

**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 . . . . .	1
<b>2 Theoretische Grundlagen</b>	<b>2</b>
2.1 Maschinelles Lernen . . . . .	2
2.1.1 Überwachtes und unüberwachtes Lernen . . . . .	3
2.1.2 Deep Learning . . . . .	3
2.1.3 Neuronale Netze . . . . .	4
2.1.4 Out-of-Distribution Daten . . . . .	7
2.1.5 Datenaugmentation und Generalisierung . . . . .	7
2.2 Synthetische Daten . . . . .	8
2.2.1 Variational Autoencoder . . . . .	8
2.2.2 Generative Adversarial Networks . . . . .	10
2.2.3 Diffusionsmodelle . . . . .	11
2.3 Semantische Datenaugmentation mit DA-Fusion . . . . .	13
2.4 Robuste Datenrepräsentation durch Contrastive Learning . . . . .	14
2.4.1 Unsupervised Contrastive Learning . . . . .	14
2.4.2 Supervised Contrastive Learning . . . . .	15
2.5 Forschungslücke . . . . .	16
2.5.1 Herausforderungen bei der Generierung synthetischer Daten . . . . .	16
2.5.2 Synthetische Daten als negativ-Beispiele im Contrastive Learning . . . . .	16
2.5.3 Integration von DA-Fusion und Supervised Contrastive Learning . . . . .	16
<b>3 Methodisches Vorgehen</b>	<b>17</b>
3.1 Forschungsfragen und Hypothesen . . . . .	17
3.2 Datensatz . . . . .	18
3.2.1 EIBA . . . . .	18

3.2.2	Teildatensatz . . . . .	18
3.2.3	Vorverarbeitung . . . . .	19
3.3	Implementierung . . . . .	19
3.3.1	DA-Fusion . . . . .	19
3.3.2	Supervised Contrastive Learning . . . . .	19
3.4	Synthetische Datengenerierung mit DA-Fusion . . . . .	20
3.5	Trainings- und Testdurchläufe mit Supervised Contrastive Learning . . . . .	20
3.6	Evaluationsmethoden und Metriken . . . . .	20
<b>4</b>	<b>Ergebnisse</b>	<b>21</b>
4.1	Die generierten synthetischen Daten . . . . .	21
4.1.1	In-Distribution . . . . .	21
4.1.2	Near Out-of-Distribution . . . . .	21
4.2	Trainings- und Testergebnisse mit Supervised Contrastive Leraning . . . . .	22
4.2.1	Contrastive Pre-Training . . . . .	22
4.2.2	Lineare Klassifikation . . . . .	22
4.3	Vergleich der Ergebnisse mit und ohne In-Distribution-Augmentationen . . . . .	22
4.4	Vergleich der Ergebnisse mit und ohne Near Out-of-Distribution-Augmentationen als Hard Negatives . . . . .	22
<b>5</b>	<b>Diskussion</b>	<b>23</b>
5.1	Eignung von DA-Fusion für die synthetische Datengenerierung . . . . .	23
5.2	Wirksamkeit von synthetischen Near Out-of-Distribution-Daten im Supervised Contrastive Learning . . . . .	23
<b>6</b>	<b>Fazit</b>	<b>24</b>
6.1	Zusammenfassung der wichtigsten Erkenntnisse . . . . .	24
6.2	Beantwortung der Forschungsfragen . . . . .	24
6.3	Ausblick und potenzielle Weiterentwicklungen . . . . .	24
<b>Literatur</b>		<b>25</b>
<b>Anhang</b>		<b>26</b>

# Abbildungsverzeichnis

2.1	Aufbau eines Convolutional Neural Networks (CNN) . . . . .	4
2.2	Darstellung eines einfachen neuronalen Netzes (O'Shea & Nash, 2015). . . . .	5
2.3	Das McCulloch-Pitts-Modell eines Neurons. . . . .	5
2.4	Beispiele für unterschiedliche Datenaugmentationstechniken und Kombinationen. (<empty citation>) . . . . .	8
2.5	Überblick über die GAN-Struktur. Quelle: Google for Developers . . . . .	10
2.6	Vergleich zwischen semantischen Augmentationen aus Baseline-Methode und DA-Fusion (Trabucco et al., 2023). . . . .	13
4.1	Beispieltext . . . . .	21

# **Tabellenverzeichnis**

# **1 Einleitung**

## **1.1 Motivation**

...

## **1.2 Zielsetzung**

...

## **1.3 Aufbau der Arbeit**

...

## 2 Theoretische Grundlagen

Im folgenden Kapitel werden die theoretischen Grundlagen des maschinellen Lernens und der verwendeten Modelle erläutert. Es wird auf die Konzepte des maschinellen Lernens, insbesondere des überwachten und unüberwachten Lernens, des Deep Learnings und der neuronalen Netze eingegangen. Anschließend wird die Funktionsweise von Diffusion-Modellen, insbesondere Stable Diffusion und DA-Fusion, sowie von Contrastive Learning und Supervised Contrastive Learning beschrieben. Zuletzt wird die bestehende Forschungslücke und die in dieser Arbeit thematisierte Integration von DA-Fusion und Supervised Contrastive Learning diskutiert.

### 2.1 Maschinelles Lernen

Die ersten großen Durchbrüche in der künstlichen Intelligenz (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 (I. Goodfellow et al., 2016).

Maschinelles Lernen (ML) beschreibt den Ansatz, Computer mit der Fähigkeit auszustatten, 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, die etwa aus Eingabe-Ausgabe-Paaren bestehen. Die Aufgaben  $T$  können sehr vielfältig sein, von einfachen Klassifikations- und Regressionsaufgaben bis hin zu komplexen Problemen wie Spracherkennung oder autonomen Fahren. Das Leistungsmaß  $P$  gibt an, wie gut das Modell die Aufgaben  $T$  löst, und kann z.B. der Anteil der korrekt klassifizierten Beispiele (Accuracy).

### 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 verschiedene Paradigmen, wobei die wichtigsten das überwachte (engl. *supervised*) und das unüberwachte (engl. *unsupervised*) Lernen sind.

Beim überwachten Lernen wird das Modell mit einem vollständig annotierten Datensatz trainiert. Das heißt, jeder Datenpunkt ist mit einem Klassenlabel versehen, sodass Eingabe-Ausgabe-Paare entstehen. Das Ziel ist es, eine Funktion zu lernen, die Eingaben auf die entsprechenden Ausgaben abbildet. Beispiele für überwachtes Lernen sind Klassifikations- und Regressionsaufgaben. Ein typisches Beispiel ist die Bilderkennung, bei der ein Modell darauf trainiert wird, Bilder von Katzen und Hunden zu unterscheiden. (<empty citation>)

Im Gegensatz dazu arbeitet unüberwachtes Lernen 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, bei dem eine Kombination aus beschrifteten und unbeschrifteten Daten verwendet wird, oder das selbstüberwachte Lernen, bei dem das Modell eigenständig Teile der Daten zur Erzeugung von Überwachungssignalen verwendet, anstatt sich auf externe, von Menschen bereitgestellte Labels zu verlassen. (<empty citation>)

### 2.1.2 Deep Learning

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

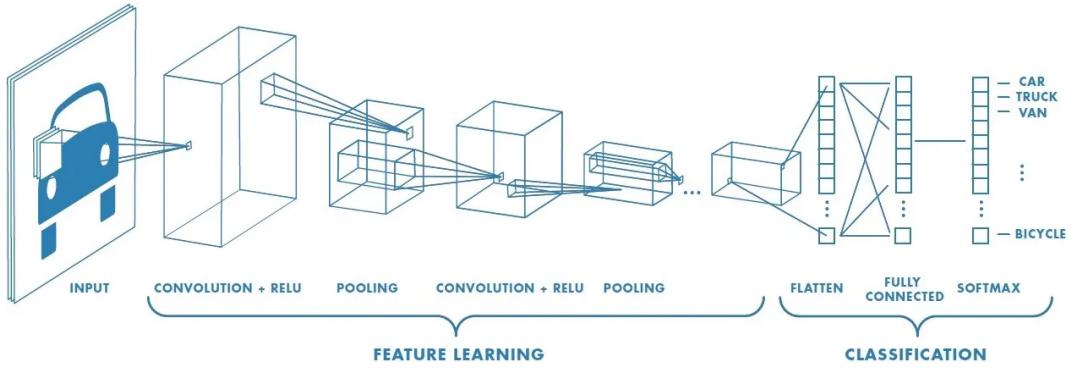


Abbildung 2.1: Aufbau eines Convolutional Neural Networks (CNN)

Ebenen (I. Goodfellow et al., 2016). Deep Learning ist daher eine spezialisierte Unterkategorie des maschinellen Lernens, in der künstliche neuronale Netze mit mehreren Schichten verwendet werden, um eine hierarchische Repräsentation von Daten zu ermöglichen. Jede Schicht transformiert die Eingabedaten in eine etwas abstraktere Darstellung.

Ein Beispiel für ein Deep Learning-Modell ist das Convolutional Neural Network (CNN), das speziell für die Verarbeitung von Bildern entwickelt wurde. Ein CNN besteht aus mehreren Schichten (siehe Abbildung 2.1), darunter Convolutional-Schichten, Pooling-Schichten und Fully-Connected-Schichten. Die Convolutional-Schichten enthalten Filter, die über das Bild gefaltet werden, um Merkmale wie Kanten, Texturen oder Formen zu extrahieren. Die Pooling-Schichten reduzieren die Dimensionalität der Daten, indem sie die Größe der Merkmalskarten verringern. Die Fully-Connected-Schichten dienen dazu, die extrahierten Merkmale zu klassifizieren oder zu regieren.

Deep Learning hat in den letzten Jahren erhebliche Fortschritte gemacht und findet Anwendung in Bereichen wie Bild- und Spracherkennung, autonomem Fahren und vielen anderen. Die Popularität von Deep Learning ist auf mehrere Faktoren zurückzuführen, darunter die Verfügbarkeit großer Datensätze, die Leistungsfähigkeit moderner Hardware und die Entwicklung effizienterer Algorithmen. (Zhou, 2021)

### 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

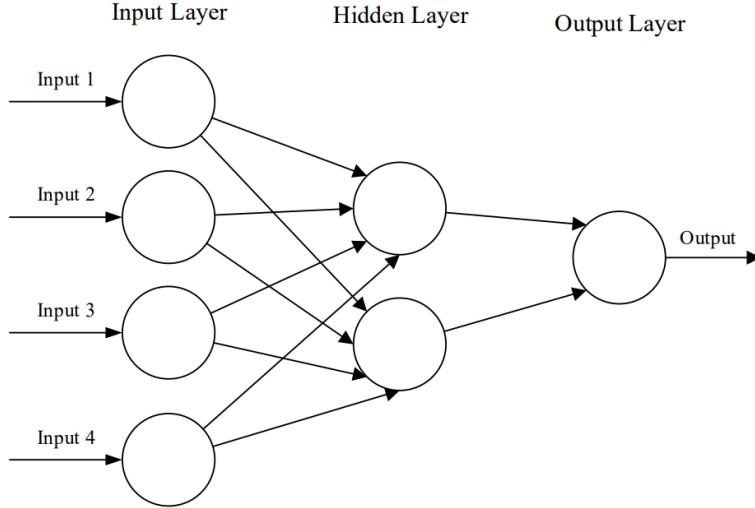


Abbildung 2.2: Darstellung eines einfachen neuronalen Netzes (O'Shea & Nash, 2015).

verbundenen Knoten (Neuronen), die in Schichten organisiert sind. Die Struktur eines neuronalen Netzes besteht aus einer Eingabeschicht, einer oder mehreren versteckten Schichten (engl. *hidden layers*) und einer Ausgabeschicht, wie in Abbildung 2.2 dargestellt.

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). 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 (engl. *weights*)  $w_{1j \dots nj}$ , welche die Stärke und Richtung (positiv oder negativ) des Einflusses der jeweiligen Eingaben auf das Neuron bestimmen.

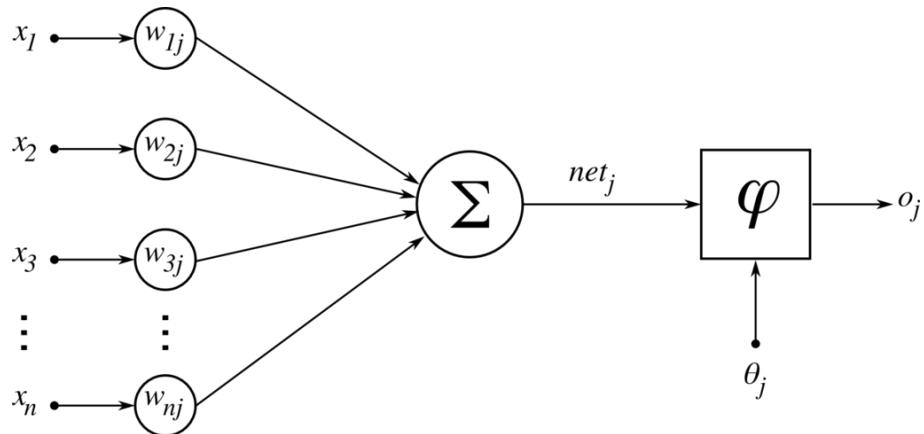


Abbildung 2.3: Das McCulloch-Pitts-Modell eines Neurons.

Das Neuron berechnet dann die gewichtete Summe aller Eingaben und falls ein bestimmter Schwellenwert (engl. *bias*)  $\theta$  überschritten wurde, wird das Neuron aktiviert:

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

Die Aktivierungsfunktion  $\phi$  kann dabei unterschiedlich gewählt werden, um die Ausgabe des Neurons zu modellieren. Eine simples Beispiel ist die sogenannte Schwellenwertfunktion (engl. *step function*), 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 (engl. *forward propagation*) durch das Netzwerk, um die Ausgabe zu berechnen. Es wird dann eine geeignete Verlustfunktion (engl. *loss function*) angewendet, um den Fehler 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 (engl. *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. Um zu berechnen, in welche Richtung die Gewichte angepasst werden sollen, wird der Gradient der Verlustfunktion berechnet. Dieser Gradient zeigt an, wie stark sich der Verlust ändert, wenn sich die Gewichte ändern. Die Anpassung der Gewichte erfolgt dann in kleinen Schritten entlang des Gradienten, um den Fehler zu minimieren. Dieser Prozess wird als Stochastic Gradient Descent (SGD) bezeichnet, da immer nur ein kleiner, zufällig ausgewählter Teil der Trainingsdaten (ein sogenannter Batch) verwendet wird, um den Gradienten zu berechnen und die Gewichte anzupassen. (I. Goodfellow et al., 2016)

## 2.1.4 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. Stattdessen werden falsche Vorhersagen mit übermäßigem Vertrauen getroffen.

Die Erkennung von OOD-Daten ist ein wichtiges Forschungsgebiet im maschinellen Lernen, da sie dazu beitragen kann, die Zuverlässigkeit und Sicherheit von KI-Systemen zu verbessern. Idealerweise sollte ein neuronales Netz höhere Softmax-Wahrscheinlichkeiten für In-Distribution-Daten und niedrigere Wahrscheinlichkeiten für OOD-Daten ausgeben. Durch Festlegen eines Schwellenwerts für diese Wahrscheinlichkeiten können Instanzen unterhalb des Schwellenwerts frühzeitig als OOD-Instanzen erkannt und entsprechend behandelt werden. In der Praxis kommt dieser Ansatz jedoch oft an seine Grenzen, da die Softmax-Wahrscheinlichkeiten nicht immer zuverlässig sind und das Modell auch für OOD-Daten hohe Wahrscheinlichkeiten ausgeben kann. Daher werden alternative Ansätze verwendet, wie etwa das Training eines binären Klassifikationsmodells zur Unterscheidung von In-Distribution und OOD-Daten.

## 2.1.5 Datenaugmentation und Generalisierung

Der Erfolg von Deep Learning-Modellen ist auf die Verfügbarkeit großer Datensätze angewiesen, die für das Training verwendet werden. Um auf zuvor ungesenen Daten generalisieren zu können, ist es wichtig, dass das Modell eine Vielzahl von Beispielen lernt, die die zugrundeliegenden Muster und Konzepte der Daten repräsentieren. Das Modell kann aber auch robuster gegenüber Variationen in den Eingabedaten gemacht werden, indem es mit leicht veränderten Versionen der Trainingsdaten trainiert wird. Dieser Prozess wird als Datenaugmentation bezeichnet.

Konkret werden unterschiedliche Transformationen auf die vorhandenen Daten angewendet, z.B. Rotation, Skalierung, Verschiebung, Spiegelung, Helligkeitsanpassung oder Rauschen. Die Transformationen werden meist mit zufälligen Parametern durchgeführt, 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 (<empty citation>). Einige Beispiele für Datenaugmentationstechniken sind in Abbildung 2.4 dargestellt.

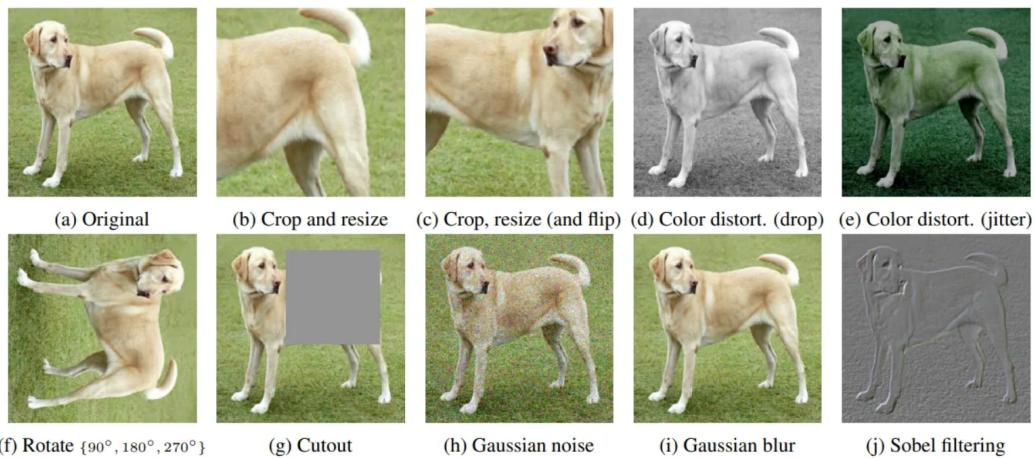


Abbildung 2.4: Beispiele für unterschiedliche Datenaugmentationstechniken und Kombinationen. (<empty citation>)

## 2.2 Synthetische Daten

Datenaugmentation kann an ihre Grenzen stoßen, wenn die verfügbaren Trainingsdaten nicht ausreichen oder nicht die notwendige Vielfalt aufweisen, um die Generalisierungsfähigkeit des Modells zu verbessern. In solchen Fällen können synthetische Daten eine nützliche Alternative oder Ergänzung zu echten Daten 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. In den folgenden Abschnitten werden einige der wichtigsten generativen Modelle vorgestellt, die in dieser Arbeit eine Rolle spielen.

### 2.2.1 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):

- einem **Encoder**, der hochdimensionale Eingabedaten in einem niederdimensionalen Darstellungsvektor komprimiert, und
- einem **Decoder**, der einen gegebenen Darstellungsvektor zurück in den ursprünglichen hochdimensionalen Raum umwandelt

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):

$$Loss = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^2, \quad (2.4)$$

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 Punkts 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:

$$D_{KL}(p(x)||q(x)) = \sum_x p(x) \log \frac{p(x)}{q(x)}, \quad (2.5)$$

wobei  $p(x)$  die tatsächliche Verteilung und  $q(x)$  die approximierte Verteilung ist. Die Gesamtverlustfunktion des VAEs ist dann die Summe aus Rekonstruktionsfehler und KL-Divergenz.

## 2.2.2 Generative Adversarial Networks

Ein Generative Adversarial Network (GAN) ist ein KI-Modell, das in (I. J. 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.

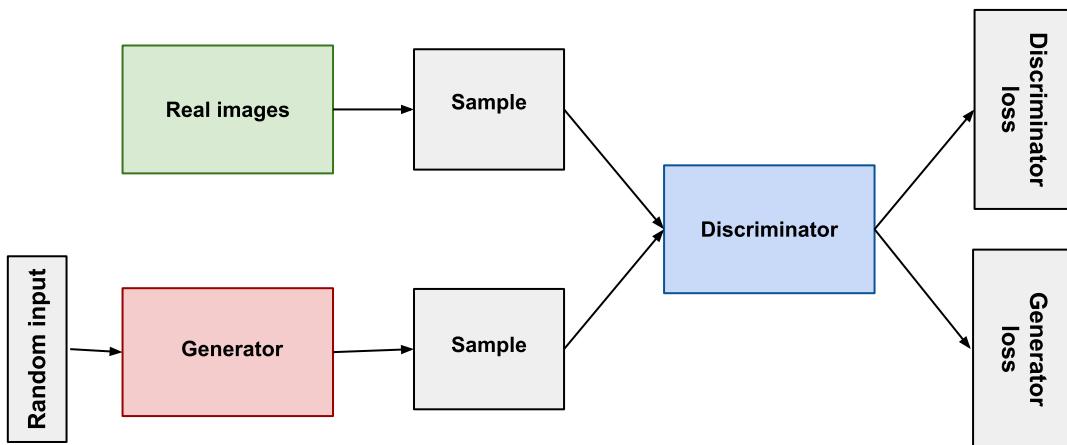


Abbildung 2.5: Überblick über die GAN-Struktur. Quelle: Google for Developers

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 Diskriminator 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 Diskriminator formuliert werden:

1. **Diskriminator-Training:** Der Diskriminator wird trainiert, um echte Daten von generierten Daten zu unterscheiden. Dies geschieht durch eine binäre Klassifikation („echt“ oder „synthetisch“). Der Diskriminator passt seine Gewichte an, um die Unterscheidung zu verbessern.
2. **Generator-Training:** Der Generator wird trainiert, um den Diskriminator zu täuschen. Dies geschieht, indem der Generator seine erzeugten Daten durch den Diskriminator laufen lässt und die Rückmeldung (Gradienten) des Diskriminators verwendet, um seine eigenen Parameter zu optimieren. Ziel ist es, den Diskriminator zu überlisten, sodass er die generierten Daten als echt klassifiziert.

Im Detail wird der Generator durch Minimierung der Log-Wahrscheinlichkeit, dass der Diskriminator die generierten Daten als echt klassifiziert, trainiert. Der Diskriminator 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 Diskriminator nicht in der Lage ist, zwischen echten und generierten Daten zu unterscheiden.

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 haben GANs auch einige Nachteile. Beispielsweise kann das Training eines GANs sehr instabil sein, da es schwierig ist, ein Gleichgewicht zwischen Generator und Diskriminator zu finden. Es kann auch zu Modus-Kollaps kommen, bei dem der Generator nur eine begrenzte Anzahl von Beispielen erzeugt, da der Diskriminator 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.

### 2.2.3 Diffusionsmodelle

Sogenannte 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. Seitdem haben sich Diffusionsmodelle als eine neue Klasse von generativen Deep-Learning-Modellen etabliert, die im Trainingsprozess die Struktur der Eingabedaten durch Hinzufügen von Rauschen schrittweise auflösen und anschließend darauf trainiert werden, aus dem verrauschten Bild das ursprüngliche Bild zu rekonstruieren.

Das Training von Diffusionsmodelle teilt sich somit 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.

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

$$q(x^{(0\dots T)}) = q(x^{(0)}) \prod_{t=1}^T q(x^{(t)}|x^{(t-1)}) \quad (2.6)$$

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:

$$p(x^{(0\dots T)}) = p(x^{(T)}) \prod_{t=1}^T p(x^{(t-1)}|x^{(t)}) \quad (2.7)$$

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., 2021) wurde DALL-E vorgestellt, ein Modell, das es ermöglicht, Bilder aus Textbeschreibungen zu generieren.

... (Rombach et al., 2022)

## 2.3 Semantische Datenaugmentation mit DA-Fusion

In (Trabucco et al., 2023) wird DA-Fusion, eine flexible, auf Stable Diffusion basierende Methode zur Datenaugmentation vorgestellt, die für diese Arbeit von besonderem Interesse ist. Der traditionelle Ansatz der Datenaugmentation, wie in Abschnitt ?? 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 Diffusionsmodells, um den Bildinhalt semantisch zu verstehen und automatisch neue, realistische Variationen zu generieren.



Abbildung 2.6: Vergleich zwischen semantischen Augmentationen aus Baseline-Methode und DA-Fusion (Trabucco et al., 2023).

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 neue Konzept bzw. für jede neue Klasse ein Pseudo-Token  $y$  in das Modell integriert, der unter Verwendung von Trainings-Prompts wie „a photo of a  $<y>$ “ und den zugehörigen Bilddaten trainiert wird.

Anschließend können aus den echten Bildern neue Augmentationen generiert werden, indem den Bildern eine geringe Menge an Rauschen hinzugefügt wird, welches dann wieder durch das feinabgestimmte Modell entfernt werden soll. Hier kommen wieder die selben Text-

Prompts zum Einsatz. Auf diese Weise bleibt die grundlegende Struktur der Bilder erhalten, während gleichzeitig neue, semantisch sinnvolle Variationen und Details erzeugt werden.

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.

## 2.4 Robuste Datenrepräsentation durch Contrastive Learning

Neben den vorgestellten Methoden zur Vervielfältigung der Trainingsdaten soll nun auch das Contrastive Learning als effektive Methode zur Verbesserung der Generalisierungsfähigkeit und Robustheit von Modellen vorgestellt werden. Die Methode zielt darauf ab, ähnliche Beispiele im Datensatz zu gruppieren und unähnliche Beispiele voneinander zu trennen. Durch die Kontrastierung der Daten wird eine Art von Supervision erzeugt, die es dem Modell ermöglicht, nützliche Repräsentationen der Eingabedaten zu lernen.

Contrastive Learning kommt ursprünglich aus dem unüberwachten Lernen. Da die Annotation von Daten sehr aufwendig sein kann, insbesondere in Domänen, in denen Expertenwissen erforderlich ist, hat sich das Contrastive Learning als vielversprechende Alternative mit äußerst starker Generalisierungsfähigkeit und Robustheit gegenüber Adversarial Attacks erwiesen (Liu, 2021).

Mittlerweile gibt es vermehrt Ansätze, Contrastive Learning auch im überwachten Setting anzuwenden, um die Repräsentationen von Daten zu verbessern. Während das Modell im unüberwachten Setting lernt, zwischen einzelnen Instanzen zu unterscheiden, werden im überwachten Setting 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.

### 2.4.1 Unsupervised Contrastive Learning

Ob unüberwacht oder überwacht: Im Contrastive Learning soll die Distanz ähnlicher Beispiele in einem Repräsentationsraum minimiert und die Distanz unähnlicher Beispiele maximiert

werden. Dazu bildet das Modell kontrastive Paare aus einem Anchor-Beispiel und verschiedenen positiv- oder negativ-Beispielen. Je nach Methode kann vor allem die Anzahl der positiv- und negativ-Beispiele pro Anchor variieren, wodurch sich auch die jeweiligen Loss-Funktionen unterscheiden.

Das wohl prominenteste Beispiel für Contrastive Learning ist **SimCLR**, das in (Chen et al., 2020) vorgestellt wurde. SimCLR verwendet den NT-Xent Loss (Normalized Temperature-scaled Cross-Entropy Loss), der die Ähnlichkeiten zwischen allen Paaren im Batch (auch *Mini-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 Convolutional Neural Network (CNN) in eine hochdimensionale 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 Loss zu bestimmen.

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

$$s_{i,j} = \frac{z_i \cdot z_j}{\|z_i\| \cdot \|z_j\|} \quad (2.8)$$

Der Loss 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.

## 2.4.2 Supervised Contrastive Learning

...

## **2.5 Forschungslücke**

...

### **2.5.1 Herausforderungen bei der Generierung synthetischer Daten**

...

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

...

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

...

# 3 Methodisches Vorgehen

In diesem Kapitel wird das methodische Vorgehen der Arbeit beschrieben. Es wird auf die Forschungsfragen und Hypothesen eingegangen, der verwendete Datensatz vorgestellt und die Implementierung der Modelle DA-Fusion und Supervised Contrastive Learning erläutert. Anschließend wird die synthetische Datengenerierung mit DA-Fusion und die Trainings- und Testdurchläufe mit Supervised Contrastive Learning beschrieben. Abschließend werden die Evaluationsmethoden und Metriken vorgestellt, die zur Analyse der Ergebnisse verwendet werden.

## 3.1 Forschungsfragen und Hypothesen

Im vorherigen Kapitel wurden Forschungslücken identifiziert, die sich aus der Verwendung von DA-Fusion im Supervised Contrastive Learning und der Verwendung von Near OOD-Augmentationen für das Negative Sampling ergeben. Um diese Lücken zu schließen, werden die folgenden Forschungsfragen und Hypothesen formuliert:

**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 3.2.1). 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.

## 3.2 Datensatz

Es soll zunächst genauer auf den verwendeten Datensatz eingegangen werden, um den untersuchten Anwendungsfall im Detail zu verstehen.

### 3.2.1 EIBA

Grundlage der Forschungsarbeit ist ein am Fraunhofer-IPK entstandener Datensatz von Gebrauchsgegenständen, darunter hauptsächlich Autoteile und Komponenten. Er wurde im Rahmen des Projekts “Sensorische Erfassung, automatisierte Identifikation und Bewertung von Altteilen anhand von Produktdaten sowie Informationen über bisherige Lieferungen” (EIBA) erstellt, das von 2019 bis 2023 lief und von der Circular Economy Solutions GmbH koordiniert und in Kooperation mit der Technischen Universität Berlin und der deutschen Akademie der Technikwissenschaften durchgeführt wurde. (<empty citation>)

Der Datensatz ist multimodal, d.h. er besteht aus verschiedenen Datenquellen, die unterschiedliche Informationen über die Gegenstände enthalten. Neben herkömmlichen RGB-Bildern aus verschiedenen Perspektiven und weiteren Bilddaten, wie z.B. Objektmasken, gibt es auch Metadaten, etwa das Gewicht, oder Beschreibungen der Objekte in natürlicher Sprache durch verschiedene Stichwörter („CarComponent“, „cylinder“, „rusty“, usw.).

### 3.2.2 Teildatensatz

Um die Rechenzeit zu reduzieren und die Experimente auf eine bestimmte Objektkategorie zu beschränken, wurde ein Teildatensatz des EIBA-Datensatzes verwendet. Dabei wurden zufällig 20 Klassen aus der super class „CarComponent“ ausgewählt. Es wurden außerdem nur die RGB-Bilder verwendet, allerdings kommen auch die Objektmasken im Pre-Processing der Daten zum Einsatz. ...

### **3.2.3 Vorverarbeitung**

...

## **3.3 Implementierung**

Für die Vorbereitung der in dieser Arbeit durchgeführten Experimente konnte sich größtenteils auf die Implementierung von DA-Fusion und Supervised Contrastive Learning aus den Quellen ... und ... gestützt werden. Beide Implementierungen sind in Python geschrieben und verwenden die Bibliothek PyTorch. ...

Dennoch mussten einige Anpassungen vorgenommen werden, um die Modelle auf den EIBA-Teildatensatz anzuwenden, und um die synthetischen Augmentationen aus DA-Fusion im Supervised Contrastive Learning zu verwenden. ...

### **3.3.1 DA-Fusion**

In ...'s Implementierung von DA-Fusion wird zunächst mit Textual Inversion ein vortrainiertes Stable Diffusion-Modell fine-tuned, indem ein neuer Token für jede Klasse im Datensatz erlernt wird. Um anschließend die Augmentationen zu generieren, werden die Bilder des Datensatzes genommen, Rauschen hinzugefügt und unter Konditionierung auf den entsprechenden Token wiederhergestellt. Je nachdem, wie viel Rauschen hinzugefügt wurde, entstehen so mehr oder weniger stark veränderte Bilder, die als synthetische Daten verwendet werden können.

...

Die Implementierung von DA-Fusion kann weitgehend unverändert angewendet werden, um synthetische Daten für den EIBA-Teildatensatz zu generieren. Es muss lediglich eine eigene Klasse für den Datensatz erstellt werden, die die Bilder und Masken aus dem EIBA-Datensatz lädt und die Token für die Klassen bereitstellt. ...

### **3.3.2 Supervised Contrastive Learning**

...'s Implementierung von Supervised Contrastive Learning beinhaltet drei Trainings-Skripte; eines für das Pre-Training der latenten Repräsentationen unter Verwendung der Supervised Contrastive Loss-Funktion, eines für die lineare Klassifikation der Repräsentationen und eines zum Training eines klassischen Klassifikator-Modells mit Cross Entropy Loss (zum Vergleich).

...

Auch hier muss eine eigene Klasse für den EIBA-Teildatensatz erstellt werden, die die Bilder und Masken lädt und die synthetischen Daten von DA-Fusion bereitstellt. Die Klasse muss nun auch Parameter bereitstellen, die die Verwendung der synthetischen Daten steuern, z.B. ob keine Augmentationen, ausschließlich positiv-Beispiele oder auch negativ-Beispiele verwendet werden sollen. ...

...

### **3.4 Synthetische Datengenerierung mit DA-Fusion**

...

### **3.5 Trainings- und Testdurchläufe mit Supervised Contrastive Learning**

...

### **3.6 Evaluationsmethoden und Metriken**

...

# 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

...



Abbildung 4.1: Beispieltext

## **4.2 Trainings- und Testergebnisse mit Supervised Contrastive Leraning**

...

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

...

### **4.2.2 Lineare Klassifikation**

...

## **4.3 Vergleich der Ergebnisse mit und ohne In-Distribution-Augmentationen**

...

## **4.4 Vergleich der Ergebnisse mit und ohne Near Out-of-Distribution-Augmentationen als Hard Negatives**

...

## **5 Diskussion**

...

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

...

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

...

# **6 Fazit**

...

## **6.1 Zusammenfassung der wichtigsten Erkenntnisse**

...

## **6.2 Beantwortung der Forschungsfragen**

...

## **6.3 Ausblick und potenzielle Weiterentwicklungen**

...

# Literatur

- Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A Simple Framework for Contrastive Learning of Visual Representations.
- 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* [<http://www.deeplearningbook.org>]. MIT Press.
- Goodfellow, I. J., 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>
- 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>
- Kingma, D. P., & Welling, M. (2022). Auto-Encoding Variational Bayes. <https://arxiv.org/abs/1312.6114>
- Liu, R. (2021). Understand and Improve Contrastive Learning Methods for Visual Representation: A Review. <https://arxiv.org/abs/2106.03259>
- Mitchell, T. M. (1997). *Machine Learning*. McGraw Hill.
- O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. <https://arxiv.org/abs/1511.08458>
- Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., & Sutskever, I. (2021). Zero-Shot Text-to-Image Generation. <https://arxiv.org/abs/2102.12092>
- 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>
- Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., & Ganguli, S. (2015). Deep Unsupervised Learning using Nonequilibrium Thermodynamics. <https://arxiv.org/abs/1503.03585>
- Trabucco, B., Doherty, K., Gurinas, M., & Salakhutdinov, R. (2023). Effective Data Augmentation With Diffusion Models.
- Zhou, Z.-H. (2021). *Machine Learning*. Springer.

# **Anhang**

Hier beginnt der Anhang. Siehe die Anmerkungen zur Sinnhaftigkeit eines Anhangs in Abschnitt ?? auf Seite ??.

Der Anhang kann wie das eigentliche Dokument in Kapitel und Abschnitte unterteilt werden. Der Befehl \appendix sorgt im Wesentlichen nur für eine andere Nummerierung.

# **Eigenständigkeitserklärung**

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

## **Viele zufällige Zahlen**

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, 21. Dezember 1940