschinelles Lernen

Maschinelles Lernen (ML) hat sich zu einem zentralen Bestandteil der Künstlichen Intelligenz (KI) entwickelt, einem interdisziplinären Forschungsfeld, das sich mit Algorithmen und Techniken befasst, welche es Computern ermöglichen, menschenähnliche Intelligenz zu erlangen.

Die ersten großen Durchbrüche in der KI kamen im Bezug auf Aufgaben, die für Menschen intellektuell eine große Herausforderung darstellten, die aber von Computern relativ einfach zu lösen waren, da sie als Liste formaler, mathematischer Regeln beschrieben werden konnten. Die große Schwierigkeit lag hingegen in den Aufgaben, die für Menschen relativ einfach und intuitiv sind, welche sich aber nur schwer formal beschreiben lassen. Hierunter fallen z.B. die Spracherkennung, oder Objekterkennung (Goodfellow et al., 2016).

Maschinelles Lernen bezeichnet einen Ansatz, bei dem Computer mit der Fähigkeit ausgestattet werden, selbstständig Wissen aus Erfahrung zu generieren, indem Muster und Konzepte aus rohen Daten erlernt werden. So kann ein Computerprogramm auf Basis von Beispielen lernen, wie es eine bestimmte Aufgabe lösen soll, ohne dass ihm explizit Regeln oder Algorithmen vorgegeben werden.

Eine allgemeine Definition für Maschinelles Lernen bietet (Mitchell, 1997):

Ein Computerprogramm soll aus Erfahrung  $E$  in Bezug auf eine Klasse von Aufgaben  $T$  und Leistungsmaß  $P$  lernen, wenn sich seine Leistung bei Aufgaben  $T$ , gemessen durch  $P$ , mit Erfahrung  $E$  verbessert.

Die Erfahrung  $E$  besteht dabei aus einer Menge von Trainingsdaten, beispielsweise Bilder. Die Aufgaben  $T$  können sehr unterschiedlich sein, von einfachen Klassifikations- und Regressionsaufgaben bis hin zu komplexen Problemen wie Spracherkennung oder autonomes Fahren. Das Leistungsmaß  $P$  gibt an, wie gut die Aufgaben  $T$  gelöst werden, und kann z.B. die sogenannte Accuracy sein, welche den Anteil korrekt klassifizierter Beispiele angibt.

Durch das Lernen aus den Trainingsdaten ergibt sich ein *Modell* des zugrundeliegenden Problems, das dann auf neue, unbekannte Daten angewendet werden kann, um Vorhersagen zu treffen oder Entscheidungen zu treffen.

### 2.1.1 Überwachtes und unüberwachtes Lernen

Wie genau Wissen aus Erfahrung bzw. aus Rohdaten generiert wird hängt vom gewählten Verfahren ab. Im Maschinellen Lernen gibt es dabei zwei zentrale Paradigmen, das überwachte und das unüberwachte Lernen.

Beim überwachten Lernen (Supervised Learning) wird das Modell mit einem vollständig annotierten Datensatz trainiert. Das heißt meistens, dass jeder Datenpunkt mit einem Klassenlabel versehen ist, sodass Eingabe-Ausgabe-Paare entstehen. Ziel ist es, eine Funktion zu lernen, welche die Eingaben auf die entsprechenden Ausgaben abbildet. Ein einfaches Beispiel wäre ein Bildklassifikator, der darauf trainiert wird, Katzen und Hunden zu unterscheiden. Hier würden alle Trainingsbilder entweder mit dem Label „Katze“ oder „Hund“ versehen sein. Im Training kann die Vorhersage des Modells dann mit dem tatsächlichen Label verglichen werden, um den Fehler zu berechnen und die Modellparameter entsprechend anzupassen.

Im Gegensatz dazu arbeitet unüberwachtes Lernen (Unsupervised Learning) mit unbeschrifteten Daten; es gibt also keine vorgegebenen Ausgaben. Stattdessen wird versucht, ein Modell zu befähigen, eigenständig Muster und Strukturen in den Daten zu erkennen und z.B. nützliche Repräsentationen der Eingangsdaten zu erlernen. Zu den häufigsten Methoden des unüberwachten Lernens gehören Clustering- und Assoziationsalgorithmen. Ein Beispiel ist die Segmentierung von Kunden in verschiedene Gruppen basierend auf ihrem Kaufverhalten (**<empty citation>**).

In der Praxis werden oft auch hybride Ansätze genutzt, wie das semi-überwachte Lernen (Semi-Supervised Learning), bei dem eine Kombination aus beschrifteten und unbeschrifteten Daten verwendet wird, oder das selbstüberwachte Lernen (Self-Supervised Learning), bei dem das Modell eigenständig Teile der Daten zur Erzeugung von Überwachungssignalen verwendet, anstatt sich auf externe, von Menschen bereitgestellte Labels zu verlassen.

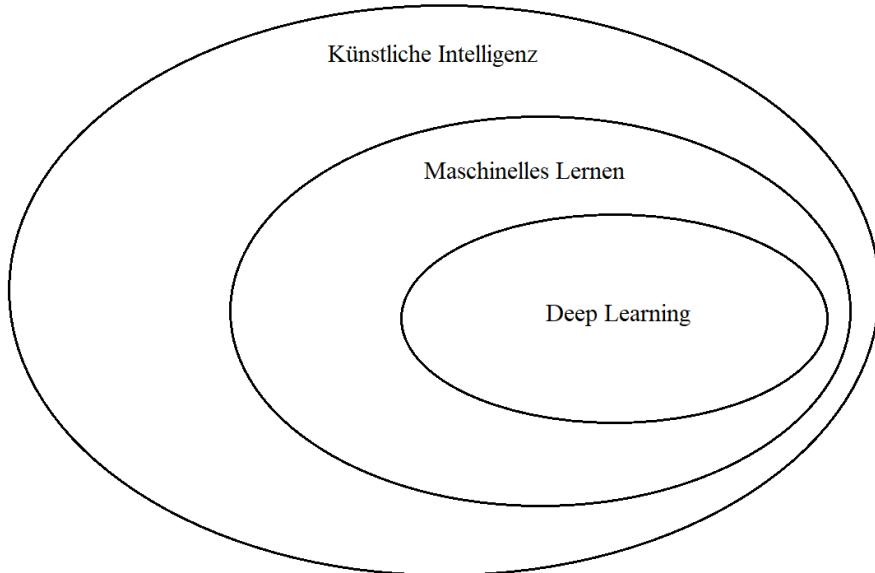


Abbildung 2.1: Ein einfaches Venn-Diagramm, das die Beziehung zwischen KI, ML und DL veranschaulicht.

### 2.1.2 Deep Learning

Das Wissen, das ein Modell aus den Trainingsdaten lernt, wird in Form von sogenannten Features repräsentiert. Diese Features können einfache Konzepte wie Kanten oder Farben sein, oder komplexere Konzepte wie Gesichter oder Objekte. Unter Deep Learning (DL) versteht man eine tiefe, hierarchische Vernetzung dieser Konzepte, sodass komplexere Konzepte auf simpleren Konzepten aufbauen können. Visuell veranschaulicht entsteht ein Graph mit vielen Ebenen, oder *Deep Layers* (Goodfellow et al., 2016).

Deep Learning ist also eine Unterkategorie des Maschinellen Lernens (siehe Abbildung 2.1), bei der die Eingabedaten mehrere Verarbeitungsschichten durchlaufen, um eine hierarchische Repräsentation zu ermöglichen. Jede Schicht transformiert die Eingabedaten in eine etwas abstraktere Darstellung. Deep Learning fällt demnach auch unter den Begriff des Representation Learning (Zhou, 2021).

Heutzutage bilden Deep Learning-Modelle die Grundlage für viele Anwendungen der Künstlichen Intelligenz, darunter Bild- und Spracherkennung, maschinelle Übersetzung, medizinische Diagnose und autonomes Fahren. Die rasante Entwicklung in diesem Bereich ist vor allem auf die Verfügbarkeit großer Datenmengen und immer leistungsfähigere Hardware zurückzuführen (Goodfellow et al., 2016).

### 2.1.3 Neuronale Netze

Während die rasante Entwicklung von Deep Learning vor allem in den vergangenen Jahren spürbar geworden ist, sind die zugrundeliegenden Algorithmen und Konzepte schon seit Jahrzehnten bekannt (Zhou, 2021). Dabei bildet das künstliche neuronale Netz (KNN) die Grundlage der allermeisten Deep-Learning-Modelle. Es ist inspiriert von der Struktur und Funktionsweise des menschlichen Gehirns und besteht aus einer Vielzahl von miteinander verbundenen Knoten (Neuronen), die in Schichten organisiert sind. Die Struktur eines neuronalen Netzes besteht aus einer Eingabeschicht (Input Layer), einer oder mehreren versteckten Schichten (Hidden Layers) und einer Ausgabeschicht (Output Layer).

Die einzelnen Neuronen, auf dem diese Netze aufbauen, sind eine mathematische Modellierung des biologischen Neurons, das erstmals 1943 von Warren McCulloch und Walter Pitts vorgestellt wurde (Zhou, 2021). In Abbildung 2.2 ist das Modell dargestellt.

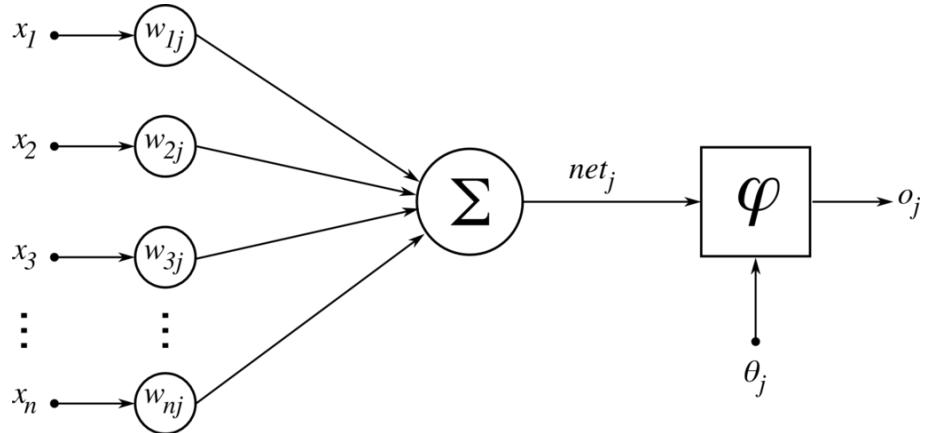


Abbildung 2.2: Abbildung des McCulloch-Pitts-Modells eines Neurons (Burgmer, 2005).

Jedes Neuron empfängt eine Reihe von Eingaben  $x_{1\dots n}$ , entweder von externen Quellen oder von den Ausgaben anderer Neuronen. Für jede dieser Eingaben gibt es zugehörige Gewichtungen (Weights)  $w_{1j\dots nj}$ , welche die Stärke und Richtung (positiv oder negativ) des Einflusses der jeweiligen Eingaben auf das Neuron  $j$  bestimmen. Das Neuron berechnet dann die gewichtete Summe  $net_j$  aller Eingaben und falls ein bestimmter Schwellenwert (Bias)  $\theta$  überschritten wurde, wird das Neuron aktiviert. Die Aktivierung  $o_j$  des Neurons kann dementsprechend folgendermaßen beschrieben werden:

$$o_j = \phi\left(\sum_{i=1}^n w_i x_i - \theta_j\right) \quad (2.1)$$

Die Aktivierungsfunktion  $\phi$  kann dabei unterschiedlich gewählt werden, um die Ausgabe des Neurons zu modellieren. Eine simples Beispiel ist die sogenannte Schwellenwertfunktion, die den Wert 1 zurückgibt, wenn die gewichtete Summe größer als der Schwellenwert ist, sonst 0. Eine häufig verwendete Aktivierungsfunktion ist jedoch die sogenannte Sigmoid-Funktion, die kontinuierlich und differenzierbar ist und somit die Optimierung des Netzwerks vereinfacht:

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (2.2)$$

Während die Sigmoid-Funktion allerdings nur für binäre Klassifikationen geeignet ist, wird für die Klassifikation von mehreren Klassen die Softmax-Funktion verwendet, die die Wahrscheinlichkeitsverteilung über alle Klassen berechnet:

$$\text{Softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}} \quad (2.3)$$

Im Training fließen die Eingabedaten in einer Vorwärtsausbreitung (Forward Propagation) durch das Netzwerk, um die Ausgabe zu berechnen. Es wird dann eine geeignete Verlustfunktion angewendet, um den Fehler (Loss) des Modells zu berechnen. Das Ziel des Trainings ist es, die Gewichtungen der Neuronen so anzupassen, dass der Fehler minimiert wird.

Diese Optimierung geschieht durch eine Rückwärtsausbreitung (Backpropagation), welche den berechneten Fehler rückwärts durch das Netz propagiert, um die Gewichte und Schwellenwerte um einen geringen Wert in die Richtung anzupassen, die den Fehler minimieren würde. Die Richtung wird bestimmt, indem der Gradient der Verlustfunktion berechnet wird. So bewegt sich das Modell iterativ entlang des Gradienten hin zu einem lokalen Minimum der Verlustfunktion. Dieser Algorithmus wird als Gradient Descent (Gradientenabstieg) bezeichnet. Im Maschinellen Lernen kommt jedoch hauptsächlich der Stochastic Gradient Descent (SGD) zum Einsatz, der immer nur einen kleinen, zufällig ausgewählten Teil der Trainingsdaten (einen sogenannten Batch) verwendet, um den Gradienten zu berechnen.

Es gibt eine Reihe von sogenannten Hyperparametern, welche einen großen Einfluss auf die Leistung des Modells haben und vorher manuell konfiguriert werden, oftmals durch Ausprobieren verschiedener Werte. Die Lernrate (Learning Rate) bestimmt beispielsweise, wie groß die Schritte sind, die entlang des Gradienten gemacht werden, während die Batch-Größe (Batch Size) die Anzahl der Trainingsdaten bestimmt, die in einem Schritt des SGD verwendet werden. Die Anzahl der Epochen gibt an, wie oft der gesamte Trainingsdatensatz durchlaufen wird. Andere Hyperparameter, wie die Anzahl der versteckten Schichten und Neuronen, bestimmen die Komplexität des Modells.

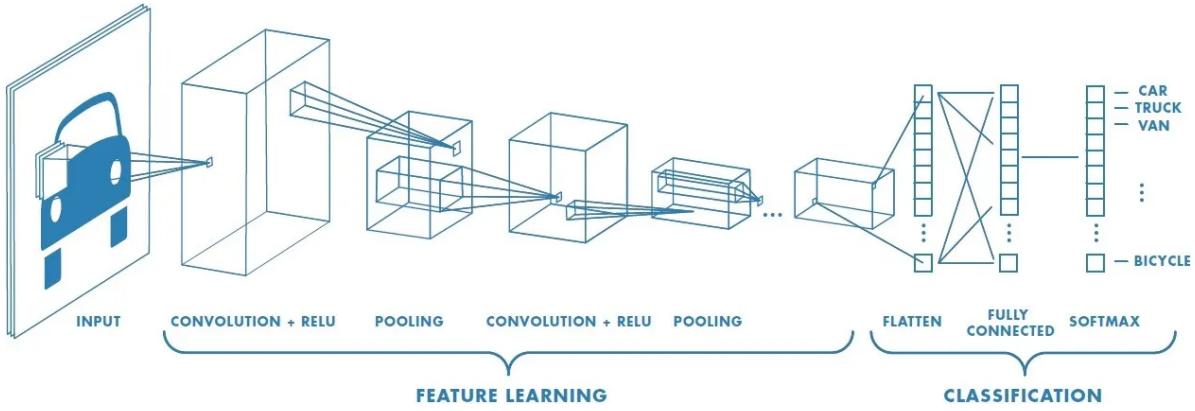


Abbildung 2.3: Architektur eines Convolutional Neural Networks (Saha, 2018).

Deep Learning mit neuronalen Netzen kann gut am Beispiel des Convolutional Neural Networks (CNN) veranschaulicht werden, welches speziell für die Verarbeitung von Bildern entwickelt wurde (siehe Abbildung 2.3). Ein CNN besteht aus mehreren Schichten, darunter Convolutional Layers, Pooling Layers und Fully Connected Layers. Die Convolutional Layers extrahieren sogenannte *Feature Maps* aus den Eingabebildern, indem sie Faltungskerne über das Bild schieben und die gewichteten Summen der Pixel berechnen. Die Pooling Layers reduzieren die Dimensionalität der Feature Maps, indem sie die Größe der Merkmale reduzieren. Die Fully Connected Layers kombinieren die extrahierten Merkmale, um die endgültige Klassifikation vorzunehmen.

Mit Backpropagation trainierte CNNs wurden erstmals in (Cun et al., 1990) behandelt, bilden aber auch heute noch die Grundlage von zahlreichen Deep Learning-Anwendungen und Methoden. Sie sind vor allem deshalb so erfolgreich in der Bildverarbeitung, weil sie die räumliche Struktur von Bildern berücksichtigen und dadurch eine hohe Genauigkeit bei der Klassifikation erreichen können.

## 2.1.4 Overfitting

Wenn ein Modell so stark an die Trainingsdaten angepasst wird, dass es nicht in der Lage ist, auf neuen, unbekannten Daten zu generalisieren, spricht man von Overfitting (Goodfellow et al., 2016). Statt die zugrundeliegenden Muster zu lernen, speichert das Modell auch zufällige Rauschsignale und Fehler. Dies führt dazu, dass das Modell auf den Trainingsdaten eine hohe Genauigkeit erreicht, aber auf neuen Daten eine schlechtere Leistung zeigt. Man verwendet daher neben den Trainingsdaten auch Validierungsdaten zur Überwachung des Overfittings.

Oftmals liegt das Problem in der Quantität und Qualität der Trainingsdaten. Sind nicht genügend Daten vorhanden, oder sind die Daten nicht divers genug, kann das Modell nicht

generalisieren.

Ein anderer Grund für Overfitting kann eine zu hohe Komplexität des Modells sein, wodurch die Trainingsdaten zu genau modelliert werden. Techniken zur Regularisierung zielen deshalb darauf ab, die optimale Komplexität eines Modells zu finden. Dropout (Srivastava et al., 2014) ist eine solche Technik, bei der zufällig ausgewählte Neuronen während des Trainings deaktiviert werden, um zu verhindern, dass das Modell sich auf spezifische Merkmale verlässt. Als Early Stopping wird ein anderer Ansatz bezeichnet, bei dem das Training beendet wird, wenn die Leistung auf den Validierungsdaten beginnt, sich zu verschlechtern.

## 2.1.5 Datenaugmentation

Datenaugmentation ist einer der wichtigsten Ansätze zur Vermeidung von Overfitting und findet in fast allen Methoden des Maschinellen Lernens Anwendung. Denn das Modell kann auch ohne völlig neue Daten zu beschaffen robuster gegenüber Variationen gemacht werden, indem es mit leicht veränderten Versionen der Trainingsdaten trainiert wird.

Bei der Datenaugmentation werden unterschiedliche Transformationen auf die vorhandenen Daten angewendet, z.B. Rotation, Skalierung, Verschiebung, Spiegelung, Helligkeitsanpassung oder Rauschen (Shorten & Khoshgoftaar, 2019). Die Transformationen werden meist mit zufälligen parametrisiert, um eine Vielzahl von Variationen zu erzeugen. Das Ziel ist es, das Modell zu zwingen, die zugrundeliegenden Muster der Daten zu lernen, anstatt sich auf spezifische Merkmale zu verlassen, die nur in den Trainingsdaten vorhanden sind. Einige Beispiele für Datenaugmentationstechniken sind in Abbildung 2.4 dargestellt.

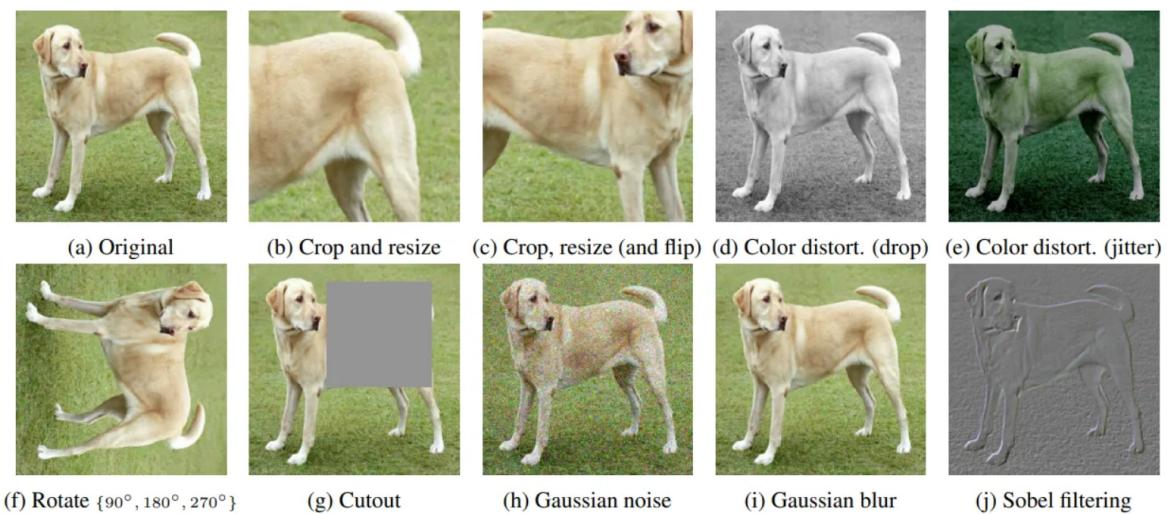


Abbildung 2.4: Einige Beispiele für unterschiedliche Datenaugmentationstechniken und Kombinationen, die in (Chen et al., 2020) verwendet wurden.

### 2.1.6 Out-of-Distribution Daten

Wenn ein KI-Modell mit Daten konfrontiert wird, die außerhalb des Bereichs liegen, den es während des Trainings gesehen hat, spricht man von Out-of-Distribution (OOD)-Daten. Es handelt sich also um Datenpunkte oder Muster, die sich signifikant von den Trainingsdaten unterscheiden. Dies kann zu Problemen führen, da das Modell möglicherweise nicht in der Lage ist, angemessene Vorhersagen oder Entscheidungen für diese Daten zu treffen. Die Erkennung von OOD-Daten ist daher ein wichtiges Forschungsgebiet im maschinellen Lernen, da sie dazu beitragen kann, die Zuverlässigkeit und Sicherheit von KI-Systemen zu verbessern.

Idealerweise gibt ein neuronales Netz höhere Softmax-Wahrscheinlichkeiten für In-Distribution (ID)-Daten und niedrigere Wahrscheinlichkeiten für OOD-Daten aus. Tatsächlich sind die Wahrscheinlichkeiten für OOD-Daten fast immer sehr hoch, meist reicht aber der Abstand zwischen ID- und OOD-Daten aus, um einen Schwellenwert festzulegen und OOD-Instanzen frühzeitig zu identifizieren (Hendrycks & Gimpel, 2018).

Oftmals ist die OOD-Detektion aber weniger trivial, etwa wenn sich ID- und OOD-Daten sehr ähnlich sind. Es wird daher stets nach alternativen Ansätzen gesucht, um die OOD-Detektion zu verbessern. In (Hendrycks & Gimpel, 2018) wird dabei ein Klassifikator erweitert, um Rekonstruktionen der Eingabedaten zu erstellen und den Fehler zwischen den Original- und Rekonstruktionsdaten zu messen. Ein hoher Rekonstruktionsfehler kann auf OOD-Daten hinweisen. Auch das Temperature Scaling (Guo et al., 2017), welches die Softmax-Wahrscheinlichkeiten kalibriert, kann zusammen mit kleinen Störungen in den Eingabedaten für zuverlässigere Wahrscheinlichkeitswerte und bessere OOD-Detektion sorgen (Liang et al., 2020).

## 2.2 Contrastive Learning

Ein zentrales Thema dieser Arbeit ist das Contrastive Learning. Es ordnet sich in den Bereich des Representation Learning ein, bei dem es darum geht, nützliche Repräsentationen von Daten zu lernen. Die Besonderheit des Contrastive Learning ist, dass es auf der Kontrastierung von Daten basiert, um ähnliche Beispiele zu gruppieren und unähnliche Beispiele voneinander zu trennen. Dies hat sich als effektive Methode mit erstaunlicher Generalisierungsfähigkeit und Robustheit gegenüber Adversarial Attacks erwiesen (Liu, 2021).

Das Contrastive Learning stammt ursprünglich aus dem unüberwachten Lernen, wird aber oft als selbstüberwachte Methode bezeichnet, da die Kontrastierung der Daten eine Art Selbstüberwachung darstellt. Die erlernten Repräsentationen haben sich als deutlich genauer als im traditionellen überwachten Lernen herausgestellt, wo diese ausschließlich zur

Klassenunterscheidung optimiert werden (Keshtman et al., 2022). Da die Methode nicht auf annotierte Daten angewiesen ist, kann sie ein sehr effizienter Ansatz sein, um Modelle vorzutrainieren, die sich durch Finetuning an spezifische Aufgaben anpassen, während sie gleichzeitig allgemeinere, von der Aufgabe unbeeinflusste Merkmale lernen (Radford et al., 2021).

Trotzdem gibt es vermehrt Ansätze, Contrastive Learning auch im überwachten Setting anzuwenden. Hier wird nicht nur zwischen einzelnen Instanzen unterschieden, sondern auch die Klassenzugehörigkeit der Beispiele berücksichtigt.

In den folgenden Abschnitten wird genauer auf die Funktionsweise von sowohl unüberwachten als auch überwachten Varianten des Contrastive Learning eingegangen, die in den letzten Jahren vielversprechende Ergebnisse erzielt haben.

### 2.2.1 Unsupervised Contrastive Learning

Ob überwacht oder unüberwacht: Im Contrastive Learning soll die Distanz ähnlicher Beispiele in einem latenten Repräsentationsraum minimiert und die Distanz unähnlicher Beispiele maximiert werden. Dabei zieht das Modell verschiedene Ankerbeispiele heran und kontrastiert sie mit positiv- und negativ-Beispielen.

Es kommen verschiedene Verlustfunktionen zum Einsatz, wobei der Fehler grundsätzlich darüber aussagt, ob ähnliche Beispiele auch im Repräsentationsraum nahe beieinander liegen. Verlustfunktionen unterscheiden sich zum Beispiel in der Wahl der Distanzmetrik, oder der Anzahl der positiven und negativen Beispiele, die für den Vergleich herangezogen werden.

Das wohl prominenteste Beispiel für Contrastive Learning in der visuellen Domäne ist **SimCLR**, das in (Chen et al., 2020) vorgestellt wurde. SimCLR verwendet den sogenannten NT-Xent Loss (Normalized Temperature-scaled Cross-Entropy Loss), der die Ähnlichkeiten zwischen allen Paaren im Batch berücksichtigt, anstatt nur einzelne Triplets oder Paare. Jedes Beispiel wird dabei zweimal augmentiert, um zwei Ansichten zu erzeugen, welche als positives Paar für das jeweilige Beispiel dienen. Alle anderen Beispiele (bzw. dessen Ansichten) im Batch werden als negativ-Beispiele gesehen.

Die Eingabedaten werden durch ein CNN in eine latente Repräsentation transformiert. Dieser Schritt wird auch als *Feature Extraction* bezeichnet. SimCLR verwendet anschließend einen sogenannten *Projection Head*, der die encodierten Repräsentationen weiter transformiert, um einen Repräsentationsraum zu erzeugen, der für die Unterscheidung der Beispiele geeignet ist. In diesem Representationsraum werden die Ähnlichkeitswerte der Paare berechnet, um den Fehler zu bestimmen.

Dafür wird die Kosinus-Ähnlichkeit  $s_{i,j}$  der Paare  $z_i$  und  $z_j$  berechnet:

$$s_{i,j} = \frac{z_i \cdot z_j}{\|z_i\| \cdot \|z_j\|} \quad (2.4)$$

Der Fehler ergibt sich dann aus der Berechnung des Softmax über die Ähnlichkeiten aller Paare im Batch, skaliert mit einem Temperaturparameter, um die Unterscheidung zwischen positiven und negativen Paaren hervorzuheben.

Durch die Verwendung aggressiver Datenaugmentation zur Erzeugung der zwei Ansichten wird die Robustheit der Repräsentationen verbessert. Größere Batch Sizes und längere Trainingszeiten begünstigen die Lernfähigkeit des Modells. Besonders die Wahl von *Hard Negatives*, also von Paaren, welche ähnliche Konzepte darstellen, aber sehr unterschiedlich aussehen, hat sich als entscheidend für den Erfolg des Modells erwiesen.

Eine neuere Variante von Contrastive Learning ist **StableRep** (Tian et al., 2023). Diese Methode verwendet synthetische Daten, die von Diffusionsmodellen generiert wurden, insbesondere von Stable Diffusion. Dabei werden alle Bilder, die aus dem selben Prompt generiert wurden, als positive Beispiele voneinander betrachtet. Es hat sich gezeigt, dass StableRep mit den richtigen Einstellungen auch mit Training nur auf synthetischen Daten die Leistung von SimCLR übertreffen kann. Noch bessere Ergebnisse werden erzielt, wenn Textsupervision in das Training einbezogen wird.

### 2.2.2 Supervised Contrastive Learning

Trotz der vielversprechenden Ergebnisse im unüberwachten Kontext, gibt es auch im überwachten Setting vermehrt Interesse an Contrastive Learning. Hierbei wird die Klassenzugehörigkeit der Daten genutzt, um die Repräsentationen der Beispiele zu verbessern. Im Gegensatz zu unüberwachten Methoden, die auf der Unterscheidung von Instanzen basieren, zielen überwachte Methoden darauf ab, die Klassenzugehörigkeit der Beispiele zu berücksichtigen.

In (Khosla et al., 2021) wird **SupCon** vorgestellt, eine Weiterentwicklung der Verlustfunktion aus SimCLR, die das Contrastive Learning auf das überwachte Setting anpasst und mehrere positiv-Beispiele pro AnkerBeispiel berücksichtigt. Im Gegensatz zu unüberwachten Methoden, die ein Anchor-Beispiel, ein positives Beispiel und viele negative Beispiele verwenden, kommen im Supervised Contrastive Learning auch viele positiv-Beispiele pro Batch zum Einsatz. Diese positiv-Beispiele werden nicht mehr als Augmentationen des Anchor-Samples generiert, sondern als Samples der gleichen Klasse herangezogen. Dadurch soll auch die Notwendigkeit des Hard-Negative Minings reduziert werden. Trotzdem wird gezeigt, dass der resultierende Loss sowohl von hard-negatives wie auch hard-positives profitiert. Die Verwendung des Mittelwertes der positiven Repräsentationen stabilisiert das Training und führt zu einer verbesserten Leistung.

Im überwachten Kontext bieten sich auch neue Möglichkeiten zur Weiterentwicklung von Contrastive Learning. In (Kim et al., 2023) wird **Generalized SCL** vorgestellt, eine Methode, die die Label-Informationen als Verteilung betrachtet. Anstatt die Klassenzugehörigkeit als harte Kategorie zu betrachten, wird die Unsicherheit der Labels berücksichtigt. Dies ermöglicht es dem Modell, die Repräsentationen der Beispiele besser zu lernen und die Generalisierungsfähigkeit zu verbessern.

Eine weitere Methode zur Verbesserung des Supervised Contrastive Learning ist **SCL with Hard Negatives** (Jiang et al., 2024). Hier wird eine zusätzliche Einschränkung des Negative Samplings vorgenommen, um Hard Negatives zu selektieren. Diese sind Beispiele, die zwar unähnlich zum Anchor-Beispiel sind, aber dennoch nah genug im Repräsentationsraum, um die Repräsentationen zu verbessern. Diese Strategie hat sich als effektiv erwiesen, um die Generalisierungsfähigkeit der Modelle zu verbessern.

## 2.3 Synthetische Daten

Datenaugmentation wurde bereits als eine Möglichkeit vorgestellt, um die Menge und Vielfalt der Trainingsdaten zu erhöhen und damit die Generalisierungsfähigkeit des Modells zu verbessern. Allerdings kann die bloße Augmentation an ihre Grenzen stoßen, wenn die verfügbaren Trainingsdaten selbst nicht genügend Vielfalt aufweisen. In solchen Fällen können synthetische Daten eine nützliche Ergänzung sein. Anstatt einfache Transformationen auf die Eingangsdaten anzuwenden, werden völlig neue Datenpunkte generiert, die die zugrundeliegenden Muster der realen Daten nachahmen.

Während synthetische Daten auch manuell mit Hilfe von Simulationssoftware erstellt werden können, hat insbesondere die Entwicklung der generativen Modellierung in den letzten Jahren zu einer neuen Ära der synthetischen Datenerzeugung geführt. Diese Modelle sind in der Lage, komplexe Datenstrukturen zu lernen und realistische Daten zu generieren, die von echten Daten kaum zu unterscheiden sind. Diese Entwicklung ist vor allem auf die Fortschritte im Deep Learning zurückzuführen (Foster, 2020): Als Hierarchie der Mustererkennung können Deep Learning-Modelle das hohe Maß bedingter Abhängigkeiten zwischen Merkmalen in den Daten lernen und reproduzieren. Und als Form des Representation Learning erleichtert Deep Learning die Generierung von Daten, indem nur eine geeignete niedrigdimensionale Repräsentation gewählt werden muss, welches das Modell wieder zu einer realistischen Dateninstanz umwandeln soll.

Es sollen nun einige der wichtigsten generativen Modelle vorgestellt werden, um eine Grundlage für den aktuellen Stand der synthetischen Datengenerierung zu schaffen.

### 2.3.1 Variational Autoencoder

Ein Autoencoder ist eine spezielle Art von KI-Modell, das entwickelt wurde, um Daten effizient zu komprimieren und anschließend zu rekonstruieren. Es wurde im Wesentlichen in (Hinton & Salakhutdinov, 2006) vorgestellt, die Grundideen gehen jedoch bis in die 1980er Jahre zurück, z.B. (Rumelhart et al., 1986), wo auch das Backpropagation-Verfahren zur Optimierung neuronaler Netze beschrieben wurde.

Autoencoder bestehen aus zwei Hauptkomponenten (Foster, 2020):

- **Encoder:** Der Encoder komprimiert hochdimensionale Eingabedaten in einem niedrigen-dimensionalen Darstellungsvektor.
- **Decoder:** Der Decoder nimmt einen gegebenen Darstellungsvektor und wandelt ihn zurück in den ursprünglichen hochdimensionalen Raum um.

Der Darstellungsvektor repräsentiert das Originalbild als Punkt in einem mehrdimensionalen latenten Raum, wobei jede Dimension eine bestimmte Eigenschaft des Bildes kodiert. Es handelt sich beim Autoencoder deshalb um eine Form des *Representation Learning*.

Das Training eines Autoencoders erfolgt durch Minimierung des Rekonstruktionsfehlers, der die Differenz zwischen den ursprünglichen Eingabedaten und den rekonstruierten Ausgaben beschreibt. Eine gängige Verlustfunktion hierfür ist der *Mean Squared Error* (MSE):

$$Loss = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^2, \quad (2.5)$$

wobei  $x_i$  die Eingabedaten und  $\hat{x}_i$  die rekonstruierten Ausgaben sind.

Für die synthetische Datengenerierung sind Autoencoders deshalb interessant, weil man theoretisch durch die Wahl eines beliebigen Punkts im latenten Raum neue Bilder erzeugen kann, indem man diesen Punkt durch den Decoder schickt, da der Decoder gelernt hat, wie man Punkte im latenten Raum in realistische Bilder umwandelt. (Foster, 2020) In der herkömmlichen Form hat der Autoencoder in Bezug auf diese Aufgabe allerdings einige Schwachstellen. So lernt er einen festen Punkt im latenten Raum für jede Eingabe, was zu einem diskreten und unstrukturierten latenten Raum führt. Denn wenn zwei Punkte im latenten Raum nahe beieinander liegen, bedeutet das nicht unbedingt, dass die entsprechenden Bilder ähnlich sind, was die Wahl eines Punktes im latenten Raum für die Generierung neuer Daten erschwert. Da keine Modellierung der Datenverteilung im latenten Raum stattfindet, können dann auch keine realistischen Daten generiert werden, die nicht in den Trainingsdaten enthalten sind.

In (Kingma & Welling, 2022) wird der **Variational Autoencoder** (VAE) vorgestellt. Er adressiert die Schwachstellen des Autoencoders und verwendet probabilistische Methoden, um die Datenverteilung im latenten Raum zu modellieren; An Stelle eines einzelnen, festen Punkt im latenten Raum wird für jede Eingabe eine Verteilung gelernt, aus der die latenten Variablen stammen. Dadurch entsteht ein strukturierter und kontinuierlicher latenter Raum, der es ermöglicht, neue, realistische Daten zu generieren.

Der Encoder des VAEs berechnet einerseits den Mittelwert und die Standardabweichung der latenten Verteilung, und erzeugt außerdem eine zufällige Stichprobe aus dieser Verteilung. Der Decoder nimmt diese Stichprobe und rekonstruiert die Eingabedaten. Neben dem Rekonstruktionsfehler wird die Verlustfunktion des VAEs auch durch den Kullback-Leibler-Divergenzterm (KL-Divergenz) erweitert, der die Ähnlichkeit der gelernten Verteilung zu einer Standardnormalverteilung misst:

$$D_{KL}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)}, \quad (2.6)$$

wobei  $P$  die tatsächliche Verteilung und  $Q$  die approximierte Verteilung ist.

Die Gesamtverlustfunktion des VAEs ist dann die Summe aus Rekonstruktionsfehler und KL-Divergenz:

$$\text{Loss} = \text{Reconstruction Loss} + \text{KL-Divergence} \quad (2.7)$$

VAEs kommen noch immer in einer Vielzahl von Anwendungen zum Einsatz, darunter Bildgenerierung, Textgenerierung, Anomalieerkennung und semantische Segmentierung. (<empty citation>)

Dennoch haben VAEs auch einige Nachteile, da ihre zugrundeliegende Architektur ein sogenanntes Bottleneck-Problem aufweist, bei dem alle Informationen in den latenten Variablen komprimiert werden müssen. Es gehen zwangsweise Informationen verloren, was zu einer ungenauen Rekonstruktion der Eingabedaten führen kann. Darüber hinaus kann die Modellierung der Datenverteilung im latenten Raum schwierig sein, insbesondere bei komplexen Datenstrukturen. (<empty citation>)

### 2.3.2 Generative Adversarial Networks

Ein weiteres berühmtes KI-Modell ist das Generative Adversarial Network (GAN), das in (Goodfellow et al., 2014) vorgestellt wurde. GANs bestehen aus zwei neuralen Netzwerken, die gegeneinander antreten, um realistische synthetische Daten zu erzeugen. Diese Technologie hat sich als äußerst mächtig in der Bild- und Datengenerierung erwiesen.

Auch GANs bestehen aus zwei Hauptkomponenten, die allerdings eine andere Funktionsweise haben als die des Autoencoders:

- **Generator:** Das generative Netzwerk nimmt Zufallsrauschen als Eingabe und erzeugt daraus Daten, die möglichst realistisch wirken sollen. Der Generator versucht, die wahre Datenverteilung zu imitieren und realistische Beispiele zu erstellen.
- **Diskriminator:** Das diskriminative Netzwerk erhält sowohl echte Daten aus dem Trainingsdatensatz als auch die vom Generator erzeugten Daten. Seine Aufgabe ist es, zwischen echten und künstlichen Daten zu unterscheiden. Der Diskriminatior gibt eine Wahrscheinlichkeit aus, dass die Eingabedaten echt sind.

Der Trainingsprozess eines GANs ist in Abbildung 2.5 dargestellt und kann als minimax-Spiel zwischen dem Generator und dem Diskriminatior formuliert werden: Der Diskriminatior wird trainiert, um echte Daten von generierten Daten zu unterscheiden. Dies geschieht durch eine binäre Klassifikation („echt“ oder „synthetisch“). Der Diskriminatior passt seine Gewichte an, um die Unterscheidung zu verbessern. Der Generator wird trainiert, um den Diskriminatior zu täuschen. Dies geschieht, indem der Generator seine erzeugten Daten durch den Diskriminatior laufen lässt und die Rückmeldung (Gradienten) des Diskriminators verwendet, um seine eigenen Parameter zu optimieren. Ziel ist es, den Diskriminatior zu überlisten, sodass er die generierten Daten als echt klassifiziert.

Im Detail wird der Generator durch Minimierung der Log-Wahrscheinlichkeit, dass der Diskriminatior die generierten Daten als echt klassifiziert, trainiert. Der Diskriminatior wird durch Maximierung dieser Wahrscheinlichkeit trainiert. Dies führt zu einem Gleichgewichtszustand, in dem der Generator realistische Daten erzeugt, die den echten Daten ähneln, und der Diskriminatior nicht in der Lage ist, zwischen echten und generierten Daten zu unterscheiden.

Als Verlustfunktion wird die sogenannte *Jensen-Shannon-Divergenz* verwendet, die die Ähnlichkeit zwischen zwei Wahrscheinlichkeitsverteilungen misst:

$$D_{JS}(P\|Q) = \frac{1}{2}D_{KL}(P\|M) + \frac{1}{2}D_{KL}(Q\|M), \quad (2.8)$$

wobei  $P$  und  $Q$  die beiden Wahrscheinlichkeitsverteilungen und  $M$  der Mittelwert der beiden Verteilungen ist.

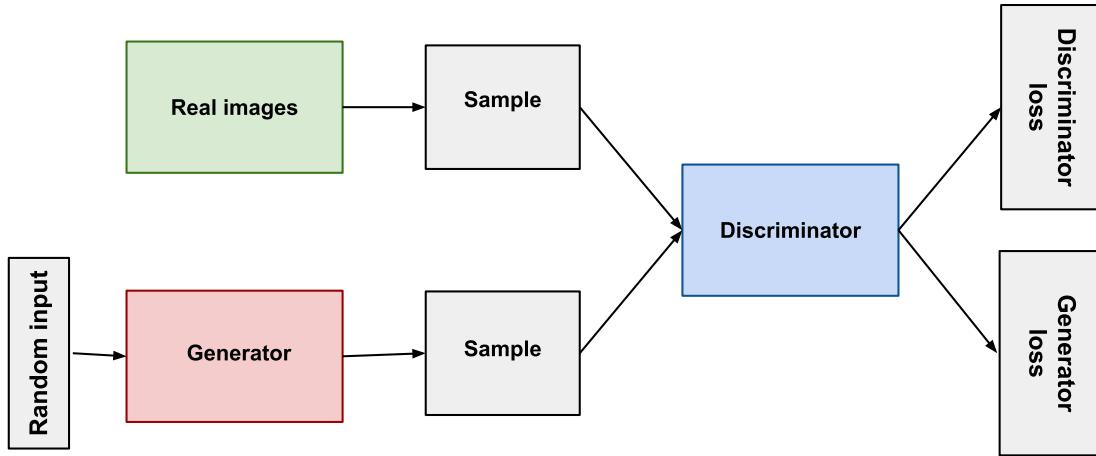


Abbildung 2.5: Überblick über die GAN-Struktur (Google for Developers, 2022).

Besonders in der Bildgenerierung haben GANs eine hohe Qualität erreicht und sind in der Lage, realistische Bilder zu erzeugen, die von echten Bildern kaum zu unterscheiden sind. GANs sind auch flexibel in Bezug auf die Eingangsdaten und können mit verschiedenen Datentypen wie Bildern, Texten oder Audiodaten arbeiten. Dennoch gibt es einige Nachteile. Beispielsweise kann das Training eines GANs sehr instabil sein, da es schwierig ist, ein Gleichgewicht zwischen Generator und Diskriminatator zu finden. Es kann auch zum sogenannten Modus-Kollaps kommen, bei dem der Generator nur eine begrenzte Anzahl von Beispielen erzeugt, da der Diskriminatator diese als besonders realistisch bewertet. Der Generator lernt dann, nur diese Beispiele zu reproduzieren, anstatt die gesamte Datenverteilung zu lernen. Darüber hinaus sind GANs sehr rechenaufwändig und erfordern leistungsstarke Hardware, um effizient trainiert zu werden.

### 2.3.3 Diffusionsmodelle

Diffusionsmodelle haben in den letzten Jahren zu einem enormen Fortschritt in der Bildgenerierung, insbesondere der Text-to-Image (T2I)-Generierung, geführt. Diese Modelle basieren auf dem Konzept der Diffusion, das aus der Physik stammt. Es beschreibt den Prozess der langsamen Vermischung von Teilchen oder Informationen über die Zeit. Im maschinellen Lernen fand das Konzept erstmals in (Sohl-Dickstein et al., 2015) Anwendung, mit der Idee, die Struktur von Daten durch Hinzufügen von Rauschen schrittweise aufzulösen und anschließend ein Modell darauf zu trainieren, das ursprüngliche Bild zu rekonstruieren. Seitdem haben sich Diffusionsmodelle als eine neue Klasse von generativen Deep-Learning-Modellen etabliert, die in der Lage sind, noch realistischere Bilder zu generieren als GANs.

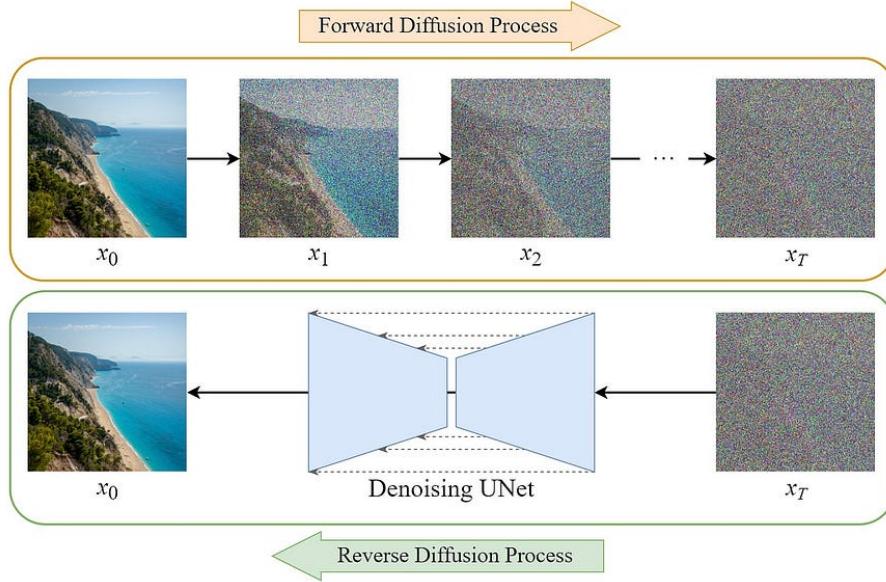


Abbildung 2.6: Darstellung der Vorwärts- und Rückwärtsdiffusion in einem Diffusionsmodell.

Das Training von Diffusionsmodelle teilt sich in zwei Phasen auf, die Vorwärts- und die Rückwärtsdiffusion, welche beide als Markov-Ketten modelliert werden können. Markov-Ketten sind stochastische Prozesse, bei denen die zukünftige Entwicklung eines Systems nur von seinem aktuellen Zustand abhängt. Im Bezug auf Diffusionsmodelle repräsentiert jeder Schritt in der Markov-Kette einen Zeitschritt  $t$  und die Zustände  $x^{(t)}$  sind die Bilder zu diesem Zeitpunkt.

In der Vorwärtsdiffusion wird ein Bild schrittweise durch ein Modell, das Rauschen hinzufügt, in ein verrauschtes Bild umgewandelt. Die Wahrscheinlichkeitsdichte des verrauschten Bildes wird durch die Produktregel der bedingten Wahrscheinlichkeiten berechnet:

$$q(x^{(0 \dots T)}) = q(x^{(0)}) \prod_{t=1}^T q(x^{(t)} | x^{(t-1)}) \quad (2.9)$$

In der Rückwärtsdiffusion wird das Modell darauf trainiert, das verrauschte Bild schrittweise in das ursprüngliche Bild zurückzuwandeln. Die Wahrscheinlichkeitsdichte des ursprünglichen Bildes wird durch die Produktregel der bedingten Wahrscheinlichkeiten berechnet:

$$p(x^{(0 \dots T)}) = p(x^{(T)}) \prod_{t=1}^T p(x^{(t-1)} | x^{(t)}) \quad (2.10)$$

Als Verlustfunktion wird die negative Log-Likelihood verwendet, die die Differenz zwischen der tatsächlichen und der modellierten Wahrscheinlichkeitsdichte misst:

$$Loss = -\log p(x^{(0 \dots T)}) \quad (2.11)$$

Anders als bei GANs gibt es keine direkten adversarialen Optimierungsmechanismen, die zu einem Ungleichgewicht führen können. Es wird stattdessen explizit die Wahrscheinlichkeitsdichte zwischen den realen Daten und den erzeugten Daten minimiert, was zu einem robusteren und stabileren Trainingsprozess führt.

Eine entscheidende Weiterentwicklung der Diffusionsmodelle war die Text-Konditionierung des generativen Prozesses. In (Ramesh et al., 2022) wurde DALL-E 2 vorgestellt, ein Diffusionsmodell, das in der Lage ist, Bilder aus Textbeschreibungen zu generieren. DALL-E 2 verwendet ein Transformer-Modell, um die Textbeschreibungen in eine latente Repräsentation zu kodieren, die dann als Eingabe für den Diffusionsprozess dient. Auf diese Weise können realistische Bilder erzeugt werden, die den Textbeschreibungen entsprechen.

Ein besonders einflussreiches Diffusionsmodell ist **Stable Diffusion** (Rombach et al., 2022). Die entscheidende Weiterentwicklung von Stable Diffusion liegt darin, dass die Diffusionsprozesse jeweils nur in einem niedrigdimensionalen latenten Raum stattfinden, bevor die Darstellungen wieder in hochauflösende Bilder umgewandelt werden (siehe Abbildung 2.7). Dies ermöglicht es, die Komplexität des Modells zu reduzieren und gleichzeitig realistische Bilder zu generieren.

### 2.3.4 DA-Fusion

In (Trabucco et al., 2023) wird DA-Fusion, eine auf Stable Diffusion basierende Methode zur Datenaugmentation vorgestellt, die für diese Arbeit von besonderem Interesse ist. Der traditionelle Ansatz der Datenaugmentation, wie in Abschnitt 2.1.5 beschrieben, hat sich als effektiv erwiesen, um die Generalisierungsfähigkeit von Modellen zu verbessern. Allerdings erfordert dieser Ansatz auch eine gute Intuition in Bezug auf den verwendeten Datensatz, um zu vermeiden, dass Transformationen gewählt werden, durch die Informationen verloren gehen, die für die Aufgabe des zu trainierenden Modells wichtig sind. Wenn beispielsweise Farbinformationen für die Klassifizierung von Blumen wichtig sind, könnte die Datenaugmentation durch zufällige Farbänderungen die Leistung des Modells verschlechtern. Ein weiteres Beispiel sind Objekte, die klein im Bild sind und durch zufällige Ausschnitte des Bildes aus der Sicht des Modells verschwinden können. DA-Fusion hingegen nutzt das Wissen eines vortrainierten Diffusionsmodelle, um den Bildinhalt semantisch zu verstehen und automatisch neue, realistische Variationen zu generieren.

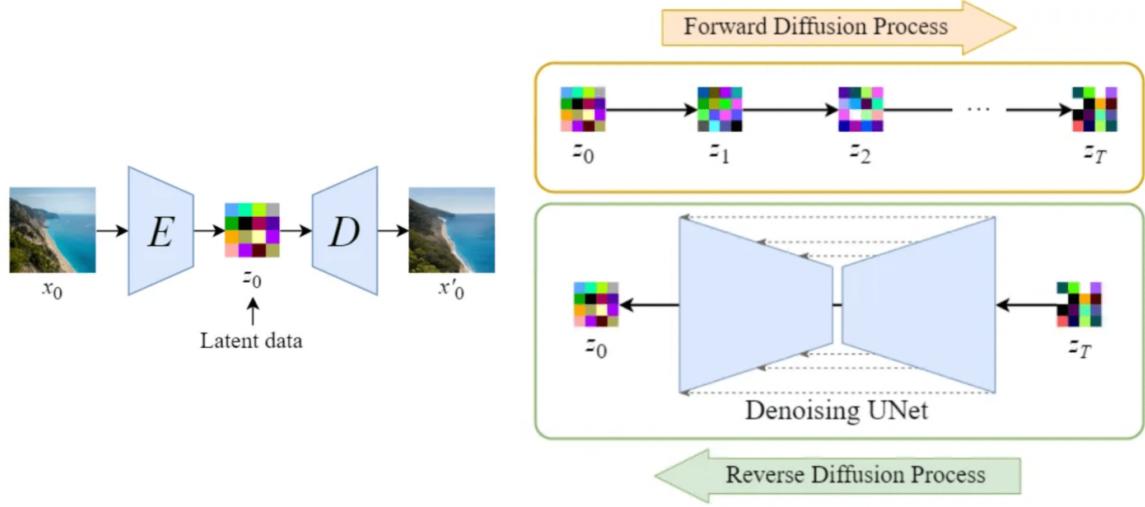


Abbildung 2.7: Darstellung des Vorwärts- und Rückwärtsdiffusion in Stable Diffusion:  
Die Diffusionsprozesse werden auf die latenten Darstellungen der Bilder angewendet (rechts), welche vorher mit einem VAE erstellt wurden (links).

Es wird zunächst die Methode Textual Inversion aus (Gal et al., 2022) angewendet, um ein vortrainiertes Stable Diffusion-Modell auf den gegebenen Datensatz feinabzustimmen. Dazu wird für jedes Konzept bzw. für jede Klasse ein neues Text-Embedding  $y$  als Platzhalter in das Modell integriert, das unter Verwendung von Trainings-Prompts wie „a photo of a  $< y >$ “ und den zugehörigen Bilddaten trainiert wird. Entscheidend ist hier, dass nicht das ganze Diffusionsmodell neu trainiert wird, sondern lediglich neue Wörter erlernt werden, welche die spezifischen Konzepte repräsentieren, sodass sich bei der Bildgenerierung weiterhin auf das vortrainierte semantische Wissen des Modells gestützt werden kann.

Anschließend können die Bilder augmentiert werden, indem ihnen eine geringe Menge an Rauschen hinzugefügt wird, welches dann durch das feinabgestimmte Modell wieder entfernt werden soll. Hier kommen die selben Text-Prompts zum Einsatz. Auf diese Weise müssen keine völlig neuen Bilder generiert werden, denn die grundlegende Struktur wird durch die ursprünglichen Bilder vorgegeben.

Ein Vorteil von DA-Fusion ist die Möglichkeit, den Grad der Augmentation durch die Wahl des Insertion Timesteps zu steuern. Der Insertion Timestep bestimmt, wie weit in den Diffusionsprozess das Bild eingefügt wird und wie stark es dafür vorher verrauscht werden muss. Ein niedriger Timestep führt zu stärkeren Augmentationen, während ein hoher Timestep subtilere Variationen erzeugt.



Abbildung 2.8: Vergleich zwischen semantischen Augmentationen aus Baseline-Methode und DA-Fusion (Trabucco et al., 2023).

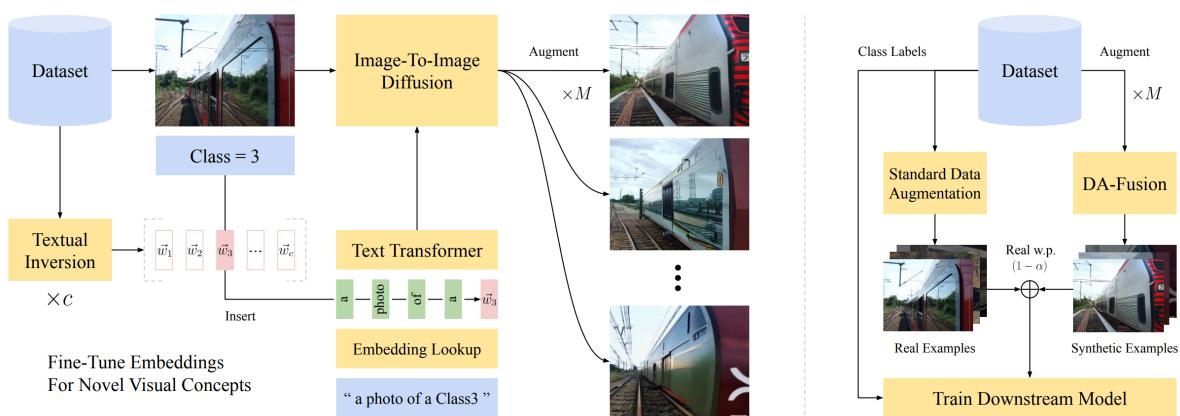


Abbildung 2.9: Überblick über den Prozess zur Datenaugmentation mit DA-Fusion (Trabucco et al., 2023).

## 2.4 Klassifikation von Gebrauchsgegenständen für die Recyclingwirtschaft

Der Anwendungsfall, auf den sich diese Arbeit bezieht, ist die Klassifikation verschiedener industrieller Objekte und Gebrauchsgegenstände für die Recyclingwirtschaft. Grundlage dafür ist das Forschungsprojekt „Sensorische Erfassung, automatisierte Identifikation und Bewertung von Altteilen anhand von Produktdaten sowie Informationen über bisherige Lieferungen“ (EIBA), das im Rahmen der Fördermaßnahme „Ressourceneffiziente Kreislaufwirtschaft – Innovative Produktkreisläufe (ReziProK)“ des Bundesministeriums für Bildung und Forschung entstand (Wagner, 2022). Neben dem Fraunhofer-IPK waren auch die Techni-

sche Universität Berlin, die Deutsche Akademie der Technikwissenschaften (acatech), sowie die Circular Economy Solutions GmbH beteiligt.

Das Ziel des Projekts war die Entwicklung eines KI-gestützten Systems, welches Daten aus verschiedenen digitalen Sensoren, sowie Kontextdaten aus dem Geschäftsprozess verarbeitet, um bei der Identifikation, Inspektion und Sortierung von Altteilen zu unterstützen. Dabei betont das Projekt die Rolle der KI zur *Unterstützung* des Menschen, nicht zur vollständigen Automatisierung (Wagner, 2022). In Abbildung 2.10 wird veranschaulicht, wie sich das System in die genannten Prozesse integriert.

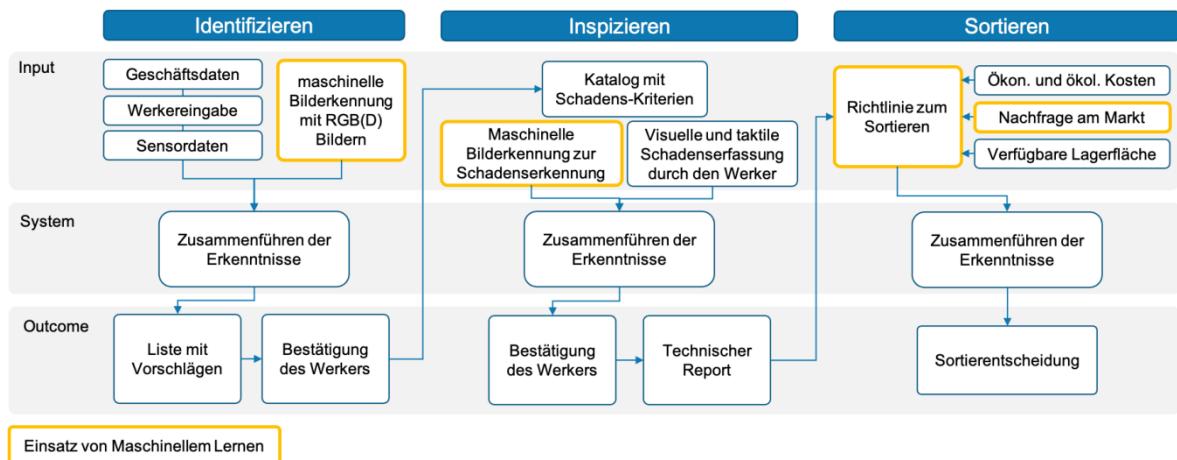


Abbildung 2.10: Integration des in EIBA entwickelten KI-gestützten Systems in den Prozess der Identifikation, Inspektion und Sortierung von Altteilen (Wagner, 2022).

Viele der Altteile sind sehr ähnlich und erfordern Expertenwissen, um sie korrekt zu identifizieren und den Beschädigungsgrad zu bewerten. Die Beschriftungen und Barcodes sind oft nicht mehr lesbar, sodass die Identifikation rein visuell erfolgen muss (Wagner, 2022). Hier setzt das KI-System an, um den Menschen zu unterstützen. Es werden dafür multimodale Sensordaten verwendet, unter anderem Tiefenkameras (RGB-D) und eine Waage zur Erfassung des Gewichts zum Einsatz. In Abbildung 2.11 ist die Mensch-Maschine-Schnittstelle einer mobilen Teststation des Systems dargestellt.

Das System wurde am Beispiel von gebrauchten Fahrzeugteilen entwickelt und konnte bei Tests basierend auf Bilddaten von ca. 1.400 Altteilen über 98% Accuracy erreichen (ReziProK, n. d.). Da die Tests jedoch unter Laborbedingungen stattfanden, wird weiterhin nach Möglichkeiten gesucht, die Generalisierungsfähigkeit des Modells zu verbessern. Insbesondere die Klassifikation von Objekten in verschiedenen Gebrauchszuständen, wie z.B. beschädigte oder verschmutzte Teile, stellt eine Herausforderung dar.

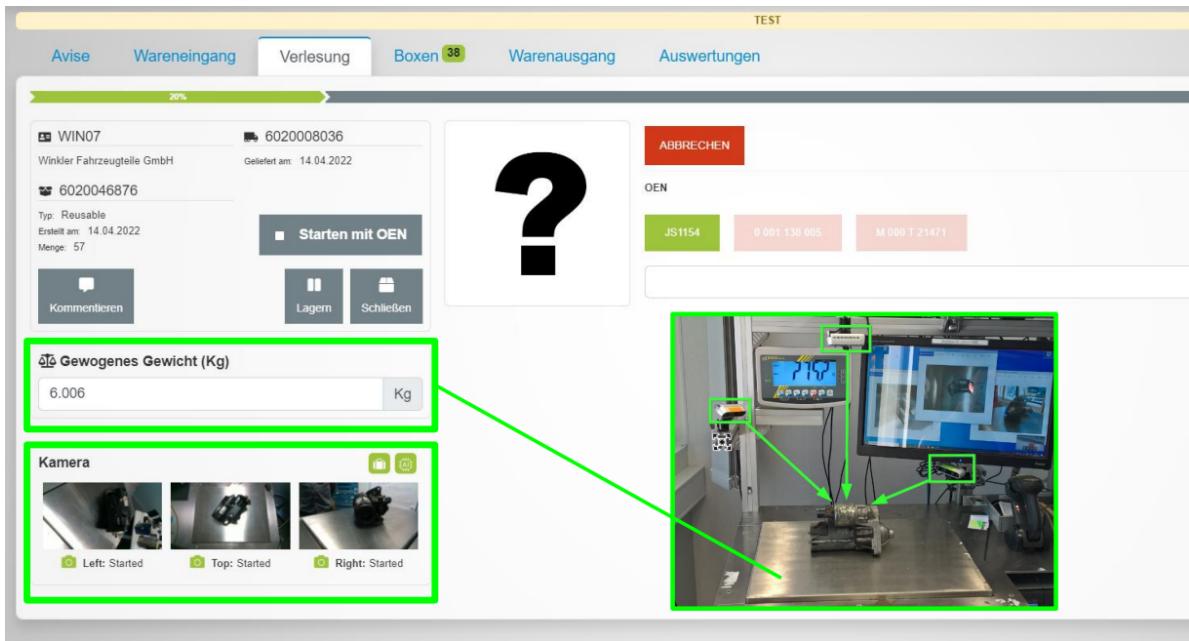


Abbildung 2.11: Die Mensch-Maschine-Schnittstelle des Systems (Wagner, 2022).

In den vorherigen Abschnitten wurden verschiedene Methoden zur Verbesserung der Generalisierungsfähigkeit und Robustheit von Modellen vorgestellt. Sowohl das Contrastive Learning als auch die synthetische Datengenerierung bzw. Datenaugmentation konnten jeweils vielversprechende Ergebnisse erzielen. In dieser Arbeit wird nun ein Ansatz zur Kombination beider Methoden vorgestellt, welcher eine neuartige Verwendung von synthetischen Daten im Contrastive Learning ermöglichen könnte.

#### 2.4.1 Herausforderungen bei der Generierung synthetischer Daten

Zunächst soll auf eine Reihe von Herausforderungen bei der Generierung synthetischer Daten aufmerksam gemacht werden, insbesondere im Kontext des vorgestellten Anwendungsfalls. Für herkömmliche Methoden zur Bildgenerierung, wie GANs oder Diffusionsmodelle, ergeben sich hier nämlich spezielle Herausforderungen:

- Die Objekte weisen teilweise eine sehr hohe **Komplexität** auf, etwa bei Motoren oder Generatoren mit vielen Details. Auch moderne Methoden zur Bildgenerierung können daran scheitern, diese Details korrekt zu erlernen und zu reproduzieren –insbesondere, wenn nur wenige Beispielbilder gegeben sind.
- Es gibt oft nur **feine Unterschiede** zwischen den Klassen, die es zu berücksichtigen gilt. Sind die generierten Daten nicht akkurat genug, kann es zu Fehlklassifikationen kommen.

- Auf Grund des Multiview-Setups haben die Bilder nur **wenig Variation**, vor allem in den Hintergründen. Auch die Objekte selbst sind zwar aus verschiedenen Perspektiven aufgenommen, bieten aber pro Klasse nicht viel Variation in Bezug auf die Beschaffenheit, die Farbe, usw.

Es ergeben sich also hohe Anforderungen an die Generierung der synthetischen Daten. Einerseits muss die Genauigkeit der generierten Daten gewährleistet sein, um die Klassifikation der Modelle nicht zu beeinträchtigen. Trotzdem muss genügend Variation ermöglicht werden, um die Generalisierungsfähigkeit der Modelle zu verbessern.

### **2.4.2 Synthetische Daten als negativ-Beispiele im Contrastive Learning**

Aus den spezifischen Herausforderungen bei der Generierung synthetischer Daten, zusammen mit den Besonderheiten des Contrastive Learning, ergibt sich eine interessante Forschungslücke: Ist es möglich, auch aus *mangelhaften* synthetischen Daten zu lernen, wenn sie ausschließlich als negativ-Beispiele im Contrastive Learning verwendet werden? Genauer, lässt sich so die Leistung eines Modells bei der Klassifikation von echten Daten verbessern und gleichzeitig die Robustheit gegenüber OOD-Daten erhöhen?

Die bisherigen Erfolge von Contrastive Learning, insbesondere von SimCLR, zeigen, dass das Modell besonders von Hard Negatives profitiert (Chen et al., 2020), also von unähnlichen Beispielen, die aber nur schwer zu unterscheiden sind. Auch (Jiang et al., 2024) baut auf dieser Erkenntnis auf und führt eine Strategie zum Hard Negative Sampling im Supervised Contrastive Learning ein, jedoch ohne Verwendung von synthetischen Daten. Was wäre also, wenn anstatt die Negativ-Beispiele aus der Nähe im Darstellungsraum auszuwählen, Near OOD-Augmentationen der Anchor-Klasse verwendet werden?

Um effektiv als Hard Negatives zu wirken, dürften die synthetischen Daten also nicht zu weit entfernt von den echten Daten sein. Sie könnten deshalb als *Near OOD*-Beispiele bezeichnet werden, wobei sie noch OOD genug sein müssen, um die Distanz zwischen den ID- und OOD-Daten zu maximieren. Ob sich diese synthetischen Near OOD-Daten tatsächlich als gute Hard Negatives herausstellen, wird in dieser Arbeit untersucht.

### **2.4.3 Integration von DA-Fusion und Supervised Contrastive Learning**

...

# 3 Methodisches Vorgehen

In diesem Kapitel wird das methodische Vorgehen der Arbeit beschrieben. Als Basis für die Untersuchung der Forschungsfragen wird zunächst der MVIP-Datensatz vorgestellt, auf dem die Experimente durchgeführt werden. Anschließend wird die Implementierung der Modelle DA-Fusion und Supervised Contrastive Learning erläutert, sowie die Herangehensweise zur Generierung synthetischer Daten mit DA-Fusion und die Trainings- und Testdurchläufe mit Supervised Contrastive Learning definiert. Zuletzt werden die Evaluationsmethoden und Metriken vorgestellt, die zur Auswertung der Experimente verwendet werden.

## 3.1 MVIP-Datensatz

Grundlage der Forschungsarbeit ist der im Rahmen des EIBA-Projekts entstandene MVIP-Datensatz (Koch et al., 2023), wobei MVIP für *Multi-View Industrial Parts* steht. Er enthält 308 Klassen, welche wiederum in 18 verschiedene Oberklassen (Super Classes) eingeteilt sind. Insgesamt gibt es etwa 71.276 Sets an Bildern, die jeweils RGB- und Tiefendaten, sowie Segmentierungsmasken enthalten.

Die Bilddaten stammen aus Intel RealSense D435 und D415 Tiefenkameras, die die Objekte gleichzeitig aus verschiedenen Perspektiven aufnehmen. Darüber hinaus gibt es auch Metadaten, etwa zum Gewicht des Objekts, oder Beschreibungen in natürlicher Sprache durch verschiedene Stichwörter („StarterMotor“, „Used“, „Rusty“, usw.).

### 3.1.1 Teildatensatz

Da für die Experimente relativ viele Trainings- und Testdurchläufe vorgesehen waren, war es notwendig, einen Teildatensatz auszuwählen, um die Rechenzeit zu reduzieren.

Konkret wurden zufällig 20 Klassen aus der Oberklasse „CarComponent“ ausgewählt. Für die Experimente wurden nur die RGB-Bilddaten verwendet, wobei die Segmentierungsmasken in der Vorverarbeitung zum Einsatz kommen (siehe Abschnitt 3.1.2).

Insgesamt enthält der Teildatensatz ... Bilder. In Tabelle 3.1 sind die Klassen im Detail aufgeführt.

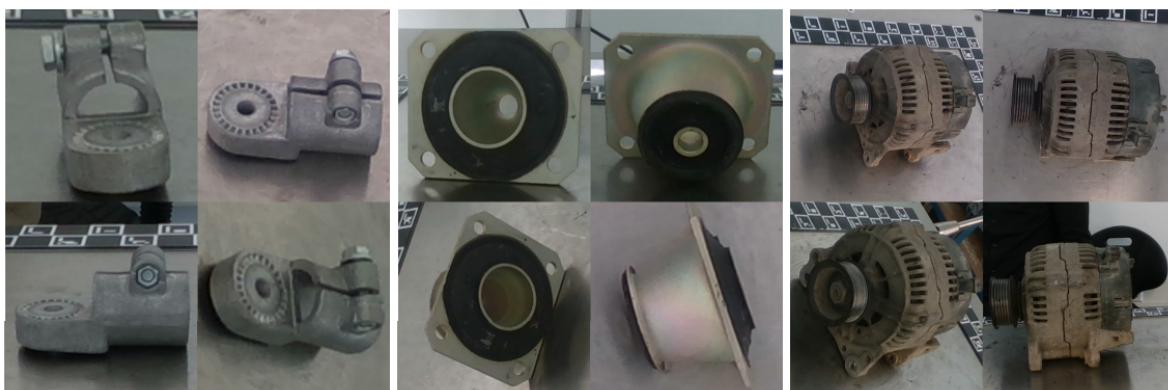


Abbildung 3.1: Beispielbilder aus dem MVIP-Datensatz, die auf die Region of Interest (ROI) zugeschnitten wurden (Koch et al., 2023).

Tabelle 3.1: Auswahl der 20 Klassen aus der Oberklasse „CarComponent“ für den MVIP-Teildatensatz.

Klasse	Beschreibungen	Anzahl Bilder
1 ...	...	...
2 ...	...	...
3 ...	...	...
4 ...	...	...
5 ...	...	...
6 ...	...	...
7 ...	...	...
8 ...	...	...
9 ...	...	...
10 ...	...	...
11 ...	...	...
12 ...	...	...
13 ...	...	...
14 ...	...	...
15 ...	...	...
16 ...	...	...
17 ...	...	...
18 ...	...	...
19 ...	...	...
20 ...	...	...

### 3.1.2 Vorverarbeitung

Die Vorverarbeitung der Bilder unterscheidet sich nur leicht zwischen den beiden Modellen DA-Fusion und Supervised Contrastive Learning. In beiden Fällen werden die Bilder auf die Region of Interest (ROI) zugeschnitten, um den Fokus auf die Objekte zu legen und den Hintergrund zu minimieren. Dazu werden die Segmentierungsmasken verwendet, um die Bounding Box der Objekte zu bestimmen und die Bilder entsprechend zuzuschneiden. Die Ausgabe ist ein quadratisches Bild, das die Objekte in der Mitte enthält.

Zusätzlich werden verschiedene „klassische“ Augmentationen angewendet, um die Daten zu erweitern und die Modelle robuster zu machen. Dazu gehören z.B. Rotation, ColorJitter und Normalisierung. Die Parameter für die Augmentationen wurden eher konservativ gewählt, damit die feinen Unterschiede zwischen den Klassen nicht verwischt werden.

## 3.2 Implementierung

Zur Vorbereitung und Durchführung der Experimente wurde seitens des Fraunhofer-IPK ein Zugang zu einem leistungsstarken Rechner bereitgestellt, der über zwei NVIDIA-Grafikkarten mit CUDA-Unterstützung<sup>1</sup> verfügt. Die Implementierung der Modelle und Experimente erfolgte in Python 3.7 unter Verwendung der Bibliotheken PyTorch 1.12.1, Torchvision 0.13.1 und weitere.

Dabei stützt sich diese Arbeit größtenteils auf die Implementierungen von DA-Fusion aus (Trabucco, 2024) und Supervised Contrastive Learning aus (Tian, 2023). Die grundlegende Funktionsweise dieser Implementierungen, sowie die Anpassungen, die im Rahmen des untersuchten Ansatzes vorgenommen wurden, sollen nun genauer erläutert werden.

### 3.2.1 DA-Fusion

Die Implementierung von DA-Fusion kann weitgehend unverändert angewendet werden, um synthetische Daten für den MVIP-Teildatensatz zu generieren. Allerdings musste dieser zunächst als eigene Klasse mit den Methoden `get_image_by_idx(idx)` und `get_label_by_idx(idx)` implementiert werden (siehe Anhang ??). Insbesondere musste die Auswahl der 20 „Car-Component“ Klassen sichergestellt werden, wofür die JSON-Dateien der Objekt-Metadaten ausgelesen werden. Anschließend wird über eine Methode das Laden der entsprechenden Bilder und Masken ermöglicht. Der Rest der Klasse entspricht dem Aufbau der anderen Datensatz-Klassen, die in dem Repository zu finden sind.

---

<sup>1</sup>Compute Unified Device Architecture, eine API-Technologie von NVIDIA für parallele Berechnungen auf Grafikkarten

Mit der implementierten Klasse kann das Skript `fine_tune_upstream.py` aufgerufen werden, um mit den entsprechenden Argumenten zur Wahl des Datensatzes und der Hyperparameter (siehe Abschnitt 3.3.1) Textual Inversion zum Finetuning eines vortrainierten Stable Diffusion-Modells anzuwenden. Dabei werden Text-Prompts verwendet, die zuvor in einer Liste definiert wurden. Für die Experimente wurde die Liste leicht angepasst, vor allem um weniger angemessene Formulierungen wie "a rendering of a {}" und "a photo of a cool {}" zu entfernen.

Während des Trainings werden außerdem mit der Funktion `log_validation` Validierungsbilder generiert, welche den `validation_prompt` verwenden, der als Argument übergeben wird. Diese Bilder sind nicht mit den später generierten Augmentationen vergleichbar, da sie von Grund auf neu generiert werden, bieten aber eine Möglichkeit, den Verlauf des Trainings zu überwachen.

Die gelernten Text-Embeddings werden als .pt-Dateien gespeichert und anschließend mit dem Skript `aggregate_embeddings.py` zusammengeführt, um ein klassenagnostisches Template zu erstellen, das im nachfolgenden Schritt verwendet wird.

Um schließlich die Augmentationen zu generieren, wird das Skript `generate_augmentations.py` ausgeführt. Hier wird eine Stable Diffusion Image-to-Image-Pipeline<sup>2</sup> verwendet, um den Bildern des Datensatzes Rauschen hinzuzufügen und unter Konditionierung auf den gelernten Text-Embeddings wiederherzustellen. Der genaue Prompt kann als Argument übergeben werden, wobei "a photo of a {}" als Standardwert verwendet wird. Das `strength`-Argument entscheidet, wie viel Rauschen hinzugefügt wird und an welchen Timestep das Bild in das Denoising-Modul eingesetzt wird, wodurch mehr oder weniger stark veränderte Bilder entstehen.

### 3.2.2 Supervised Contrastive Learning

Die Implementierung von Supervised Contrastive Learning verwendet zwei Trainings-Skripte; eins für das Pre-Training der latenten Repräsentationen und eins für die lineare Klassifikation der Repräsentationen.

Zunächst wird die Klasse für den MVIP-Teildatensatz, die für die Implementierung von DA-Fusion verwendet wurde, größtenteils übernommen und in die Trainingsskripte integriert. Die Klasse musste nun allerdings um einige Parameter und Funktionen erweitert werden, um die Verwendung der synthetischen Augmentationen aus DA-Fusion zu konfigurieren (siehe Anhang ??). So steuert der Parameter `aug_mode`, ob keine Augmentationen, ausschließlich ID-Augmentationen oder auch die Near OOD-Augmentationen verwendet werden sollen. Über `aug_dir_id` und `aug_dir_oob` werden die Pfade zu den entsprechenden Augmentationen

---

<sup>2</sup>Die Pipeline verwendet das **SDEdit**-Framework, das in (Meng et al., 2022) vorgeschlagen wurde.

angegeben. Außerdem wurde mit den Parametern `aug_ex_id` und `aug_ex_ood` eine Einstellung für die Anzahl der verwendeten ID- und OOD-Augmentationen pro Klasse implementiert.

Das Pre-Training der latenten Repräsentationen erfolgt mit dem Skript `main_supcon.py`, indem ein klassischer Trainingsdurchlauf implementiert wird. Das Modell wird dabei mit einem ResNet-Backbone und einem Projection Head initialisiert. Im Training generiert es die latenten Repräsentationen der Bilder, die dann zusammen mit den Labels in die kontrastive Verlustfunktion gegeben werden. Diese berechnet zunächst eine kontrastive Maske, die die positiven und negativen Paare definiert. Anschließend werden die Kosinus-Ähnlichkeiten der Paare berechnet und die durchschnittliche negative Log-Wahrscheinlichkeit der positiven Paare als Verlust zurückgegeben.

Für die Experimente dieser Arbeit war die Anpassung dieser Verlustfunktion entscheidend (siehe Anhang ??). Dafür wird die kontrastive Maske zunächst so gefiltert, dass keine positiven Paare mit OOD-Daten entstehen, indem diese durch ein negatives Label identifiziert wurden. Nachdem dann die Ähnlichkeitswerte berechnet wurden, werden diese nochmals maskiert, sodass aus den negativen Paaren mit OOD-Daten nur solche übrig bleiben, bei denen die OOD-Augmentation aus der selben Klasse generiert wurde, wie das ID-Bild. Die Verlustfunktion wird dann nur auf diese Paare angewendet.

Das trainierte Modell, welches im Verlauf des Pre-Trainings gespeichert wurde, wird anschließend mit dem Skript `main_linear.py` für die lineare Klassifikation verwendet. Dabei wird das Modell mit einem neuen Head initialisiert, der die latenten Repräsentationen in die Klassenlabels überführt. Das Training erfolgt dann mit einer klassischen Cross Entropy-Verlustfunktion, die die Repräsentationen auf die Klassen abbildet.

### 3.3 Versuchsaufbau

Die Untersuchung der Forschungsfragen erfolgte in zwei Schritten: Zunächst werden synthetische Daten mit DA-Fusion generiert. Anschließend werden die synthetischen Daten in den Trainings- und Testdurchläufen mit Supervised Contrastive Learning verwendet, um die Auswirkungen der Augmentationen auf die Klassifikation zu evaluieren.

#### 3.3.1 Synthetische Datengenerierung

Es wurden mit DA-Fusion zwei unterschiedliche Sätze von Augmentationen generiert: ID-Augmentationen und Near OOD-Augmentationen.

Für beide Sätze wurde das Modell `CompVis/stable-diffusion-v1-4`<sup>3</sup> verwendet, das mit dem

---

<sup>3</sup>Ein vortrainiertes HuggingFace-Modell: <https://huggingface.co/CompVis/stable-diffusion-v1-4>

Token "motor" initialisiert wurde. Dabei wurden 16 Vektoren pro Token verwendet. Das Fine-tuning erfolgte mit einer Batch Size von 16, einer Lernrate von 0.0005 und einer konstanten Lernrate mit Warmup über 150 Schritte. Es wurde Mixed Precision mit FP16 verwendet und das Training wurde nach 1000 Schritten beendet. Die erlernten Text-Embeddings wurden für beide Sätze verwendet, um die Augmentationen zu generieren.

Die Generierung erfolgte mit der Real Guidance Pipeline und einer Guidance Scale von 15. Es wurden 16 Beispiele pro Klasse zur Augmentation herangezogen und für jedes Beispiel vier synthetische Bilder generiert. Die Segmentierungsmasken wurden verwendet, um die Augmentation auf den Bereich des Objekts zu begrenzen. Die ID- und Near OOD-Augmentationen unterscheiden sich nur durch den strength-Parameter, wobei für die ID-Augmentationen ein Wert von 0.2 und für die Near OOD-Augmentationen ein Wert von 0.5 verwendet wurde.

### 3.3.2 Trainings- und Testdurchläufe

Die Trainings- und Testdurchläufe mit Supervised Contrastive Learning sind in drei Versuchsreihen unterteilt. In jedem Versuch wird zunächst das Pre-Training der latenten Repräsentationen durchgeführt, gefolgt von der linearen Klassifikation der Repräsentationen. Die Versuchsreihen unterscheiden sich in der Verwendung der synthetischen Augmentationen:

1. Nur reale Daten werden verwendet, sowohl für das Pre-Training als auch für das Training des Klassifikators.
2. Neben den realen Daten werden auch ID-Augmentationen verwendet, sowohl für das Pre-Training als auch für das Training des Klassifikators.
3. Wie Versuch 2, aber im Pre-Training werden zusätzlich Near OOD-Augmentationen als harte negativ-Beispiele verwendet, wie in Abschnitt [2.4.3](#) beschrieben.

Für das Pre-Training wurde eine Batch Size von 16, eine Lernrate von 0.001 mit Kosinus-Annealing und einer Dauer von 110 Epochen verwendet. Die Bilddaten wurden im Rahmen der Augmentation auf eine Größe von 224x224 Pixeln zugeschnitten und normalisiert. Bei der linearen Klassifikation sind die Hyperparameter identisch, jedoch mit einer Dauer von 25 Epochen.

Nach den Trainingsdurchläufen werden die Modelle auf den (echten) Testdaten evaluiert.

## 3.4 Evaluationsmethoden und Metriken

Die Qualität der synthetischen Daten, die mit DA-Fusion generiert wurden, werden durch einfache visuelle Inspektion selbst überprüft. Bei den ID-Augmentationen wird darauf geachtet, dass die Objekte in den Bildern in ihrer Form möglichst unverändert bleiben, während die Farben und Texturen variieren können und sollen. Bei den OOD-Augmentationen wird darauf geachtet, dass die Objekte stark genug verändert werden, um nicht mehr als Beispiele für die ID-Klassen erkannt zu werden, während sie gleichzeitig noch genug Ähnlichkeit aufweisen, um herausfordernde negative Beispiele zu sein.

Die Ergebnisse der drei Versuchsreihen werden ausgewertet, indem folgende Metriken beim Test der linearen Klassifikation auf den Testdaten gemessen werden:

- **Top-1 Accuracy:** Der Anteil der korrekt klassifizierten Beispiele.
- **ID-Confidence:** Die durchschnittliche Konfidenz des Klassifikators auf den ID-Daten.
- **OOD-Confidence:** Die durchschnittliche Konfidenz des Klassifikators auf den OOD-Daten.

# **4 Ergebnisse**

Dieses Kapitel präsentiert die Ergebnisse der Arbeit. Es wird auf die generierten synthetischen Daten eingegangen und die Trainings- und Testergebnisse der Modelle beschrieben. Anschließend wird die Klassifikations-Performance der Modelle verglichen und die Out-of-Distribution-Detektion analysiert.

## **4.1 Die generierten synthetischen Daten**

...

### **4.1.1 In-Distribution**

...

### **4.1.2 Near Out-of-Distribution**

...

## **4.2 Trainings- und Testergebnisse**

...

### **4.2.1 Contrastive Pre-Training**

...

...

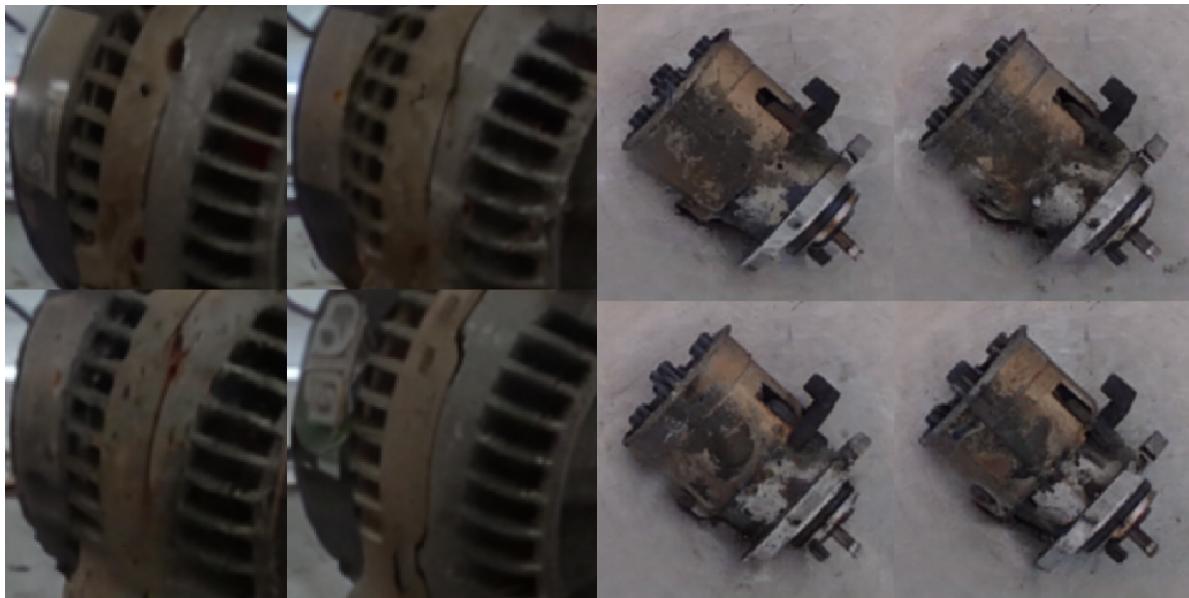


Abbildung 4.1: Vergrößerte Ausschnitte der synthetischen In-Distribution-Daten.

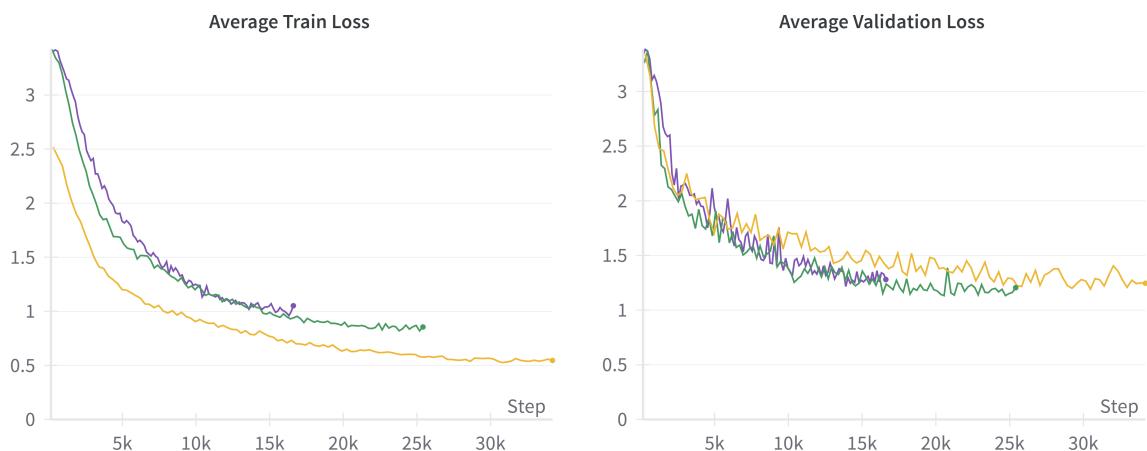


Abbildung 4.2: Beispieltext

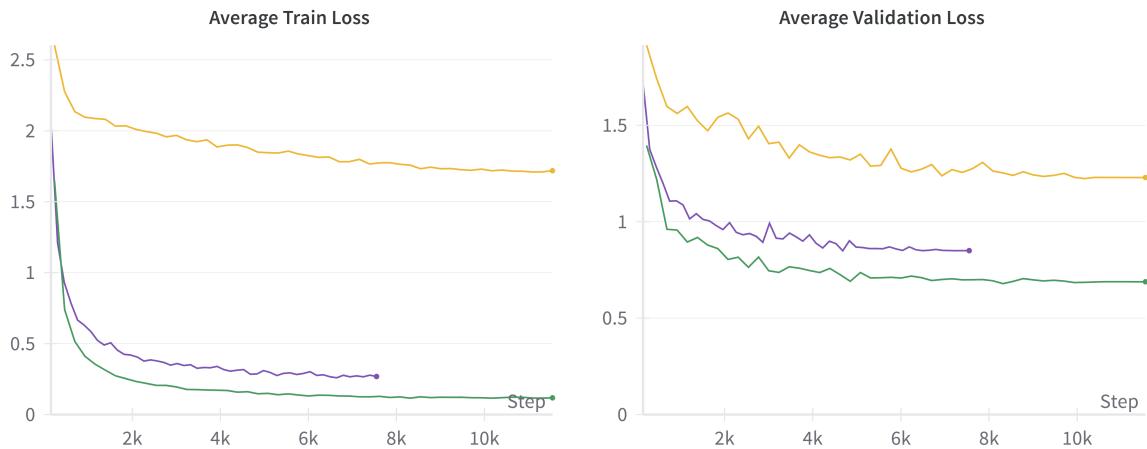


Abbildung 4.3: Beispieltext 1

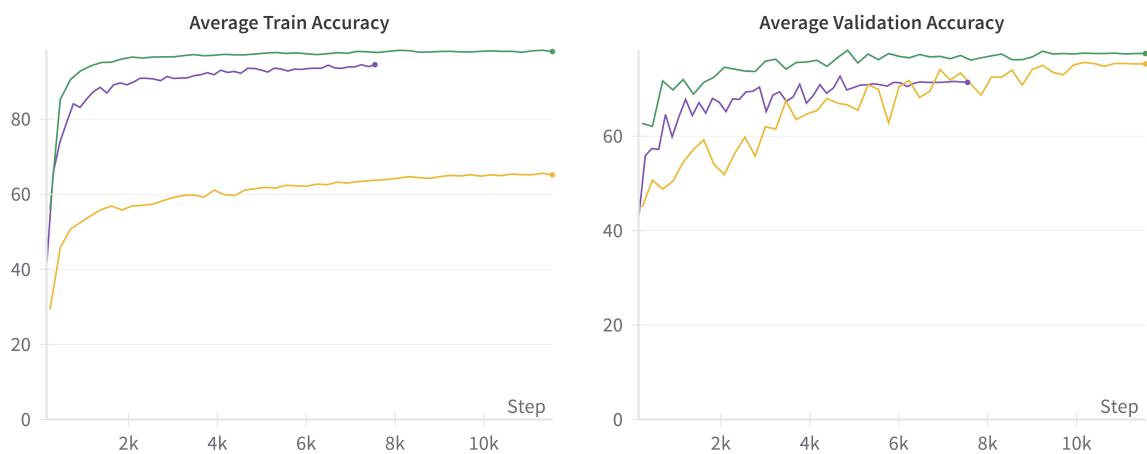


Abbildung 4.4: Beispieltext 2

#### 4.2.2 Lineare Klassifikation

...

...

### 4.3 Vergleich der Ergebnisse

...

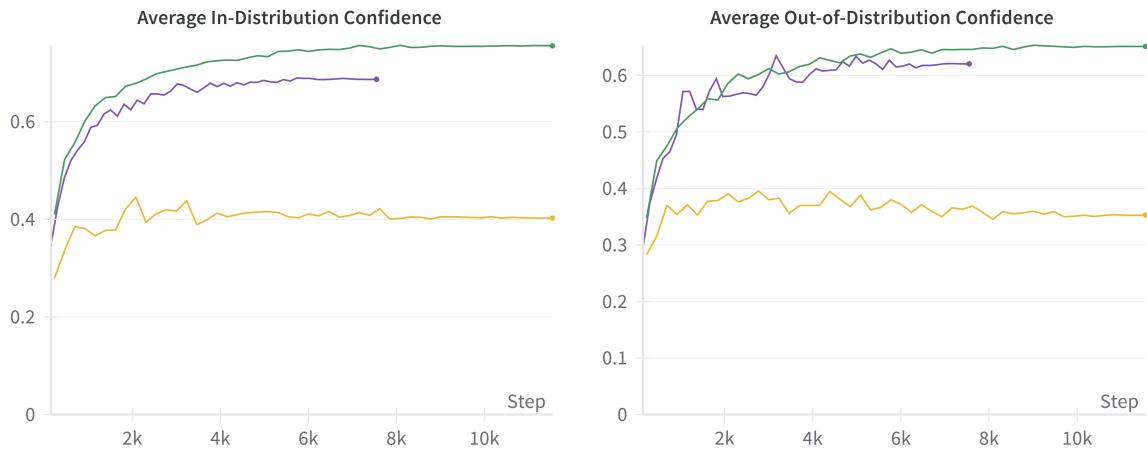


Abbildung 4.5: Beispieltext 3

Tabelle 4.1: Testergebnisse der linearen Klassifikation.

Versuch	Accuracy (%)	ID-/OOD-Confidence (Differenz)
1	71.4	0.69 / 0.62 (0.07)
2	77.5	0.76 / 0.65 (0.11)
3	75.3	0.40 / 0.35 (0.05)

### 4.3.1 Mit und ohne In-Distribution-Augmentationen

...

### 4.3.2 Mit und ohne Near Out-of-Distribution-Augmentationen

...

# **5 Diskussion**

...

## **5.1 Eignung von DA-Fusion für die synthetische Datengenerierung**

...

## **5.2 Wirksamkeit von Near Out-of-Distribution-Augmentationen im Supervised Contrastive Learning**

...

# **6 Fazit**

...

## **6.1 Zusammenfassung der wichtigsten Erkenntnisse**

...

## **6.2 Beantwortung der Forschungsfragen**

...

## **6.3 Ausblick und potenzielle Weiterentwicklungen**

...

# Literatur

- Burgmer, C. (2005, Juli). *Schema eines künstlichen Neurons* [Lizenziert unter der Creative Commons Attribution-Share Alike 3.0 Unported Lizenz: <https://creativecommons.org/licenses/by-sa/3.0/deed.en>]. <https://commons.wikimedia.org/wiki/File:ArtificialNeuronModel.png>
- Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A Simple Framework for Contrastive Learning of Visual Representations.
- Cun, Y. L., Boser, B., Denker, J. S., Howard, R. E., Hubbard, W., Jackel, L. D., & Henderson, D. (1990). Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems 2* (S. 396–404). Morgan Kaufmann Publishers Inc.
- Foster, D. (2020). *Generatives Deep Learning*. O'Reilly.
- Gal, R., Alaluf, Y., Atzmon, Y., Patashnik, O., Bermano, A. H., Chechik, G., & Cohen-Or, D. (2022). An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion. <https://arxiv.org/abs/2208.01618>
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Networks. <https://arxiv.org/abs/1406.2661>
- Google for Developers. (2022). *Overview of GAN Structure* [Lizenziert unter der Creative Commons Attribution 4.0 Lizenz: <https://creativecommons.org/licenses/by/4.0/>]. [https://developers.google.com/machine-learning/gan/gan\\_structure](https://developers.google.com/machine-learning/gan/gan_structure)
- Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On Calibration of Modern Neural Networks. <https://arxiv.org/abs/1706.04599>
- Hendrycks, D., & Gimpel, K. (2018). A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. <https://arxiv.org/abs/1610.02136>
- Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786), 504–507. <https://doi.org/10.1126/science.1127647>
- Jiang, R., Nguyen, T., Ishwar, P., & Aeron, S. (2024). Supervised Contrastive Learning with Hard Negative Samples. <https://arxiv.org/abs/2209.00078>
- Keshtmand, N., Santos-Rodriguez, R., & Lawry, J. (2022). Understanding the properties and limitations of contrastive learning for Out-of-Distribution detection.

- Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., & Krishnan, D. (2021). Supervised Contrastive Learning.
- Kim, J., Lee, H., Chang, J., & Park, S. M. (2023). Generalized Supervised Contrastive Learning. <https://arxiv.org/abs/2206.00384>
- Kingma, D. P., & Welling, M. (2022). Auto-Encoding Variational Bayes. <https://arxiv.org/abs/1312.6114>
- Koch, P., Schlüter, M., Briese, C., & Chavan, V. (2023, November). MVIP: A Dataset for Industrial Part Recognition. <https://doi.org/http://dx.doi.org/10.24406/fordatis/300>
- Liang, S., Li, Y., & Srikant, R. (2020). Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks. <https://arxiv.org/abs/1706.02690>
- Liu, R. (2021). Understand and Improve Contrastive Learning Methods for Visual Representation: A Review. <https://arxiv.org/abs/2106.03259>
- Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.-Y., & Ermon, S. (2022). SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations. <https://arxiv.org/abs/2108.01073>
- Mitchell, T. M. (1997). *Machine Learning*. McGraw Hill.
- Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning Transferable Visual Models From Natural Language Supervision. <https://arxiv.org/abs/2103.00020>
- Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical Text-Conditional Image Generation with CLIP Latents. <https://arxiv.org/abs/2204.06125>
- ReziProK. (n. d.). *EIBA - Sensorische Erfassung, automatisierte Identifikation und Bewertung von Altteilen anhand von Produktdaten sowie Informationen über bisherige Lieferungen* [Aufgerufen: 13.09.2024]. <https://innovative-produktkreislaeufe.de/Projekte/EIBA.html>
- Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-Resolution Image Synthesis with Latent Diffusion Models. <https://arxiv.org/abs/2112.10752>
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536. <https://doi.org/10.1038/323533a0>
- Saha, S. (2018). *A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way* [Aufgerufen: 17.09.2024]. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- Shorten, C., & Khoshgoftaar, T. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6. <https://doi.org/10.1186/s40537-019-0197-0>
- Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., & Ganguli, S. (2015). Deep Unsupervised Learning using Nonequilibrium Thermodynamics. <https://arxiv.org/abs/1503.03585>
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. <http://jmlr.org/papers/v15/srivastava14a.html>
- Tian, Y. (2023). *SupContrast* [GitHub Repository]. <https://github.com/HobbitLong/SupContrast>

- Tian, Y., Fan, L., Isola, P., Chang, H., & Krishnan, D. (2023). StableRep: Synthetic Images from Text-to-Image Models Make Strong Visual Representation Learners. <https://arxiv.org/abs/2306.00984>
- Trabucco, B. (2024). *DA-Fusion* [GitHub Repository]. <https://github.com/brandonorabucco/da-fusion>
- Trabucco, B., Doherty, K., Gurinas, M., & Salakhutdinov, R. (2023). Effective Data Augmentation With Diffusion Models. <https://arxiv.org/abs/2302.07944>
- Wagner, M. (2022, Juni). *BMBF-Fördermaßnahme „Ressourceneffiziente Kreislaufwirtschaft - Innovative Produktkreisläufe (ReziProK)“* [ReziProK Transferkonferenz am 23. und 24. Juni 2022 im Tagungswerk, Berlin]. [https://innovative-produktkreislaeufe.de/Projekte/EIBA/\\_/ReziProK\\_Statuskonferenz\\_Pr%C3%A4sentation\\_EIBA\\_2022\\_06\\_23\\_final.pdf](https://innovative-produktkreislaeufe.de/Projekte/EIBA/_/ReziProK_Statuskonferenz_Pr%C3%A4sentation_EIBA_2022_06_23_final.pdf)
- Zhou, Z.-H. (2021). *Machine Learning*. Springer.

# **Anhang**

# **Eigenständigkeitserklärung**

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit mit dem Titel

**Contrastive Learning mit Stable Diffusion-basierter Datenaugmentation**

selbstständig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen habe, habe ich deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.

Hamburg, 20. September 2024