

# 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 . . . . .	4
2.1.3 Neuronale Netze . . . . .	5
2.1.4 Overfitting . . . . .	7
2.1.5 Datenaugmentation . . . . .	8
2.1.6 Out-of-Distribution Daten . . . . .	9
2.2 Synthetische Daten . . . . .	9
2.2.1 Variational Autoencoder . . . . .	10
2.2.2 Generative Adversarial Networks . . . . .	12
2.2.3 Diffusionsmodelle . . . . .	14
2.3 Semantische Datenaugmentation mit DA-Fusion . . . . .	16
2.4 Robuste Datenrepräsentation durch Contrastive Learning . . . . .	18
2.4.1 Unsupervised Contrastive Learning . . . . .	18
2.4.2 Supervised Contrastive Learning . . . . .	19
2.5 Forschungslücke . . . . .	20
2.5.1 Herausforderungen bei der Generierung synthetischer Daten . . . . .	21
2.5.2 Synthetische Daten als negativ-Beispiele im Contrastive Learning . . . . .	21
2.5.3 Integration von DA-Fusion und Supervised Contrastive Learning . . . . .	22
<b>3 Methodisches Vorgehen</b>	<b>21</b>
3.1 Forschungsfragen und Hypothesen . . . . .	21

3.2	Datensatz . . . . .	22
3.2.1	EIBA . . . . .	22
3.2.2	Teildatensatz . . . . .	22
3.2.3	Vorverarbeitung . . . . .	23
3.3	Implementierung . . . . .	23
3.3.1	DA-Fusion . . . . .	23
3.3.2	Supervised Contrastive Learning . . . . .	23
3.4	Synthetische Datengenerierung mit DA-Fusion . . . . .	24
3.5	Trainings- und Testdurchläufe mit Supervised Contrastive Learning . . . . .	24
3.6	Evaluationsmethoden und Metriken . . . . .	24
<b>4</b>	<b>Ergebnisse</b>	<b>25</b>
4.1	Die generierten synthetischen Daten . . . . .	25
4.1.1	In-Distribution . . . . .	25
4.1.2	Near Out-of-Distribution . . . . .	25
4.2	Trainings- und Testergebnisse mit Supervised Contrastive Leraning . . . . .	26
4.2.1	Contrastive Pre-Training . . . . .	26
4.2.2	Lineare Klassifikation . . . . .	26
4.3	Vergleich der Ergebnisse mit und ohne In-Distribution-Augmentationen . . . . .	26
4.4	Vergleich der Ergebnisse mit und ohne Near Out-of-Distribution-Augmentationen als Hard Negatives . . . . .	26
<b>5</b>	<b>Diskussion</b>	<b>27</b>
5.1	Eignung von DA-Fusion für die synthetische Datengenerierung . . . . .	27
5.2	Wirksamkeit von synthetischen Near Out-of-Distribution-Daten im Supervised Contrastive Learning . . . . .	27
<b>6</b>	<b>Fazit</b>	<b>28</b>
6.1	Zusammenfassung der wichtigsten Erkenntnisse . . . . .	28
6.2	Beantwortung der Forschungsfragen . . . . .	28
6.3	Ausblick und potenzielle Weiterentwicklungen . . . . .	28
<b>Literatur</b>		<b>29</b>
<b>Anhang</b>		<b>31</b>

# Abbildungsverzeichnis

2.1	Ein einfaches Venn-Diagramm, das die Beziehung zwischen KI, ML und DL veranschaulicht . . . . .	4
2.2	Das McCulloch-Pitts-Modell eines Neurons. . . . .	5
2.3	Aufbau eines Convolutional Neural Networks (CNN) . . . . .	7
2.4	Einige Beispiele für unterschiedliche Datenaugmentationstechniken und Kombinationen, die in (Chen et al., 2020) verwendet wurden. . . . .	8
2.5	Überblick über die GAN-Struktur. Quelle: Google for Developers . . . . .	13
2.6	Darstellung des Vorwärts- und Rückwärtsdiffusion in einem Diffusionsmodell. . . . .	14
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). .	16
2.8	Vergleich zwischen semantischen Augmentationen aus Baseline-Methode und DA-Fusion (Trabucco et al., 2023). . . . .	17
2.9	Überblick über den Prozess zur Datenaugmentation mit DA-Fusion (Trabucco et al., 2023). . . . .	17
4.1	Beispieltext . . . . .	25

# **Tabellenverzeichnis**

## 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 die Forschungslücke vorgestellt, die die Arbeit adressiert, und ein eigener Ansatz zur Integration von DA-Fusion und Supervised Contrastive Learning definiert.

### 2.1 Maschinelles Lernen

Maschinelles Lernen (ML) ist als Teilgebiet der Künstlichen Intelligenz (KI) einzuordnen, einem interdisziplinären Forschungsfeld, das sich mit der Entwicklung von 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 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, 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.

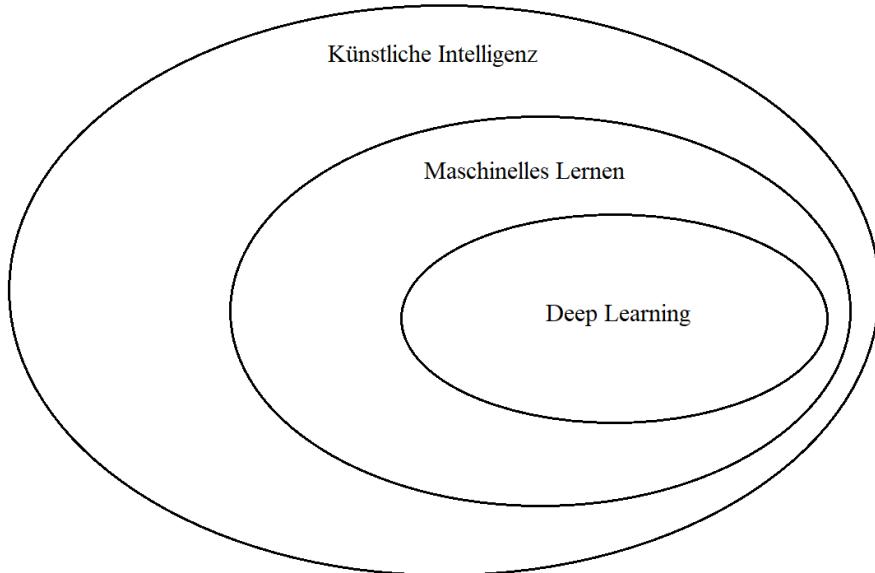


Abbildung 2.1: Ein einfaches Venn-Diagramm, das die Beziehung zwischen KI, ML und DL veranschaulicht.

### 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.

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.

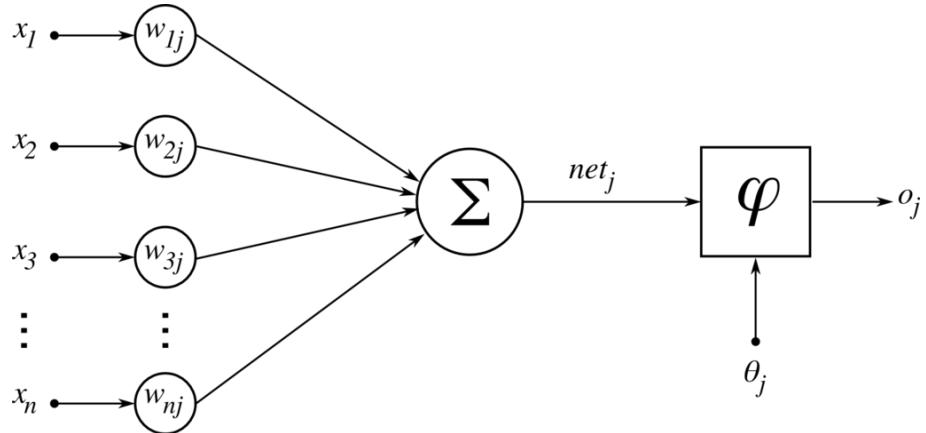


Abbildung 2.2: Das McCulloch-Pitts-Modell eines Neurons.

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.

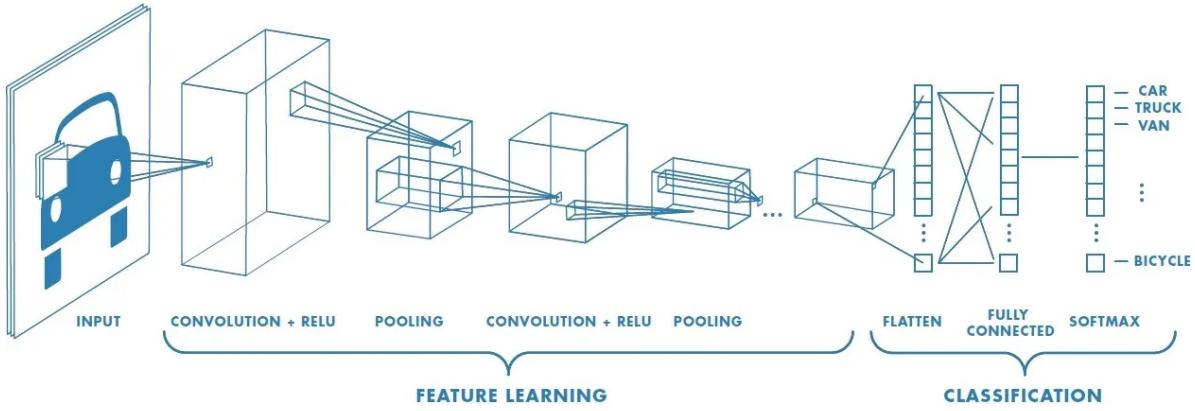


Abbildung 2.3: Aufbau eines Convolutional Neural Networks (CNN)

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 (LeCun et al., 1989) 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.

Ein 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. Beim Early Stopping wird das Training beendet, wenn die Leistung auf den Validierungsdaten beginnt, sich zu verschlechtern.

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

## 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 anstatt völlig neue Daten zu beschaffen, kann das Modell auch 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.

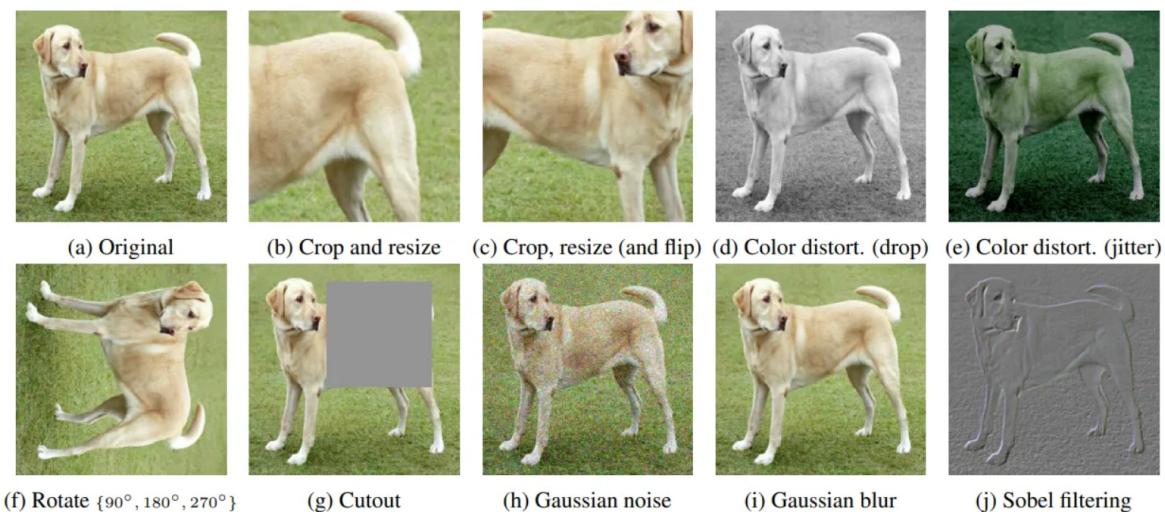


Abbildung 2.4: Einige Beispiele für unterschiedliche Datenaugmentationstechniken und Kombinationen, die in (Chen et al., 2020) verwendet wurden.

### 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 sollte ein neuronales Netz höhere Softmax-Wahrscheinlichkeiten für In-Distribution-Daten und niedrigere Wahrscheinlichkeiten für OOD-Daten ausgeben. Tatsächlich sind die Wahrscheinlichkeiten für OOD-Daten fast immer sehr hoch, meist reicht aber der Abstand zwischen In-Distribution 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 In-Distribution 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 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.

### 2.2.1 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 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:

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

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:

$$\text{Loss} = \text{Reconstruction Loss} + \text{KL-Divergence} \quad (2.6)$$

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>)

## 2.2.2 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.

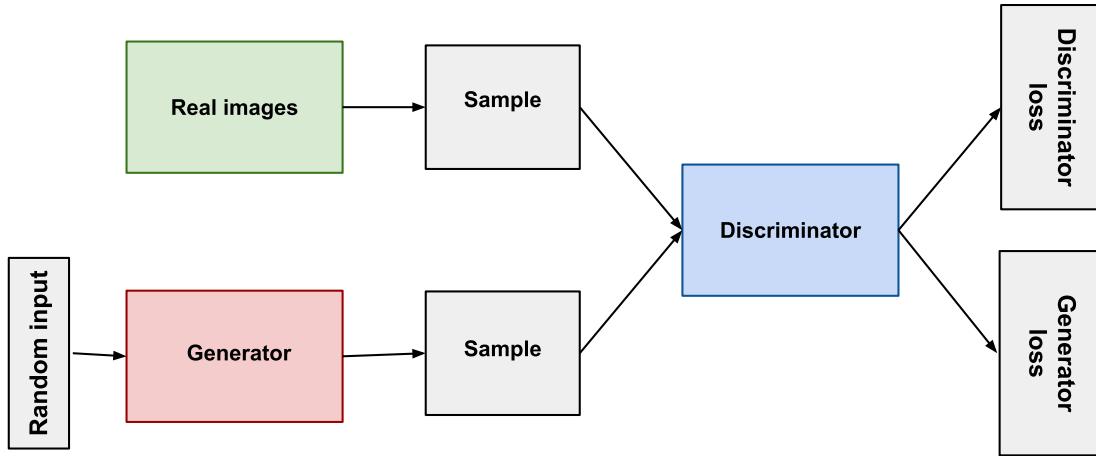


Abbildung 2.5: Überblick über die GAN-Struktur. Quelle: Google for Developers

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

$$D_{JS}(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M), \quad (2.7)$$

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

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.

### 2.2.3 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.

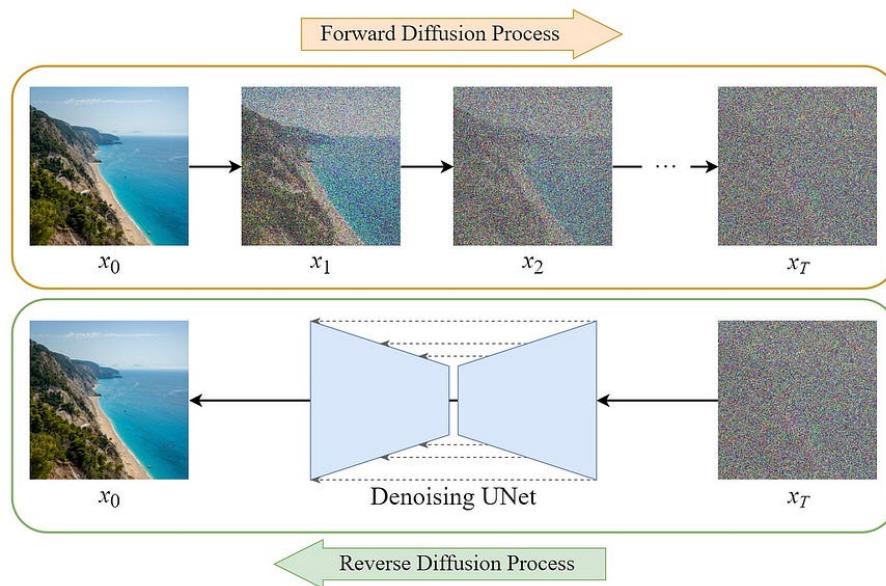


Abbildung 2.6: Darstellung des Vorwärts- und Rückwärtsdiffusion in einem Diffusionsmodell.

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 verrausches Bild umgewandelt. Die Wahrscheinlichkeitsdichte des verrauschten Bildes wird durch die Produktregel der bedingten Wahrscheinlichkeiten berechnet:

$$q(x^{(0\dots T)}) = q(x^{(0)}) \prod_{t=1}^T q(x^{(t)}|x^{(t-1)}) \quad (2.8)$$

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:

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

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

$$\text{Loss} = -\log p(x^{(0\dots T)}) \quad (2.10)$$

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.

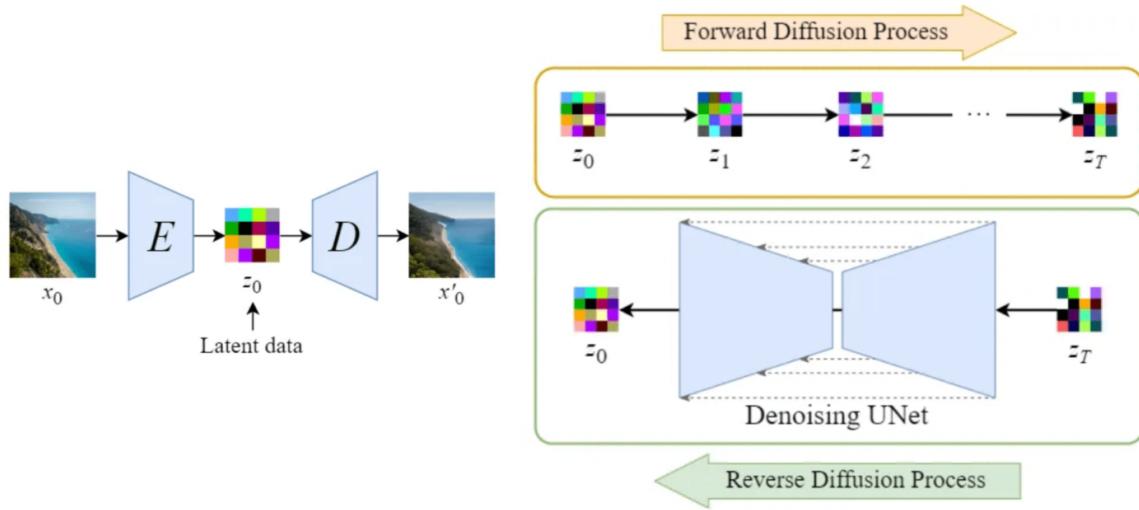


Abbildung 2.7: Darstellung des Vorwärts- und Rückwärtsdiffusion in Stable Diffusion:  
Die Diffusionsprozesse werden auf die latenten Darstellungen der Bilder  
angewendet (rechts), welche vorher mit einem VAE erstellt wurden (links).

## 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 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 Diffusionsmodells, um den Bildinhalt semantisch zu verstehen und automatisch neue, realistische Variationen zu generieren.

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



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

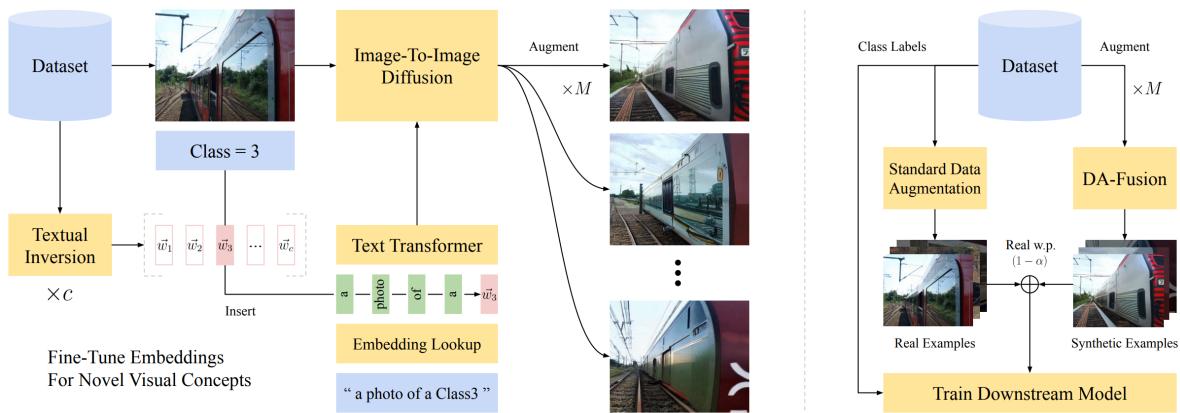


Abbildung 2.9: Überblick über den Prozess zur Datenaugmentation mit DA-Fusion (Trabucco et al., 2023).

das vortrainierte semantische Wissens 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 Timestep 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 Verlustfunktionen unterscheiden.

Das wohl prominenteste Beispiel für Contrastive Learning in der visuellen Domäne 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 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 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:

$$s_{i,j} = \frac{z_i \cdot z_j}{\|z_i\| \cdot \|z_j\|} \quad (2.11)$$

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.

## 2.4.2 Supervised Contrastive Learning

Trotz der vielversprechenden Ergebnisse im unüberwachten Kontext, gibt es auch im überwachten Setting vermehrt Interesse an Contrastive Learning. Hierbei wird die Label-Information 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 aufs überwachte Setting anpasst und mehrere positiv-Beispiele pro Anchor-Sample 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.

## 2.5 Forschungslücke

In den vorherigen Abschnitten wurden verschiedene Methoden zur Verbesserung der Generalisierungsfähigkeit und Robustheit von Modellen vorgestellt. Während die synthetische Datengenerierung bzw. Datenaugmentation und das Contrastive Learning vielversprechende Ergebnisse erzielen, gibt es noch einige Herausforderungen, die es zu bewältigen gilt. Im Folgenden sollen diese Herausforderungen genauer beleuchtet und ein neuer Ansatz vorgestellt werden, der in dieser Arbeit untersucht wird.

### 2.5.1 Herausforderungen bei der Generierung synthetischer Daten

Der Anwendungsfall, der schon in der Einleitung angesprochen wurde, stellt eine besondere Herausforderung für die Generierung synthetischer Daten dar:

- 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.

### 2.5.2 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, jedoch ohne Verwendung von synthetischen Daten. Die synthetischen Daten dürften also nicht zu weit entfernt von den echten Daten sein, um die Repräsentationen zu verbessern. Sie könnten daher als *Near OOD*-Beispiele bezeichnet werden, wobei sie noch OOD genug sein müssen, um die Distanz zwischen den In-Distribution und OOD-Daten

zu maximieren. Ob sich diese synthetischen Near OOD-Daten tatsächlich als gute Hard Negatives herausstellen, wird in dieser Arbeit untersucht.

...

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

...