

## **Bachelor-Thesis**

Name: Ilian Kohl

Thema: Datensatzanalyse mithilfe von lernenden Verfahren

Arbeitsplatz: Karlsruhe University of Applied Sciences, Karlsruhe

Referent: Prof. Dr. Hein

Korreferent: Prof. Dr. Baier

Abgabetermin: 18.03.2024

Karlsruhe, 17.11.2023

Der Vorsitzende des Prüfungsausschusses



Prof. Dr. Heiko Körner



---

## **Kurzfassung**

### **Datensatzanalyse mithilfe lernender Verfahren**

In dieser Arbeit wird ein Residual Convolutional Autoencoder (R-CAE) zur Dimensionsreduktion, Analyse und Bewertung komplexer Datensätze unterschiedlicher Domänen und Eigenschaften verwendet. Durch Training auf ausgewählte Domänenrepräsentierende Datensätze soll das Modell die Charakteristiken der zu vergleichenden Datensätze erfassen, um sie anschließend mithilfe des Encoder-Parts stark in ihrer Dimensionalität zu reduzieren. Aus den gewonnenen Datensatz-Kodierungen wird mit einer weiteren Dimensionsreduktion durch Aggregation für jeden Datensatz ein repräsentativer Feature-Vektor erstellt, der diesen kompakt in wenigen Werten beschreibt. Die Repräsentanten werden in einem distanzbasierten Auswahlverfahren hinsichtlich ihrer Domänen-Eigenschaften und Real-Nähe bewertet, um ähnliche zu identifizieren, zu clustern und ein ordinales Ranking zu erstellen, welches die grundsätzliche Eignung der Datensätze für ML-Aufgaben aufzeigen soll. Zur Performanz-Schätzung wurden 17 Original-Datensätze, um 99,22% in ihrer Dimensionalität reduziert auf eine Latente Dimension mit 3.072 Werten pro Datensatz. Das Auswahlverfahren zeigt, dass Datensätze derselben Domäne trotz variierender Eigenschaften eine hohe Ähnlichkeit aufweisen. Jedoch lassen die resultierenden Distanzen keine zuverlässigen Schlussfolgerungen außerhalb der Domäne zu, da diese trotz der starken Dimensionsreduktion noch immer dem Fluch der Dimensionalität unterliegen.

#### **Schlüsselwörter:**

Datensatz-Analyse, Residual Convolutional Autoencoder, Merkmalsextraktion, Dimensionsreduktion, Aggregationsmethoden, Auswahlverfahren, Performanz-Abschätzung

---

## **Abstract**

### **Dataset Analysis using Machine Learning Techniques**

This work uses a Residual Convolutional Autoencoder (R-CAE) for dimensionality reduction , analysis, and evaluation of complex datasets of different domains and properties. By training on selected domain-representative datasets, the model aims to capture the characteristics of the variant datasets and then drastically reduce their dimensionality using the encoder component. A representative feature vector is derived for each dataset from the acquired dataset encodings, through additional dimensionality reduction via aggregation, compactly encapsulating its key features. The representatives are then evaluated in a subsequent distance-based selection process based on their domain properties and proximity to real data, aiming to identify similar datasets, cluster them, and create an ordinal ranking indicating the general suitability of the datasets for ML tasks. For performance estimation, 17 original datasets were reduced by 99.22% in dimensionality to a latent dimension with 3.072 values per dataset. The selection process reveals those datasets within the same domain exhibit high similarity despite varying properties. However, the resulting distances do not allow for reliable inferences outside the domain, as they still succumb to the curse of dimensionality despite significant dimensionality reduction.

#### **Keywords:**

Dataset Analysis, Residual Convolutional Autoencoder, Feature Extraction, Dimensionality Reduction, Aggregation, Dataset Selection, Performance Estimation

---

## **Danksagung**

Mein herzlicher Dank geht an die Forschungsgruppe IRAS für die Bereitstellung von Ressourcen. Insbesondere danke ich Professor Björn Hein und den Doktoranden Moritz Weisenböhler und Philipp Augenstein für die fachliche Unterstützung in diesem Projekt.

Des Weiteren möchte ich meiner Hochschule für die Möglichkeit danken, dieses Projekt durchzuführen.

Ein besonderer Dank gilt auch meinen Freunden und meiner Familie, die mich während des gesamten Projekts unterstützt und ermutigt haben.



---

# Inhaltsverzeichnis

Abkürzungsverzeichnis .....	XII
1 Einleitung .....	1
1.1 Problemstellung .....	1
1.2 Zielsetzung.....	1
1.3 Aufbau der Arbeit .....	3
2 Theoretische Grundlagen .....	5
2.1 Dimensionsreduktion .....	5
2.1.1 Definition der Dimensionsreduktion .....	5
2.1.2 Fluch der Dimensionalität .....	5
2.1.3 Ziele der Dimensionsreduktion.....	7
2.1.4 Feature Selection vs. Feature Extraction .....	7
2.1.5 Principal Component Analysis (PCA).....	8
2.1.6 Manifold Learning.....	10
2.1.7 Kernel-PCA, t-SNE und UMAP.....	13
2.2 Autoencoder (AE).....	14
2.2.1 Feature-Vektor im Latenten Raum.....	15
2.2.2 Klassischer Autoencoder (AE) .....	15
2.2.3 Convolutional Autoencoder (CAE).....	16
2.2.4 Aktivierungsfunktionen.....	18
2.3 Residual-Connections .....	19
2.4 Evaluierungs-Metriken .....	20
2.4.1 Structural Similarity Index (SSI).....	20

---

2.4.2	Peak to Signal Noise Ratio (PSNR).....	20
2.4.3	Mean Absolute Error (MAE) und Mean Squared Error (MSE) .....	20
2.5	Distanz-Metriken .....	21
2.5.1	Euklidische Distanz .....	21
2.5.2	Manhattan Distanz.....	21
3	Stand der Technik.....	23
3.1	Generative AE.....	23
3.2	Undercomplete vs. Overcomplete.....	24
3.3	Deep AE und Stack AE.....	24
3.4	Regularized AE.....	25
3.5	Denoising AE.....	26
3.6	Zusammenfassung.....	26
4	Methoden und Analyse.....	27
4.1	Anforderungsanalyse für Lern- und Auswahlverfahren .....	27
4.1.1	Anforderungen zur Merkmalsextraktion und Dimensionsreduktion .....	27
4.1.2	Anforderungen zum Auswahlverfahren und Bewertungskriterien.....	27
4.2	Methoden zur Merkmalsextraktion und Dimensionsreduktion.....	28
4.3	Evaluierungsmethoden der CAE-Modelle .....	29
4.4	Analyse des Latenten Raums .....	30
4.4.1	Clustering-Verfahren.....	30
4.4.2	Heat Maps.....	30
4.5	Aggregationsmethoden .....	31
4.6	Methoden des Auswahlverfahrens .....	31

---

---

4.6.1	Voraussetzung zur Anwendung der Distanz-Metriken.....	32
4.6.2	Konsequenzen der Flatten-Operation .....	32
4.6.3	Distanzen auf Feature-Vektor-Level .....	32
4.6.4	Distanzen auf Feature-Map-Level .....	32
5	Konzept .....	33
5.1	Planung der Modellarchitektur.....	33
5.2	Geplante Trainings-Konfigurationen .....	35
5.3	Weiterverarbeitung der Ergebnisse.....	36
5.4	Planung des Rankings durch Auswahlverfahren .....	36
5.4.1	Distanzbasierter Ansatz .....	37
5.4.2	Clusteranalyse und SVM als Alternativen.....	41
6	Implementierung .....	43
6.1	Datenvorbereitung.....	43
6.1.1	Verfügbare Datensätze .....	43
6.1.2	Padding .....	44
6.1.3	Masken .....	45
6.2	AE- Modelle.....	45
6.2.1	Modell Bausteine.....	45
6.2.2	Eingabeparameter und Datenfluss .....	49
6.2.3	Erweiterung mit Dense Layer.....	50
6.2.4	Leicht stärkerer Encoder.....	51
6.3	Modell-Training.....	52
6.3.1	Ressourcenbeschränkung.....	52

---

---

6.3.2	Ermittlung der Grenzen .....	52
6.3.3	Hyperparameter .....	55
6.3.4	Trainings-, Validierungs- und Testdaten .....	55
6.3.5	Konfigurationen der Vorstudie .....	55
6.3.6	Konfigurationen der Hauptstudie .....	56
6.4	Kodierung der Datensätze.....	58
6.5	Aggregation der Feature-Vektoren .....	60
6.6	Distanzbasiertes Auswahlverfahren.....	61
6.6.1	Grundlage der Bewertung.....	61
6.6.2	Aufstellung der Distanzmatrix und Normalisierung.....	62
6.6.3	Erster Schritt: Domänen-Reihenfolge.....	63
6.6.4	Zweiter Schritt: Domänen-Zugehörigkeit.....	64
6.6.5	Dritter Schritt: Real-Nähe.....	65
6.6.6	Variance-Penalty-Reward.....	65
6.6.7	Visualisierung mit MDS.....	66
6.6.8	Clusteranalyse mit UMAP.....	67
6.6.9	SVM-Training .....	67
7	Ergebnisse und Evaluierung .....	69
7.1	Evaluierung der Modell-Trainings.....	69
7.1.1	Evaluierung der Vorstudie.....	69
7.1.2	Evaluierung der Hauptstudie .....	75
7.2	Latenter Raum.....	84
7.3	Evaluierung des Auswahlverfahrens.....	85

---

---

7.3.1	Referenzwerte der Objekterkennungsaufgabe.....	85
7.3.2	Distanzmatrix auf Feature-Vektor-Level.....	86
7.3.3	Erfasste Domänen-Reihenfolge .....	88
7.3.4	Evaluierung der Domänen-Zuordnung.....	88
7.3.5	Domain-Based Ranking mit Real-Nähe .....	90
7.3.6	Variance-Penalty-Reward Ranking .....	91
7.3.7	Vergleich der Ergebnisse.....	91
7.3.8	Auswertung der SVM-Ergebnisse .....	93
7.3.9	Auswertung der Clusteranalyse .....	94
8	Diskussion .....	95
8.1	Modell-Grenzen .....	95
8.2	Grenzen des Auswahlverfahrens.....	96
9	Zusammenfassung und Ausblick.....	99
	Literaturverzeichnis.....	101
	Abbildungsverzeichnis .....	107
	Tabellenverzeichnis.....	110
	Anhang .....	113

---

## Abkürzungsverzeichnis

AE	Autoencoder
CAE	Convolutional Autoencoder
R-CAE	Residual Convolutional Autoencoder
DAE	Denoising Autoencoder
SAE	Sparse Autoencoder
UAE	Undercomplete Autoencoder
OAE	Overcomplete Autoencoder
RAE	Regularized Autoencoder
AAE	Adversarial Autoencoder
CNN	Convolutional Neural Network
MDS	Multidimensional Scaling
t-SNE	t-Stochastic Neighbour Estimation
PCA	Principal Component Analysis
UMAP	Uniform Manifold Approximation and Projection
DR	Domain Randomization
SE	Scene Engineering
NLDR	Non-Linear-Dimension-Reduction
SVM	Support Vector Machine

---

# **1 Einleitung**

## **1.1 Problemstellung**

Die Auswahl geeigneter Datensätze für das Training von Maschine-Learning-Modellen ist eine schwierige Aufgabe. Trainings verbrauchen Zeit- und Rechenressourcen, insbesondere bei mächtigen Modellen. Diese sind jedoch oft erwünscht und erforderlich, um gute Leistungen zu erzielen. Dabei hängt die Qualität des Modells direkt von den Trainingsdaten ab, wobei ungeeignete Daten die Leistung erheblich beeinträchtigen können. Daher ist es üblich, eine Vielzahl von Datensätzen zu testen, um die optimal geeigneten auszuwählen. Allerdings ist es angesichts einer großen Anzahl potenziell in Frage kommender Datensätze nicht nur unpraktikabel, sondern auch sehr ressourcenintensiv, Modelle mit allen verfügbaren zu evaluieren, um die Leistung jedes einzelnen Datensatzes zu bewerten.

Die Eignung eines Datensatzes variiert je nach Anwendungsfall, und oft besteht Unsicherheit darüber, welcher die Anforderungen am besten erfüllt. Selbst wenn bereits ein geeigneter Datensatz für das Training vorhanden ist, kann es schwierig sein, einen passenden Ersatz mit ähnlichen Eigenschaften zu finden, wenn zusätzliche Trainingsdaten erforderlich sind. Synthetische Daten werden oft als Alternative zu teuren Real-Daten verwendet, da diese meist schwer zu beschaffen sind. Die Generierung synthetischer Daten mittels Techniken wie Domain-Randomization oder Scene-Engineering führt oft zu spezialisierten Domänen mit Datensätzen die eine Vielzahl von Variationen in Beleuchtung, Texturen und anderen Umgebungsparametern aufweisen. Auch wenn diese Datensätze oft als nützlich und vielseitig anerkannt werden, sind sie nicht alle gleichermaßen geeignet. Jede Variation kann die Leistung des Modells beeinflussen, selbst wenn die Bilddaten für das menschliche Auge sehr ähnlich erscheinen. Wenn solche feinen Abweichungen zuverlässig erkannt und bewertet werden können, könnte das eine effizientere Datenauswahl unterstützen.

Das Problem ist daher, dass eine effektive Methode zur Auswahl und Bewertung von Datensätzen notwendig ist, um die Datensätze zu identifizieren, die den größten Mehrwert bieten, ohne umfangreiche Ressourcen für das Training und Testen einer Vielzahl von Datensätzen zu verbrauchen.

## **1.2 Zielsetzung**

Ausgehend von der Problemstellung ist das Hauptziel der Arbeit, ein maschinelles Lernverfahren, im Speziellen Autoencoder, zur Analyse und Dimensionsreduktion komplexer Datensätzen zu nutzen, um diese auf ihre essenziellen Merkmale zu reduzieren. Hierbei sollen die Datensätze möglichst kompakt in wenigen, jedoch aussagekräftigen Werten beschrieben und für jeden Datensatz einen Feature-Vektor generiert werden, der ihn mit seinen charakteristischen Eigenschaften repräsentiert.

---

Es soll ein System entwickelt werden, das die Datensätze mithilfe der gewonnenen Feature-Vektoren nach ihrer grundsätzlichen Eignung für ML-Trainings bewertet und ein ordinales Ranking erstellt, mit dem ihre Performanz abgeschätzt werden kann. Diese Bewertungen sollen dazu dienen, eine gezielte Vor-Auswahl von Datensätzen zu ermöglichen, um ressourcenintensive Trainings zu vermeiden und Rechenkapazitäten und Zeit effizienter nutzen zu können, indem nur die weiter berücksichtigt werden, die vom System am besten eingestuft werden. Konkret soll das System fähig sein, Datensätze zu vergleichen und zu priorisieren, die am besten geeigneten auszuwählen und ungeeignete mit hinderlichen Merkmalen auszusortieren. Des Weiteren soll es auch in der Lage sein Datensätze mit ähnlichen Eigenschaften zu identifizieren und anhand der Kodierungen, subtile Unterschiede erkennen, die visuell nicht offensichtlich sind, jedoch als feine Nuancen die Modellleistung beeinflussen können. Es soll zur allgemeinen Analyse dienen und daher auf verschiedene Datensätze und Domänen anwendbar sein. Die Bewertung soll möglichst unabhängig von spezifischen Anwendungsfällen und der Datensatz-Arten selbst sein. Zusammengefasst, soll ein System zur Performanz-Abschätzung entwickelt werden, um die Komplexität bei der Auswahl von einer Vielzahl an Datensätzen zu vereinfachen und einen automatisierten effizienten Ansatz zur Bewertung bereitzustellen.

Ausgehend von der Zielsetzung hängt die Effizienz des gesamten Systems maßgeblich von der Fähigkeit des Autoencoders ab, essenzielle Merkmale aus den Daten zu extrahieren. Sollte der Autoencoder dabei auf Schwierigkeiten stoßen – sei es durch eine unzureichende Extraktion wichtiger Merkmale oder mangelnde Repräsentation der Variabilität der Domänen –, würde das eine schlechte Grundlage für das Auswahlverfahren bieten und dadurch würde auch das Ranking Ergebnisse erzeugen, die nicht aussagekräftig sind.

Daraus ergeben sich folgende forschungsleitenden Fragen:

1. Inwieweit sind Autoencoder in der Lage, eine starke Dimensionsreduktion auf komplexe Datensätze durchzuführen und gleichzeitig aussagekräftige Feature-Vektoren zu generieren?
2. Welche Datensätze eignen sich am besten für das Training, wenn die zu bewertenden Datensätze aus verschiedenen Domänen stammen und spezifische Eigenschaften aufweisen?
3. Ist es möglich, den Anteil an Real-Trainingsdaten gering zu halten und dennoch sicherzustellen, dass der Autoencoder die Eigenschaften des Real-Datensatzes ausreichend erfasst, insbesondere wenn Real-Eigenschaften ein Bewertungskriterium des Auswahlverfahren sind?
4. Wie kann ein Bewertungssystem gestaltet werden, das die Feature-Vektoren der Datensatz-Kodierungen nutzt, um die Eignung der Datensätze ganzheitlich zu beurteilen, ohne dass ein separates Training zur Bewertung jedes kodierten Datensatzes erforderlich ist?

---

Basierend auf diesen forschungsleitenden Fragen lassen sich folgende Hypothesen ableiten:

1. Die Dimensionsreduktion durch Autoencoder wird abhängig der Reduktionsstärke Feature-Vektoren erzeugen, die die wesentlichen Merkmale des Datensatzes erfassen.
2. Durch eine gezielte Auswahl von Datensätzen als Repräsentanten für vorhandenen Domänen wird die Auswahl der Trainingsdaten aus diesen Datensätzen dazu führen, dass der Autoencoder in der Lage ist, auch Variationen innerhalb der Domäne in seinen Kodierungen zu erfassen, selbst wenn die Variationen Eigenschaften aufweisen, auf denen nicht explizit trainiert wurde.
3. Um die Charakteristiken der realen Domäne angemessen zu erfassen, wird ein ausgewogener Anteil an echten und synthetischen Trainingsdaten erforderlich sein.
4. Die zuverlässige Bewertung und das Ranking von Datensätzen werden problematisch, wenn keine eindeutige Ähnlichkeit zu den Repräsentanten der existierenden Domänen erkannt wird. Die Unklarheit in der Zuordnung kann dazu führen, dass bestimmte Datensätze außerhalb des Bewertungsverfahrens fallen, da sie keiner spezifischen Kategorie zugeordnet werden können, die als Basis zur Performanz-Abschätzung dient.

### **1.3 Aufbau der Arbeit**

Im nachfolgenden Kapitel werden zunächst die Theoretische Grundlagen (2) erklärt, die zum Verständnis der im Hauptteil angewandten Methoden und Verfahren nötig sind. Anschließend wird im Stand der Technik (3) auf die Anwendungen der Autoencoder eingegangen und welche Autoencoder-Typen sich entwickelt haben. Methoden und Analyse (4) ist das erste Kapitel des Hauptteiles. Hier werden die Anforderungen analysiert und ausgehend davon besprochen, welche Methoden für die Umsetzung der Lösung in Frage kommen und warum sie möglichen Alternativen vorgezogen werden. Insbesondere wird darauf eingegangen, wieso Autoencoder als Lernverfahren für die Merkmalsextraktion und Dimensionsreduktion verwendet werden. Danach wird das Konzept (5) erklärt, nach dem das Performanz-Schätzungssystem entwickelt wurde – wie der Autoencoder angewandt und entwickelt werden soll, wie er trainiert wird und wie seine Ergebnisse weiterverwendet werden sollen, um sie letztlich im Auswahlverfahren zu nutzen. Danach wird die Implementierung (6) des Konzepts detailliert erläutert, von der Datenvorbereitung, zur Implementierung der Modell-Architektur, den Trainingskonfigurationen, den Datensatz-Kodierungen und der Umsetzung des Auswahlverfahrens zur Erstellung des Rankings. Im letzten Kapitel des Hauptteiles Ergebnisse und Evaluierung (7) werden die Ergebnisse des Trainings und des Auswahlverfahrens ausgewertet. In der Diskussion (8) wird die Aussagekraft der Ergebnisse nochmals aufgegriffen, inwiefern weitere Untersuchungen notwendig sind und welche Grenzen und Schwächen das System hat. Zuletzt werden im Kapitel Zusammenfassung und Ausblick (9), die Ergebnisse und umgesetzten Ziele zusammengefasst, mit Rückblick auf die zu Beginn getroffenen Hypothesen.



---

## 2 Theoretische Grundlagen

Hier werden die Grundlagen der Methoden und Verfahren vorgestellt, die als Basis für das Verständnis der späteren Implementierung, Datensatz-Analyse und Evaluierung dienen.

### 2.1 Dimensionsreduktion

#### 2.1.1 Definition der Dimensionsreduktion

Die Dimensionalität bezieht sich auf die Anzahl der Merkmale eines Datensatzes. Jedes Merkmal ist eine eigenständige Variable, die in ML-Modellen als Eingabe dienen kann (Vlachos, 2011). Datensätze, die viele Merkmale enthalten, werden als hochdimensionale Daten bezeichnet. Das ist abzugrenzen vom Begriff „Big Data“, der Datensätze charakterisiert, die viele Datenpunkte aufweisen. Diese können auch hochdimensional sein, wenn sie viele Merkmale enthalten, jedoch schließen hochdimensionale Daten auch Datensätze ein, die eine hohe Anzahl an Features haben, mit wenig verfügbaren Datenpunkten (Schintler, 2020, S. 1). Die Dimensionsreduktion bezieht sich daher auf Techniken, die darauf abzielen, die Anzahl der Eingangsvariablen hochdimensionaler Daten zu reduzieren und in wenigen Dimensionen, darzustellen. Mathematisch beschrieben durch:

$$Y = f(X); X = [x_1, x_2, \dots, x_n]^T; Y = [y_1, y_2, \dots, y_m]^T; m < n \quad (2.1)$$

$Y$  soll nach Anwendung der Reduktionsfunktion  $f$  nur die wichtigsten Merkmale der Originaldaten enthalten, während redundante Informationen entfernt werden. Sie wird meist zur Vorverarbeitung von Datensätzen angewandt, um den „Fluch der Dimensionalität“ zu bewältigen (Jia, Sun, Lian, & Hou, 2022, S. 1-2).

#### 2.1.2 Fluch der Dimensionalität

Entgegen der Annahme „Je mehr Informationen desto besser“, können hochdimensionale Daten zu Problemen führen (Altman & Krzywinski, 2018, S. 1). Der Fluch der Dimensionalität, nach (Bellman, 1957), bezieht sich auf die Herausforderungen, die bei der Verarbeitung, Modellierung und Visualisierung von hochdimensionalen Daten auftreten.

Angenommen, es gibt zwei binäre Merkmale,  $A$  und  $B$ , die 0 oder 1 sein können. Nur Merkmal  $A$  als Eingangsvariable bedeutet eine Dimension ( $d = 1$ ) mit zwei  $2^1$  möglichen Kombinationen: (0), (1). Mit Merkmal  $B$  gibt es zwei Dimensionen ( $d = 2$ ) und dadurch vier  $2^2$  Kombinationen: (0,0), (0,1), (1,0), (1,1). Für ein weiteres Merkmal  $C$  sind es drei Dimensionen ( $d = 3$ ) und damit acht  $2^3$  Kombinationen: (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1). Die Anzahl möglicher Kombinationen verdoppelt sich für jedes zusätzliche binäre Merkmal.

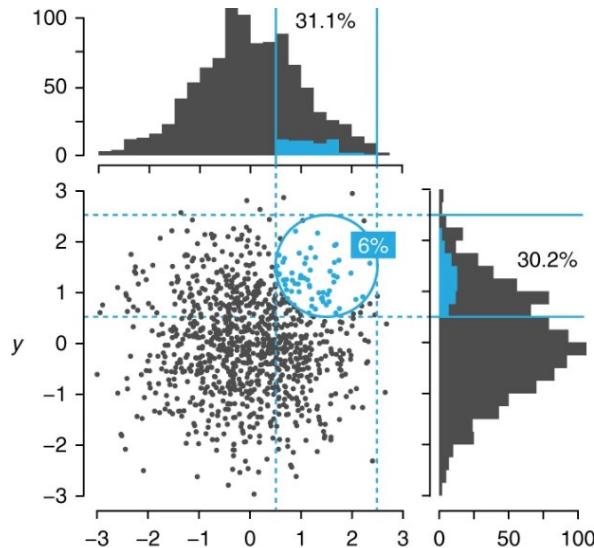


Abbildung 2.1: Veranschaulichung der Data Sparsity in hochdimensionalen Daten; Quelle: (Altman & Krzywinski, 2018)

Bei wenigen, aber hochdimensionalen Daten wird von „Data Sparsity“ geredet, da der Raum, den die Daten einnehmen mit steigender Dimensionalität größer wird, aber gleichzeitig die Dichte der Datenpunkte abnimmt, da die Punkte weiter voneinander weggezogen werden, wie in Abbildung 2.1 dargestellt. Mit einer wachsenden Anzahl von Dimensionen, wächst die Anzahl an benötigten Datenpunkten für eine gute Modell-Performanz exponentiell. Denn wird das Modell auf hochdimensionalen Datenpunkten eines Datensatzes mit wenigen Datenpunkten trainiert, kann das zu Overfitting führen, da die Trainingsdaten, nicht alle Merkmals-Kombinationen abdecken, die als Datenpunkt im hoch-dimensionalen Raum auftreten könnten (Berisha, et al., 2021, S. 3). Und auch wenn ausreichend Datenpunkte – die eine genügende Variation bieten - existieren, ist die Verarbeitung hochdimensionaler Daten selbst rechenaufwändig (Jia, Sun, Lian, & Hou, 2022, S. 1-2). Auch die Visualisierung ist problematisch. 2D- oder 3D- Daten sind noch gut interpretierbar, jedoch wird die Datenstruktur mit zunehmender Dimension unübersichtlicher.

Auf Distanzfunktionen wirken sich hochdimensionale Daten auch negativ aus, da mit zunehmender Dimensionalität der Unterschied zwischen der größten und kleinsten Distanz relativ gesehen immer kleiner wird. Die folgende Formel zeigt das der Grenzwert des Verhältnisses gegen Null strebt, wenn die Dimensionalität gegen unendlich geht (Houle, 2010, S. 483).

$$\lim_{dist \rightarrow \infty} \frac{dist_{max} - dist_{min}}{dist_{min}} = 0 \quad (2.2)$$

Das bedeutet, dass die Distanzen zwischen den Datenpunkten in einem hochdimensionalen Raum konvergieren – also immer ähnlicher werden - und dadurch an Unterscheidungskraft verlieren.

Um diese Probleme zu bewältigen, kann die Dimensionalität reduziert werden.

### 2.1.3 Ziele der Dimensionsreduktion

Mit der Dimensionsreduktion soll daher die Anzahl der Eingangsvariablen so verringert werden, dass irrelevante und redundante Merkmale entfernt, aber wesentliche Informationen beibehalten werden, um sie für das Modell-Training vorzubereiten oder zur Visualisierung und leichterer Interpretation. Durch die Reduktion wird die Anzahl der zu berücksichtigen Merkmals-Kombinationen der Modelle verringert, was wiederum die Gefahr des Overfitting minimiert, die Modell-Leistung durch Entfernen der Störfaktoren und Rauschen verbessert und zu weniger rechenaufwendigen Operationen führt.

### 2.1.4 Feature Selection vs. Feature Extraction

Dabei gibt es zwei Arten von Dimensionsreduktion. Bei der Feature-Selection werden bestimmte Merkmale aus dem ursprünglichen Datensatz ausgewählt und andere verworfen. Ziel ist es eine Teilemenge der ursprünglichen Dimensionalität des Datensatzes zu ermitteln, die die relevantesten Merkmale enthält (Genender-Feltheimer, 2018, S. 114). Der einfachste Ansatz dazu ist, die Merkmale zu selektieren, die nach Vorwissen, wohl am meisten für das Modelltraining beitragen. Wenn jedoch kein Vorwissen besteht, gibt es weitere Selektions-Methoden: Wrapper-, Filter- und Embedded. Wrapper-Methoden „binden“ ein ML-Modell zur Bestimmung der wichtigsten Features ein, indem ein Modell mit Teilmengen trainiert wird, die jeweils verschiedene Eingabemerkmale enthalten und dann die ausgewählt wird, die zur besten Modellleistung führt (was jedoch zu einem erheblichen Rechenaufwand führen kann) (Genender-Feltheimer, 2018, S. 115). Mit Filtermethoden wird versucht die Features nach ihrer Wichtigkeit zu bewerten, indem beispielsweise eine Korrelationsanalyse durchgeführt wird, die die Abhängigkeit zwischen einer Eingangsvariable und der Zielvariable zeigt oder die Varianz als Maß für die Wichtigkeit nimmt, ohne dass ein Lernprozess stattfindet (Hira & Gillies, 2015, S. 2). Embedded-Methoden integrieren die Merkmalsauswahl als Teil des Lernprozesses und gewichten sie während des Trainings basierend auf ihrem Beitrag zu den Vorhersagen (Genender-Feltheimer, 2018, S. 115).

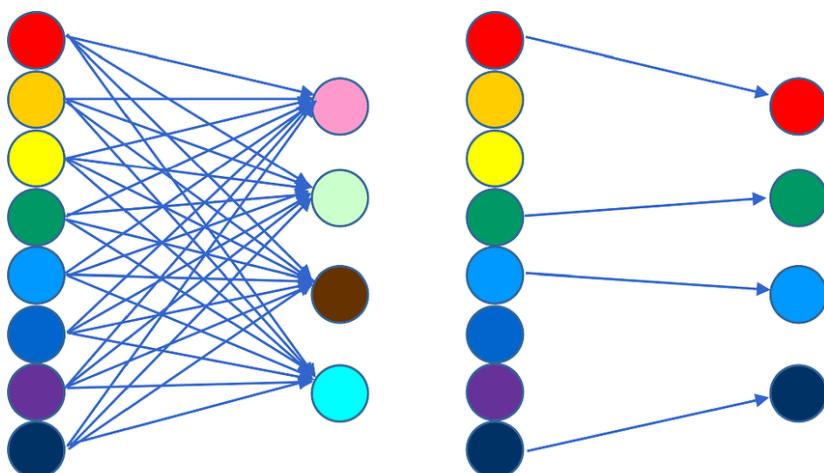
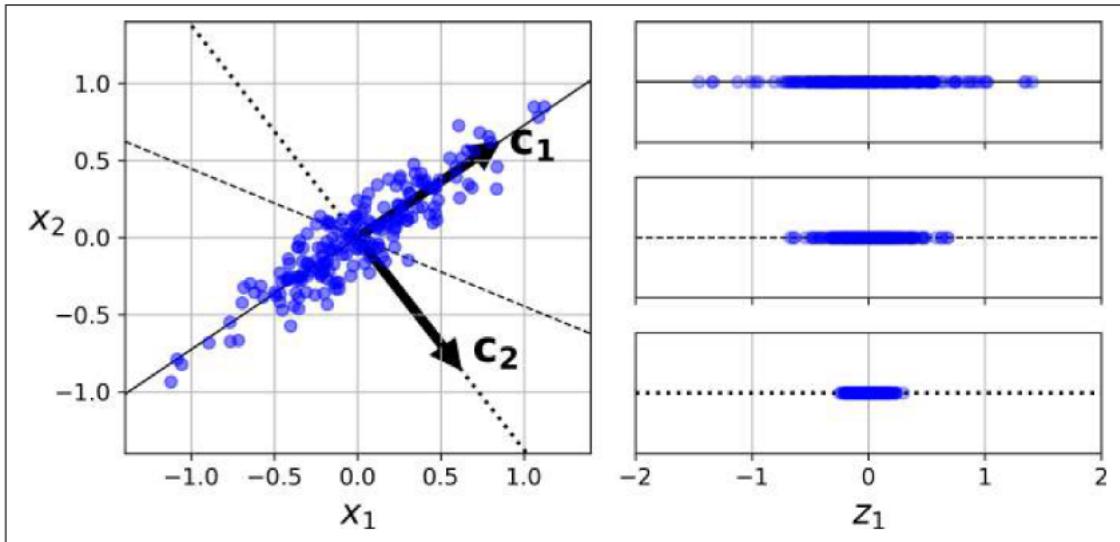


Abbildung 2.2: Veranschaulichung der Dimensionsreduktion: Feature-Extraction (links) und Feature-Selection (rechts); Quelle: (Ding, Kui, & Weihong, 2020)

Bei der Feature-Selection wird somit die Dimension durch Auswahl wichtiger Merkmale reduziert, wobei diese reduzierten Features in ihrer originalen Form – als Teilmenge der originalen Merkmale – beibehalten werden. Bei der Feature-Extraction werden neue Merkmale erstellt, als Kombination bzw. Transformation der ursprünglichen Merkmale, wie in Abbildung 2.2 zu sehen. Dabei kann die Transformation linear oder nicht linear stattfinden kann. (Hira & Gillies, 2015, S. 6). Die meisten Verfahren, die solche Transformationen nutzen, sind unterteilbar in die, die Projektionen anwenden, wie PCA oder die, die auf Manifold-Learning basieren, wie t-SNE und UMAP (Géron, 2019, S. 216).

## 2.1.5 Principal Component Analysis (PCA)

Die Principal Component Analysis (PCA) ist eine klassische Methode zur Dimensionsreduktion und kommt aus dem Bereich der multivarianten Statistik. PCA zielt darauf ab, die Varianz in den Daten zu maximieren, indem wenige neue Variablen erstellt werden, die die relevanten Informationen beschreiben. Die neuen Variablen sind eine Kombination der ursprünglichen Merkmale, die am meisten zur Varianz beitragen und werden als Hauptkomponenten bezeichnet. Das beruht auf der Annahme, dass die Beibehaltung von Merkmalen, die die meiste Varianz in den Daten aufweisen, wichtiger ist als der Verlust weniger variabler Merkmale. Dafür wird zunächst eine Hyperebene gesucht, auf der die Daten, darauf projiziert, die meiste Varianz beibehalten (Géron, 2019, S. 222)



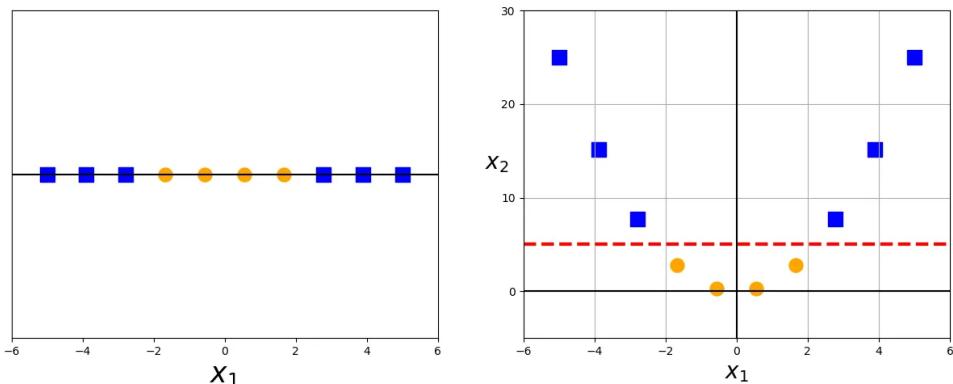
*Abbildung 2.3: Veranschaulichung von PCA;  
Bestimmung der Achsen, dessen Projektion zur meisten Varianz beiträgt;  
Quelle: ( (Géron, 2019, S. 222), Figure 8-7)*

Die Abbildung 2.3 zeigt ein einfaches Beispiel von drei Achsen, als eindimensionale Hyperebenen, auf die die Daten projiziert werden. Da die erste Achse  $C_1$  die meiste Varianz im Vergleich zu den anderen zwei beibehält, wird diese als Projektionsebene ausgewählt. Je nachdem wie viele Komponenten durch die Reduktion entstehen sollen werden weitere Achsen gesucht, die die nächstgrößere Varianz bieten, unter der Voraussetzung, dass diese orthogonal zu den vorigen bestimmten ist, was im Beispiel die Achse  $C_2$  wäre (Géron, 2019, S. 223).

---

Konkret wird dafür erst die Kovarianzmatrix der Originaldaten berechnet, um die Beziehungen zwischen verschiedenen Merkmalen aufzuzeigen. Danach werden die Eigenwerte und Eigenvektoren der Matrix berechnet. Die Eigenwerte repräsentieren dabei die Menge der Varianz in Richtung der entsprechenden Eigenvektoren. Danach werden die Eigenwerte in absteigender Reihenfolge sortiert und die zugehörigen Eigenvektoren als Hauptkomponenten gewählt. Die ersten  $k$  Hauptkomponenten, mit der größten Varianz, werden beibehalten und bilden die Koeffizienten der Linearkombination für die Berechnung der neuen Variablen. Um die Daten in einen neuen Raum zu transformieren, wird jeder Datenpunkt dabei durch die Gewichtung seiner Merkmale mit den Hauptkomponenten repräsentiert. (Ezukwoke & Zareian, 2019, S. 2).

Die nicht-lineare Erweiterung ist der Kernel-PCA. Das Vorgehen ist ähnlich zum Klassischen-PCA, nur das zu Beginn die Datenpunkte durch die Kernel-Funktion, temporär in einen hochdimensionalen Raum, den „Feature Space“ transformiert werden. Der "Trick" besteht darin, dass Kernelmethoden die Daten durch eine Reihe von paarweisen Ähnlichkeitsvergleichen, repräsentieren. Dadurch müssen die Daten nicht explizit in den höherdimensionalen Raum transformiert werden (was Rechenaufwand spart) und es muss auch nicht bekannt sein, wie die eigentliche Transformation berechnet werden müsste (Géron, 2019, S. 160).



*Abbildung 2.4: Prinzip des Kernel-Tricks:  
Transformation vom eindimensionalem zum linear separierbaren zweidimensionalen Raum;  
Quelle: (Wilimitis, 2018)*

Die Abbildung 2.4 zeigt eine einfache Transformation der Daten durch Quadrierung. Im eindimensionalen Raum sind die Daten nicht linear separierbar, im zweidimensionalen Raum jedoch schon. Dabei gibt es verschiedene Kernel-Methoden, wie Linear-, Sigmoid- oder RBF-Kernel, die die Daten entsprechend transformieren (Ezukwoke & Zareian, 2019, S. 4). Durch die implizite Projektion sollen die Daten im neuen Raum durch eine Hyperebene trennbar sein, sodass Klassen unterscheidbar werden. Kernel-PCA transformiert daher nichtlineare Strukturen in einen Raum, in dem sie linear separierbar sind. Denn dann kann das klassische PCA auf diesen höherdimensionalen Raum angewandt werden, um wieder die Hauptkomponenten zu identifizieren (Géron, 2019, S. 228).

## 2.1.6 Manifold Learning

Das Manifold-Learning ist ein Ansatz im maschinellen Lernen, um höherdimensionalen Daten in einen niedrigdimensionalen Raum zu transformieren, durch Identifikation des „Manifold“. Das Manifold bezieht sich auf die „Mannigfaltigkeit“. Diese beschreibt einen topologischen Raum, wie der euklidische Raum und wird als abstrakte geometrische Struktur gesehen. Es wird angenommen, dass die relevanten Informationen in Daten in einer Struktur (dem Manifold) vorliegen, die durch weniger Dimensionen repräsentiert werden kann als der ursprüngliche Raum (Skliar & Weiler, 2023, S. 7). Ein gängiges Beispiel zur Veranschaulichung der Mannigfaltigkeit ist die Betrachtung der Erdoberfläche.

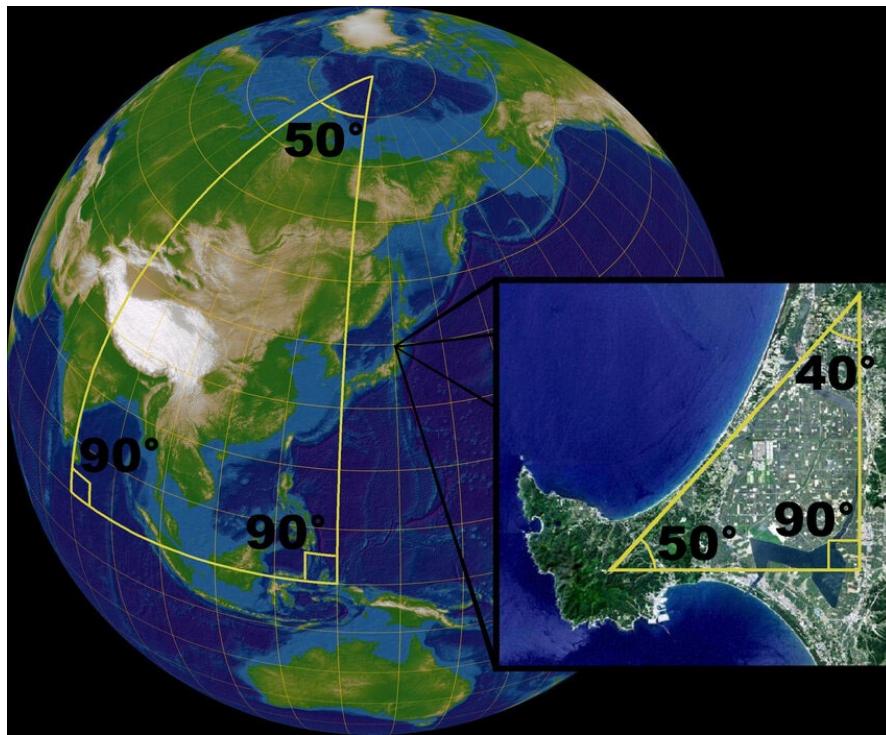


Abbildung 2.5: Erdoberfläche zur Veranschaulichung eines Manifold;  
Quelle: (Rohwedder, 2007), (Skliar & Weiler, 2023, S. 7)

Die Erdoberfläche der Erdkugel, als 3D-Objekt, kann durch eine Weltkarte, als 2D-Objekt, nicht mit der gesamten Oberfläche dargestellt werden, ohne bestimmte Bereiche abzuschneiden. Stattdessen gibt es mehrere kleinere Karten von Gebieten, die in ihrer Gesamtheit die Erdoberfläche beschreiben. Dabei sind alle Karten, und daher auch das Manifold (als gemeinsame Struktur), zweidimensional und beschreiben gemeinsam die Mannigfaltigkeit – also wie die lokalen Karten miteinander zusammenhängen. Konkret ist daher die Mannigfaltigkeit eine spezielle Art von topologischem Raum, der sich bei Betrachtung kleiner Teile wie ein gewöhnlicher Raum verhält, auch wenn er im ursprünglichen Maßstab komplizierter sein kann (Géron, 2019, S. 220).

Manifold-Learning-Algorithmen wie t-SNE und UMAP versuchen diese eingebettete Struktur in den hochdimensionalen Daten zu erfassen und sie im niedrigdimensionalen Raum darzustellen, um die Muster und Beziehungen visualisieren und analysieren zu können und Daten zu clustern.

---

### 2.1.6.1 T-Distributed Stochastic Neighbor Embedding (t-SNE)

t-distributed Stochastic Neighbor Embedding (t-SNE) gehört zu den nicht-linearen Manifold-Learning-Methoden und wird häufig zur Visualisierung hochdimensionaler Daten verwendet. t-SNE versucht die Nachbarschaftsbeziehungen zwischen Datenpunkten im niedrigdimensionalen Raum zu berücksichtigen, indem die Entferungen (gemessen durch Euklidische Distanz) dargestellt werden als Wahrscheinlichkeiten (Géron, 2019, S. 235). Erst werden die Ähnlichkeiten zwischen Nachbar-Datenpunkten im ursprünglichen hochdimensionalen Raum berechnet und um jeden Datenpunkt eine Gaußsche Verteilung modelliert. Die Anzahl der zu berücksichtigen Nachbarn wird durch einen Hyperparameter, der Perplexität, festgelegt, wobei die Perplexität monoton mit der Varianz zunimmt (Van der Maaten & Geoffrey, 2008, S. 2582). Nachdem für jeden Datenpunkt die Dichte der Gaußverteilung gemessen und die Ergebnisse normalisiert wurden, ergeben sich die bedingten Wahrscheinlichkeiten für den hochdimensionalen Raum. Um die Daten im niedrigdimensionalen Zielraum darzustellen, werden sie erst zufällig – also randomisiert initialisiert - verteilt und dann wie zuvor die Ähnlichkeiten der Datenpunkte im neuen Raum berechnet. Allerdings wird für den niedrigdimensionalen Raum keine Gaußverteilung, sondern die t-Student-Verteilung verwendet (Van der Maaten & Geoffrey, 2008, S. 2583). Der Algorithmus passt dabei die Positionen der Datenpunkten im Zielraum so an, dass die Wahrscheinlichkeiten im Zielraum, möglichst gut zu denen im ursprünglichen Raum übereinstimmen. Das geschieht durch die Minimierung der Kullback-Leibler-Divergenz (KL-Divergenz) zwischen den Wahrscheinlichkeitsverteilungen im Original- und Zielraum. Die KL-Divergenz misst, wie eine Wahrscheinlichkeitsverteilung  $P$  von einer Verteilung  $Q$  relativ abweicht - um den Informationsverlust zu messen, der auftritt, wenn  $Q$  durch  $P$  modelliert wird:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right) \quad (2.3)$$

Das Ergebnis ist eine Darstellung, in der ähnliche Datenpunkte nahe beieinander und unähnliche Datenpunkte weiter voneinander entfernt sind (Géron, 2019, S. 235). Deswegen wird generell gesagt, dass t-SNE besser darin ist, lokale Strukturen zu visualisieren, da sie die Nachbarschaftsbeziehungen erhalten, aber dadurch die globalen Strukturen vernachlässigt. Das bedeutet, dass Datenpunkte innerhalb eines Clusters sich tatsächlich sehr nahe sind, aber an den Entferungen zwischen den Clustern nicht erkannt werden kann, wie unähnlich sie sich sind – also, ob bestimmte Cluster zueinander ähnlicher sind als andere. Aufgrund dessen ist t-SNE sensibel gegenüber dem Fluch der Dimensionalität und nicht so gut darin, globale Strukturen beizubehalten (Van der Maaten & Geoffrey, 2008, S. 2598).

---

### 2.1.6.2 Uniform Manifold Approximation and Projection (UMAP)

Die Uniform Manifold Approximation and Projection (UMAP) ist im Vergleich zu t-SNE und PCA eine jüngere Methode. Sie basiert auf dem Manifold Learning und versucht die Manifold-Struktur in den hochdimensionalen Daten zu identifizieren, sie im niedrigdimensionalen Raum darzustellen und möglichst die lokalen und globalen Beziehungen der Datenpunkte beizubehalten. Dafür konstruiert UMAP zuerst ein hochdimensionales Graphen-Modell, basierend auf dem k-Neighbor-Algorithmus, mit Gewichtung der Kanten. Dann wird eine niedrigdimensionale Darstellung gesucht, die eine ähnliche topologische Struktur aufweist (McInnes & Healy, 2018, S. 13). Das bedeutet, dass sie im Gegensatz zu t-SNE die Daten als topologisches Konstrukt darstellt, statt als Wahrscheinlichkeitsverteilung (McInnes & Healy, 2018, S. 39). Als Verlustfunktion verwendet UMAP die Cross Entropy:

$$CE(P, Q) = - \sum_{x \in X} P(x) \log(Q(x)) \quad (2.4)$$

Wie t-SNE minimiert UMAP die Verlustfunktion mithilfe vom Gradient Descent. Das bedeutet, Gradient Descent beginnt mit einer initialen Konfiguration von Punkten und verbessert dann in jeder Iteration die Verteilung, um die Verlustfunktion zu verringern. Im Gegensatz zu t-SNE ist aber UMAP nicht zufällig initialisiert, sondern basiert auf den Laplacian Eigenmaps, was auch der Grund dafür ist, dass UMAP die globalen Strukturen besser erhält, denn wie im Paper von (Kobak & Linderman, 2019, S. 1) gezeigt, verhält sich UMAP bei gleicher Random-Initialisierung ähnlich schlecht wie t-SNE für die globale Erhaltung und t-SNE ähnlich gut, wenn eine spezifische Initialisierung stattgefunden hat. Allerdings zeigt (Oskolkov, 2019), dass die Optimierung der Initialisierung das grundlegende Problem in t-SNE - globale Strukturen zu erhalten – nicht lösen könnte:

„As long as tSNE uses KL-divergence as cost function, it can not compete against  
UMAP in global distance preservation“.

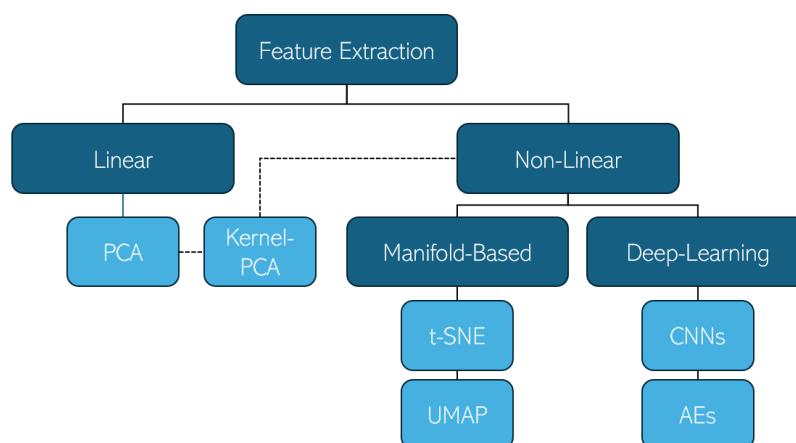
### 2.1.6.3 Multidimensional Scaling (MDS)

Multidimensional Scaling (MDS) ist eine Methode zur Dimensionsreduktion, die ähnlich wie t-SNE versucht, die hochdimensionalen Daten in einen niedrigdimensionalen Raum darzustellen. Das passiert durch Verwendung der „wahrgenommenen Ähnlichkeiten“, gemessen auf verschiedene Skalen. Das heißt, dass MDS, Informationen zu den Ähnlichkeiten braucht, diese aber in unterschiedlichen Strukturen vorliegen können, wie Häufigkeit oder Korrelationen. Meist wird mit der euklidischen Distanz eine Ähnlichkeitsmatrix bzw. Distanzmatrix erstellt (Jäckle, 2017, S. 180). Auf der Datengrundlage erstellt MDS dann eine „Konfiguration“, die die neuen Koordinaten der ursprünglichen Datenpunkte im niedrigdimensionalen Raum darstellen und damit die Ähnlichkeiten aller Daten in ihrer Gesamtstruktur beschreibt (Jäckle, 2017, S. 182).

Dabei gibt es zwei Arten von MDS. Das metrische MDS versucht die Datenpunkte mithilfe der Ähnlichkeits- oder Distanzmatrix so in den niedrigdimensionalen Raum zu transformieren, dass die Distanzen zwischen den Datenpunkten so gut wie möglich erhalten bleiben. Das heißt, die paarweisen Distanzen zwischen den Datenpunkten im Originalraum und die im Zielraum sollen so gut wie möglich übereinstimmen. Im Gegensatz dazu geht es beim nicht-metrischen MDS darum, die Rangordnung der Ähnlichkeiten zwischen den Objekten zu bewahren, ohne die genauen Distanzen zu reproduzieren. Das bedeutet, dass der Fokus darauf liegt, eine Konfiguration zu erstellen, in der die Reihenfolge der Distanzen zwischen den Punkten möglichst der Rangordnung der wahrgenommenen Ähnlichkeiten entspricht (Jäckle, 2017, S. 180).

## 2.1.7 Kernel-PCA, t-SNE und UMAP

Alle drei Methoden können zur Dimensionsreduktion oder Visualisierung hochdimensionaler Daten angewandt werden und auf nichtlineare Daten arbeiten, erhalten jedoch die Aspekte der ursprünglichen Daten unterschiedlich bei. Kernel-PCA ist durch die Erhaltung der maximalen Varianz, eher darauf ausgelegt globalen Strukturen zu erfassen (Pani, 2022). T-SNE hingegen priorisiert die Nachbarschaftsbeziehungen und ist weniger gut darin globale Strukturen zu erhalten. UMAP versucht beide – lokale und globale – Strukturen gleichermaßen zu erhalten und erzielt dabei oft bessere Ergebnisse. Den Rechenaufwand betrachtet, ist Kernel-PCA etwas effizienter als t-SNE und UMAP, da kein iterativer Prozess stattfindet. t-SNE ist aufgrund der iterativen Wahrscheinlichkeitsberechnungen bei großen Datenmengen sehr langsam, da die Rechenkomplexität quadratisch mit der Anzahl an Datenpunkten steigt. (Van der Maaten & Geoffrey, 2008, S. 2593). Sie wird weniger zu Dimensionsreduktion und häufiger zur Visualisierung verwendet. UMAP skaliert besser als t-SNE, und kann auch zur Visualisierung von Cluster verwendet werden (McInnes & Healy, 2018, S. 39).



*Abbildung 2.6: Übersicht der Feature-Extraction Methoden;  
Angelehnt, an Quelle: ( Berahmand, Daneshfar, Salehi, Li, & Xu, 2024, S. 4), Fig.1).*

Ein anderer Bereich zur nicht-linearen Feature-Extraction ist das Deep-Learning mit ML-Modellen, wofür klassischerweise Convolutional Neural Networks (CNNs) verwendet werden, als überwachtes Lernverfahren, aber auch Autoencoder (AEs), die unüberwacht arbeiten.

## 2.2 Autoencoder (AE)

Autoencoder (AE) sind neuronale Netze und werden als nichtlineare Methode zur Dimensionsreduktion verwendet. Das Modell wird darauf trainiert, die Daten in einem niedrigdimensionalen Raum zu komprimieren und dann wieder zu rekonstruieren (Michelucci, 2022, S. 1). Während des Trainings wird das Netzwerk so optimiert, dass die Differenz zwischen der ursprünglichen Eingabe und der rekonstruierten Ausgabe minimiert wird, als „Reconstruction Loss“. Dieser Rekonstruktionsfehler wird als Maß verwendet, um zu beurteilen, wie gut das Modell die Eingabedaten reproduzieren kann, wofür häufig der Mean Squared Error (MSE) verwendet wird (Michelucci, 2022, S. 7).

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

Dabei beschreibt  $n$  die Anzahl der Datenpunkte,  $x$  die Originaldaten und  $\hat{x}$  die Rekonstruktionen. MSE misst damit die durchschnittliche quadratische Abweichung zwischen den tatsächlichen und den vorhergesagten Werten. Je niedriger der MSE, desto besser die Vorhersagegenauigkeit, bzw. im Falle von Autoencoder, desto besser die Rekonstruktion. Das Modell lernt so, eine interne Repräsentation der Daten zu erstellen, ohne dass Labels benötigt werden. Deswegen werden Autoencoder generell zu den unüberwachten Lernverfahren gezählt.

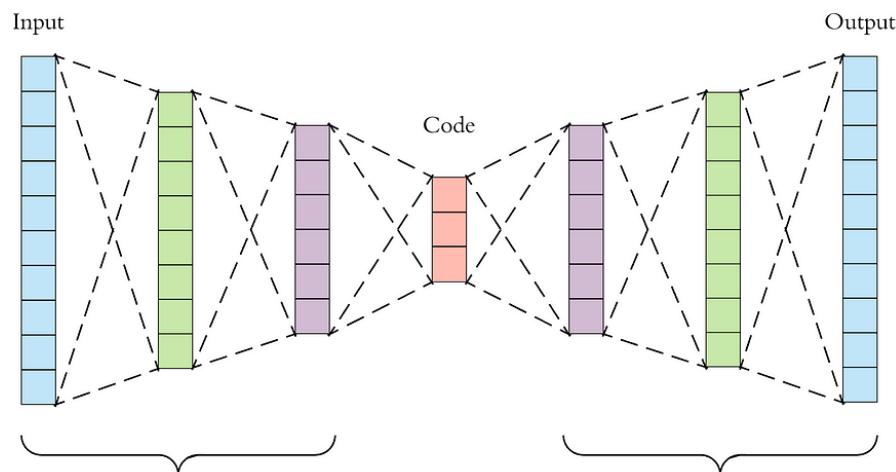


Abbildung 2.7: Klassisches Autoencoder-Modell mit Encoder und Decoder als Sub-Modelle; Quelle: (Dertat, 2017)

Wie Abbildung 2.7 zeigt haben Autoencoder die Struktur einer Sanduhr und sie bestehen aus zwei Sub-Modellen, dem reduzierenden Encoder und dem expandierenden Decoder, wobei beide Komponenten durch ein oder mehrere Hidden-Layers verbunden sind. Der Encoder, nimmt die hochdimensionalen Eingabedaten entgegen und komprimiert sie durch Transformationen in eine kompakte Darstellung, die als Latenter Raum, „Bottleneck“ oder „Hidden Layer“ bezeichnet wird (Michelucci, 2022, S. 13). In diesem werden die Eingabedaten effizient repräsentiert und gleichzeitig Rauschen entfernt. Man bezeichnet das Ergebnis des Encoders auch als Feature-Vektor, da dieser die wichtigsten Merkmale, die

---

das Modell durch das Training gelernt hat, enthält. Der Decoder nimmt wiederum die kodierten Daten vom latenten Raum und versucht die Eingabedaten zu rekonstruieren. Das Output-Layer des Decoders muss daher zwangsläufig die gleiche Dimension haben, wie die Eingabedaten. Der Decoder hat dabei typischerweise genauso viele Hidden-Layers wie der Encoder. Bei nur einem Hidden-Layer ist der Output von diesem auch gleichzeitig der Latente Raum, der die komprimierte Darstellung der Original-Daten enthält. Bei mehreren Hidden-Layer, wird der Output der letzten Encoder-Schicht als Latenter Raum bezeichnet. Für jeden Hidden-Layer muss die die Größe angegeben werden, wobei sie mit jedem weiteren Hidden-Layer im Encoder abnimmt und symmetrisch im Decoder erhöht wird (Michelucci, 2022, S. 2).

AEs mit mehr als einem Hidden-Layer in den Encodern und Decodern, werden als „Deep Autoencoders“ bezeichnet. Das einfachste Autoencoder Modell kann deswegen nur durch drei Layer beschrieben werden: Einem Input-Layer, dass die Eingabedaten entgegennimmt, einem Hidden Layer, welches die Daten-Kodierung darstellt und einem Output-Layer, der die rekonstruierten Daten wieder ausgibt (Berahmand, Daneshfar, Salehi, Li, & Xu, 2024, S. 11).

## 2.2.1 Feature-Vektor im Latenten Raum

Da ein AE aus den Submodellen Encoder und Decoder besteht können diese auch isoliert betrachtet werden. Werden Daten durch den Encoder geführt, werden sie im Latenten Raum als Kodierungen der Eingabedaten gesehen, dargestellt durch Feature-Vektoren. Die Kodierungen enthalten die gelernt wichtigsten Features als komprimierte Form der Originaldaten. Die Form der Feature-Vektoren hängt von der Architektur des AEs ab. Im Kontext von Bilddaten, für einen Convolutional Autoencoder (CAE), sind die Eingabedaten der Form  $(n, height, width, channels)$  wobei  $n$  die Anzahl der Bilder,  $height$  und  $width$  die Bildabmessungen und  $channels$  die Anzahl der Farbkänele darstellen. Somit haben die resultierenden Feature-Vektoren die Form  $(n, fm_{height}, fm_{width}, dim)$ , wobei nun  $n$  die Anzahl der Feature-Vektoren,  $fm_{height}$  und  $fm_{width}$  die Feature-Map-Abmessungen und  $dim$  die Dimensionen dieser repräsentieren. Anderes gesagt, entstehen für die  $n$  Eingabebilder  $n$  Feature-Vektoren, wobei jeder Feature-Vektor  $dim$  Feature-Maps, der Größe  $fm_{height} * fm_{width}$ , enthält (Michelucci, 2022, S. 23).

## 2.2.2 Klassischer Autoencoder (AE)

Der klassische Autoencoder wird auch als Fully-Connected-Autoencoder, bezeichnet, da er Dense-Layer (auch Fully Connected Layer) zur Transformation bzw. Reduktion der Eingabedaten verwendet. Durch Dense-Layer werden die Neuronen, mit allen Neuronen des vorherigen Layer verbunden (O'Shea & Nash, 2015, S. 8). Meistens werden hier nichtlineare Funktionen wie Rectified-Linear-Unit (ReLU) oder die Sigmoid-Funktion verwendet (Berahmand, Daneshfar, Salehi, Li, & Xu, 2024, S. 11). Der Encoder des klassischen AE besteht daher aus dem Input-Layer gefolgt von ein oder mehreren Dense-

Layers, wobei der Output des letzten Encoder-Dense Layer der Latente Raum darstellt, dessen Größe durch die Angabe der Unit-Anzahl bestimmt wird. Der Decoder besteht aus der gleichen Anzahl an Dense Layer, gefolgt vom Output-Layer. Da Dense-Layer auf eindimensionalen Daten arbeiten, müssten Bilder durch eine Flatten-Operation in eindimensionale Vektoren umgewandelt werden. Jedoch können dadurch die räumlichen Beziehungen der Daten verloren gehen, weshalb sich Convolutional Autoencoder (CAEs) als effizienter im Umgang mit Bildern erwiesen haben (Michelucci, 2022, S. 24).

### 2.2.3 Convolutional Autoencoder (CAE)

Convolutional Autoencoder (CAE) ist eine Variante, die speziell für die Verarbeitung von Bildern ausgelegt ist. Im Vergleich zu den klassischen AE mit Dense Layer, verwendet ein CAE Convolutional Layers, um räumliche Hierarchien, Abhängigkeiten und Muster zu erfassen und besteht typischerweise aus einer Reihe von Convolutional Layers gefolgt von Pooling-Layer oder einem weiteren Convolutional Layer mit Stride, der die räumliche Dimension der Feature-Maps reduziert, wie zu sehen in Abbildung 2.8 (Berahmand, Daneshfar, Salehi, Li, & Xu, 2024, S. 21-22).

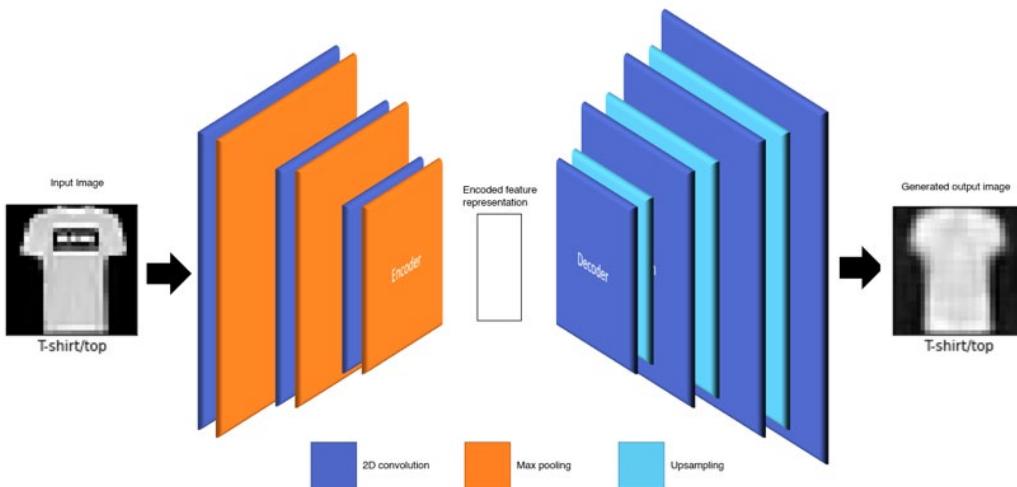


Abbildung 2.8: Modellarchitektur eines Convolutional Autoencoder (CAE);  
Quelle: (Lanham, 2023)

In einer Studie von (Springenberg, Dosovitskiy, Brox, & Riedmiller, 2015) wurde gezeigt, dass die Verwendung von Strided Convolutions anstelle von Max-Pooling-Layern vorteilhaft ist:

“... when pooling is replaced by an additional convolution layer with stride  $r = 2$   
performance stabilizes and even improves on the base mode”

Ein Convolutional Layer führt Faltungsoperationen auf den Eingabebildern mithilfe von Filtern aus. Jeder Filter ist eine kleine Matrix von Gewichtungen, der schrittweise über das Bild verschoben wird, wobei an jeder Position die gewichtete Summe der Werte berechnet wird und diese als neuer Wert in der räumlich reduzierten Feature-Map eingetragen wird. Dadurch erkennt der Filter bestimmte

---

Merkmale, wie Kanten, Texturen oder andere lokale Muster. Der Stride ist ein zusätzlicher Parameter, der bestimmt, wie weit der Filter bei jeder Anwendung verschoben wird. Ohne Angabe vom Stride wird er auf den Default-Wert von 1 gesetzt, wodurch der Filter immer um einen Pixel verschoben wird. Bei größerem Stride, beispielsweise einem Stride von 2, würde immer ein Pixel übersprungen werden, wodurch sich die räumliche Dimension halbiert. (O'Shea & Nash, 2015, S. 7). Ein Pooling-Layer würde die Dimension reduzieren, durch eine Aggregation der lokalen Regionen der Eingabedaten. Im Max-Pooling wird der maximale Wert in einem lokalen Bereich ausgewählt, wodurch das Vorhandensein bestimmter Muster hervorgehoben wird. Im Average-Pooling wird der Durchschnitt der Werte im lokalen Bereich berechnet. Im Gegensatz zu Strided Convolutions zur Reduktion der räumlichen Dimension, haben Pooling-Layer keine Gewichtungen und dadurch keine trainierbaren Parameter (O'Shea & Nash, 2015, S. 8). Typischerweise würde mit jeder Halbierung der räumlichen Dimension, die Anzahl der Filter erhöht werden, nach klassischem CNN-Prinzip. Die Strategie wird häufig in der Bildklassifizierung, Objekterkennung oder Segmentierung verfolgt, bei denen mit zunehmender Tiefe des Netzwerks die Anzahl der Filter erhöht wird, für eine tiefe Merkmalsextraktion. Eine stetige Erhöhung der Filteranzahl schließt das Ziel der Dimensionsreduktion nicht aus, aber eine stetige Reduktion bei gleichzeitiger Halbierung der räumlichen Dimensionen kann den Prozess verstärken. Daher gibt es sowohl CAEs bei dem die Filteranzahl mit jeder räumlichen Reduktion erhöht, aber auch welche bei denen sie reduziert wird, wie beispielsweise in (Chollet, 2016).

Der Encoder nimmt die Eingabedaten durch das Input-Layer entgegen, wendet ein oder mehrere Convolutional-Layer zum Feature-Learning an und reduziert dann die Dimension entweder mit einer Strided Convolution oder einer Pooling-Funktion. Das kann so lange wiederholt werden, bis die gewünschte Größe des Latenten Raums erreicht wurde. Die Struktur und Funktionen des Decoders sind spiegelbildlich zum Encoder. Dieser nimmt als Input-Layer die kodierten Daten des Encoders vom Latenten Raum. Dann können wieder Convolutional-Layers zum Feature-Learning angewandt werden gefolgt von einer Erhöhung der Dimension durch Upsampling-Funktionen oder ein Transposed-Convolutional-Layer mit Stride (Berahmand, Daneshfar, Salehi, Li, & Xu, 2024, S. 21). Der Transposed-Convolutional-Layer, als inverse Operation der Convolution, führt die Deconvolution aus. Die räumliche Dimension der Feature-Maps wird erhöht, bei einem Stride größer 1 – also eine Verdopplung bei einem Stride von 2. Alternativ können zur Erhöhung Upsampling-Funktionen verwendet werden, wie Interpolation-Algorithmen, die jedoch wie die Pooling-Funktionen keine lernbaren Parameter haben: Nearest-Neighbor-Interpolation nimmt für jeden Pixelwert im vergrößerten Raum den Wert des nächsten Nachbarn aus dem ursprünglichen Raum. Die Bilinear-Interpolation verwendet eine gewichtete Durchschnittsberechnung für mehrere nächste Nachbarn im ursprünglichen Raum, um den Wert im vergrößerten Raum zu bestimmen, was glattere Übergänge im Gegensatz zum Nearest-Neighbor-Upsampling erzeugt (Kolarik, Burget, & Riha, 2019).

---

## 2.2.4 Aktivierungsfunktionen

### 2.2.4.1 LeakyReLU

Die Rectified Linear Unit (ReLU) ist eine nichtlineare Aktivierungsfunktion, die häufig in Convolutional Layer verwendet wird und definiert ist als Max-Operation:

$$f(x) = \max(0, x) \quad (2.6)$$

Sie gibt für negative Eingaben Null zurück und für positive Eingaben den unveränderten positiven Eingabewert. Das bedeutet, dass ReLU für alle positiven Werte linear ist, wodurch der Gradient groß genug ist (bei aktivem Neuron), um einen Einfluss auf die Gewichtsaktualisierung während der Backpropagation zu haben. Die Ableitung von ReLU ist entweder 0 für negative Eingaben oder 1 für positive Eingaben. Und da die Ableitung für positive Werte konstant bleibt, wird der Gradient nicht durch die Aktivierungsfunktion selbst verringert (solange er nicht durch negative Eingaben auf 0 gesetzt wird) (Michelucci, 2022, S. 7). Während des Trainings werden die Gewichte und Biases angepasst, um den Rekonstruktionsfehler im Gradient-Descent zu minimieren. AEs verwenden im Gradient-Descent die ReLU-Funktion oft in den Hidden-Layers, sowohl im Encoder als auch im Decoder. LeakyReLU ist eine Variante, um das Problem des „Neuronensterbens“ während des Trainings anzugehen, das auftreten kann, wenn Neuronen bei negativen Eingaben auf null gesetzt werden.

$$f(x) = \begin{cases} x & \text{für } x > 0 \\ a x & \text{für } x \leq 0 \end{cases} \quad (2.7)$$

Dabei werden negative Eingaben mit einem Faktor  $a$  multipliziert, der als „Leckparameter“ bezeichnet wird und standardmäßig einen Wert von 0,01 hat. Dadurch wird für negative Eingaben immer ein kleiner Gradient erzeugt, wodurch das Neuron aktiv bleibt (Bing, Naiyan, Tianqi, & Mu, Empirical Evaluation of Rectified Activations in Convolutional Network, 2015, S. 2). Nach (Bing, Naiyan, Tianqi, & Mu, 2015) kann dies für tiefe Netze eine Verbesserung zu ReLU sein.

### 2.2.4.2 Sigmoid-Funktion

Die Sigmoid-Funktion ist eine Aktivierungsfunktion die meist für Aufgaben wie Binary-Klassifikation verwendet wird, da sie jede reelle Zahl in einen Wert zwischen [0,1] konvertiert, der als Wahrscheinlichkeit interpretiert werden kann, beschrieben als:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.8)$$

Da die Ableitung für sehr hohe oder sehr niedrige Eingabewerte nahezu (0) ist, sie besonders anfällig für das Vanishing Gradient:

$$\sigma'(x) = \sigma(x)(1 - \sigma) \quad (2.9)$$

Da während der Backpropagation der Gradient der Verlustfunktion durch die Kettenregel berechnet wird – wiederholte Multiplikation der Ableitung der Sigmoid-Funktion - wird der Gradient in den unteren Schichten des Netzwerks extrem klein. Dadurch lernen die Schichten nur sehr langsam, da die Gewichte nicht signifikant verändert werden oder gar nicht. Die Sigmoid-Funktion wird oft im Output-Layer des Decoders eingesetzt, da Bilddaten normalisiert zwischen [0,1] liegen, wodurch sie sich eignet die Rekonstruktionen zurück in den Wertebereich zu konvertieren (Michelucci, 2022, S. 7).

## 2.3 Residual-Connections

Residual Networks (ResNets) sind spezielle neuronale Netze, die sogenannte Residual-Connections, auch Shortcut- oder Skip-Connections genannt, verwenden, um das Problem des Vanishing Gradient anzugehen. Wie in der nachfolgenden Abbildung 2.9 zu sehen, wird der Input, während dem Training umgeleitet, indem die Eingabe direkt an die Ausgabe einer späteren Schicht übergeben wird. Durch die Weiterleitung lernt das Netz die Identitätsfunktion. Wenn dann eine Änderung an der Ausgabe der Schicht vorgenommen werden soll während der Backpropagation, kann das Modell sich auf die Differenz zwischen der erwarteten und der aktuellen Ausgabe richten. Durch den alternativen Informationsfluss wird es einfacher Änderungen an den Gewichten vorzunehmen und die Gradienten durch das Netzwerk zu propagieren, insbesondere bei sehr tiefen Architekturen.

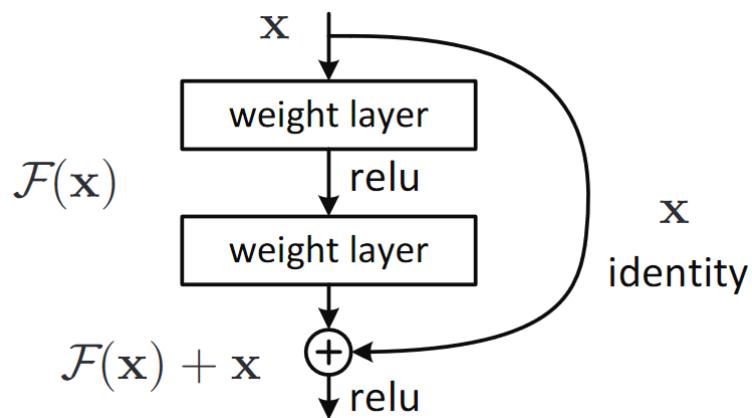


Abbildung 2.9: Residual-Learning-Block;  
Quelle: ( (Kaiming, Xiangyu, Shaoqing, & Jian, 2015, S. 2), Fig.2)

Dadurch können ResNets viel tiefer sein als herkömmliche neuronale Netze, ohne dass das Problem des Verschwindens von Gradienten auftritt, wobei die Identity-Shortcuts auch nicht komplex sind und keine Parameter enthalten (Kaiming, Xiangyu, Shaoqing, & Jian, 2015, S. 2).

---

## 2.4 Evaluierungs-Metriken

### 2.4.1 Structural Similarity Index (SSI)

Der Structural Similarity Index (SSI) ist eine Methode, um die Ähnlichkeit zwischen zwei Bildern zu messen, indem sie die strukturelle Ähnlichkeit sowie Kontrast und Helligkeit quantifiziert. Ein höherer SSI-Wert deutet auf eine größere strukturelle Ähnlichkeit hin, wobei ein Wert von 1 die maximale Ähnlichkeit darstellt und 0 keine Ähnlichkeit. Damit eignet sie sich zur Bewertung, ob CAEs die Bildstrukturen in ihren Rekonstruktionen erfassen können, (Wang, Bovik, & Sheikh, 2004).

### 2.4.2 Peak to Signal Noise Ratio (PSNR)

Die Peak to Signal Noise Ratio (PSNR) ist ein Maß, um das Verhältnis zwischen der bestmöglichen Leistung eines Signals und der Leistung des verrauschenenden Fehlers zu messen. Das Maß wird in logarithmischen Dezibel (dB) ausgedrückt. In der Regel entstehen Werte von 20 – 50 dB, wobei höhere PSNR-Werte auf eine bessere Qualität hinweisen. Ein PSNR-Wert von 20 bis 30 dB deutet darauf hin, dass das rekonstruierte Signal eine gute bis sehr gute Qualität im Vergleich zum Originalsignal hat. Je höher der PSNR-Wert ist, desto geringer ist die wahrgenommene Qualitätseinbuße im Vergleich zum Originalsignal. In der Regel wird ein PSNR-Wert von etwa 30 dB als akzeptabel für die meisten Anwendungen angesehen, während Werte über 40 dB oft als nahezu verlustfrei betrachtet werden. (Nadipally, 2019, S. 2.4.4.).

### 2.4.3 Mean Absolute Error (MAE) und Mean Squared Error (MSE)

Beide Metriken messen die durchschnittliche Differenz zwischen den Pixelwerten der Originalbilder und den rekonstruierten Bildern. Je niedriger die Werte, desto besser die Rekonstruktionen und damit die Rekonstruktionsleistung des AEs (Michelucci, 2022, S. 8). Der Mean Squared Error (MSE) misst die durchschnittliche quadratische Differenz zwischen den Originalen und Rekonstruktionen, wodurch mehr Gewicht auf größere Fehler gelegt wird, da Fehlerquadrate betrachtet werden:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2.10)$$

Der Mean Absolute Error (MAE) berechnet den durchschnittlichen absoluten Unterschied, als Maß dafür, wie der CAE im Durchschnitt Daten rekonstruieren kann, wobei jeder Fehler unabhängig seiner Größe gleichgewichtet wird und ist. Dadurch ist er weniger empfindlich gegenüber großen Fehlern:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (2.11)$$

---

## 2.5 Distanz-Metriken

Durch Distanz-Metriken kann der Abstand zwischen Datenpunkten im mehrdimensionalen Raum gemessen werden, um dadurch die Ähnlichkeit zwischen (Feature)-Vektoren darzustellen.

### 2.5.1 Euklidische Distanz

Die Euklidische Distanz ist das am häufigsten verwendete Maß, um den direkten Abstand zwischen zwei Datenpunkten in einem hochdimensionalen Raum zu messen. Bei zwei Punkten  $P$  und  $Q$  wird die Distanz mit folgender Formel berechnet:

$$d(P, Q) = \|P - Q\| = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2} \quad (2.12)$$

Je kleiner der Wert, desto ähnlicher liegen die Vektoren beieinander. Sie kann daher verwendet werden, um zu bestimmen, wie ähnlich zwei Datenpunkte im latenten Raum sind, wobei kleine Distanzen Daraufhinweisen, dass die Punkte ähnliche Merkmale haben (Harmouch, 2021).

### 2.5.2 Manhattan Distanz

Die Manhattan-Distanz, auch als L1-Norm oder City-Block-Distanz bezeichnet, ist eine Methode, um die Distanz zwischen zwei Punkten in einem Raster zu messen, indem nur horizontale und vertikale Pfade berücksichtigt werden. Statt also wie bei der euklidischen Distanz den „direkten“ und „kürzesten“ Weg zwischen den Punkten durch eine Diagonale darzustellen, summiert die Manhattan-Distanz die absoluten Differenzen ihrer Koordinaten entlang der Achsen. Sie kann auch zur Messung hochdimensionaler Vektoren der Form  $V = (v_1, v_2, \dots, v_n)$  verwendet werden. Die Distanz im  $n$ -dimensionalen Raum zwischen zwei Vektoren  $P$  und  $Q$  wird berechnet durch:

$$d(P, Q) = \sum_{i=1}^n |p_i - q_i| \quad (2.13)$$

Da sie die absoluten Differenzen summiert, fallen große Unterschiede weniger stark ins Gewicht, wodurch sie robuster gegenüber Ausreißern ist (Harmouch, 2021). Dadurch ist sie auch weniger vom Fluch der Dimensionalität betroffen als die euklidische Distanz und wird bei Distanzmessungen in hohen Dimensionen bevorzugt (Aggarwal, Hinneburg, & Keim, 2002).



### 3 Stand der Technik

Hauptanwendung finden AEs vor Allem in der Dimensionsreduktion. Neben dem Convolutional Autoencoder (CAE) (2.2.3), der sich besonders für den Umgang mit Bilddaten eignet, gibt es auch eine Menge an anderen Anwendungen, für die spezifische AE-Modelle zum Einsatz kommen. In den folgenden Abschnitten wird auf die verschiedenen AE-Typen eingegangen, die sich entwickelt haben, um spezifische Herausforderungen und Aufgaben zu bewältigen.

#### 3.1 Generative AE

Generative AEs sind eine spezielle Variante, die nicht die Dimensionsreduktion zum Ziel haben, sondern die Generierung neuer Daten, die den Eingabedaten ähnlich sind. Variational AEs (VAE) erzeugen dabei neue Datenpunkte, indem der Encoder die Eingabedaten auf eine Wahrscheinlichkeitsverteilung abbildet und nicht auf einen festen Punkt im Latenten Raum, wie klassischen AEs. Der Decoder nimmt dann aus der erlernten Verteilung zufällig ein Sample, um sie zu rekonstruieren, wie dargestellt in Abbildung 3.1. Dadurch lernt das Modell eine kontinuierliche und glatte Repräsentation des Latenten Raums (Berahmand, Daneshfar, Salehi, Li, & Xu, 2024, S. 20).

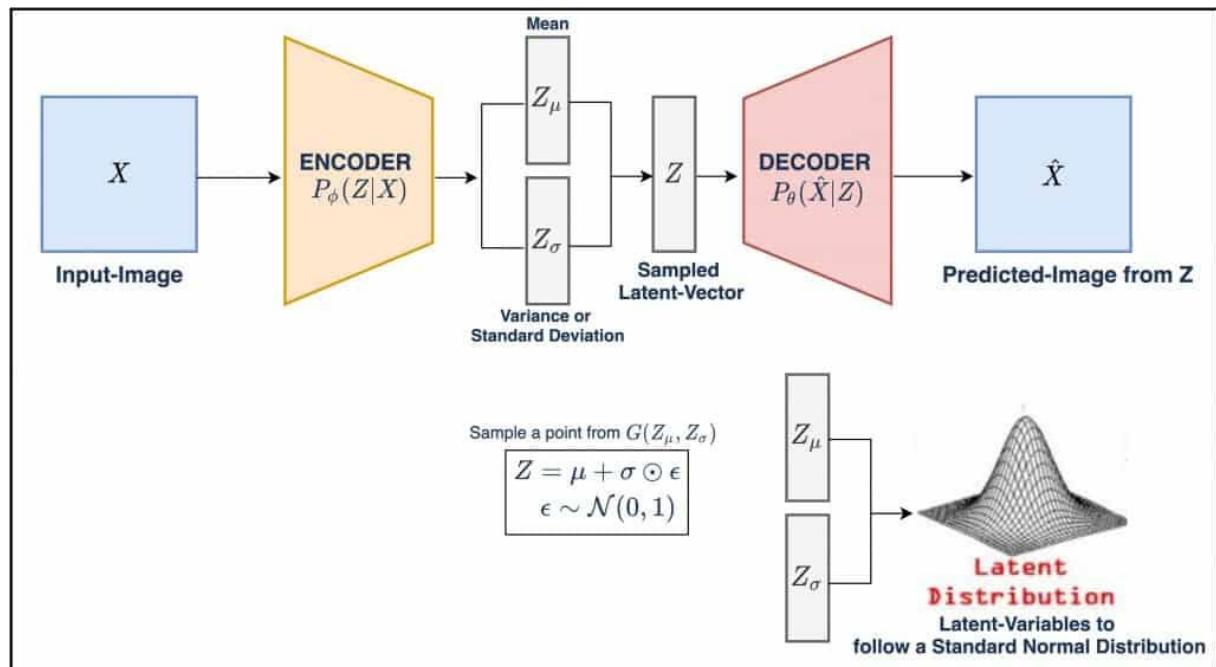


Abbildung 3.1: Modellarchitektur eines Variational Autoencoders (VAE);  
Quelle: (Sharma, 2021)

Adversarial AEs (AAE), arbeiten ähnlich zu Generative Adversarial Networks (GAN). Sie bestehen aus einem Encoder und Decoder, werden aber durch einen Diskriminatator erweitert, der versucht echte und generierte Bilder zu unterscheiden. Dabei kann der Decoder als Generator gesehen werden, der die Eingabedaten rekonstruiert (Berahmand, Daneshfar, Salehi, Li, & Xu, 2024, S. 20-21).

## 3.2 Undercomplete vs. Overcomplete

Undercomplete AEs haben im Latenten Raum eine geringere Dimensionalität als die Eingabedaten, wodurch das Modell gezwungen wird, eine komprimierte Darstellung der Eingabedaten zu lernen. Daher sind sie nützlich, um die wichtigsten Merkmale zu lernen und werden zur Dimensionsreduktion verwendet. Im Gegensatz dazu haben Overcomplete AEs im Latenten Raum eine Dimensionalität die größer oder gleich, wie die der Eingabedaten ist. Ohne weitere Regularisierung würden Overcomplete-AEs jedoch einfach eine Identitätsfunktion lernen, was bedeutet, dass die Eingabedaten ohne nützliche Kompression oder Feature-Extraktion kopiert werden. Aufgrund des Overfitting-Risikos sind sie schwerer zu trainieren als Undercomplete AEs und scheinen nicht so intuitiv, weil keine Komprimierung stattfindet. Sie haben jedoch ihre spezielle Anwendung, wie beispielsweise in der Studie von (Guo, et al., 2021), in der ein Over-and-Under-Complete-Convolutional-Recurrent-Neural-Network (OUCR) entwickelt wurde. Mit dem Ziel bestimmte Bereiche von MR-Bilder genauer rekonstruieren zu können, wurde der Overcomplete-Part verwendet, um spezielle lokale Strukturen der Bilddaten zu lernen und der Undercomplete-Part, um globale Strukturen zu erhalten durch das Lernen von Low-Level-Features.

## 3.3 Deep AE und Stack AE

Da das einfachste AE-Modell aus jeweils einer Encoder- und Decoder-Schicht besteht, würde das keine effektive Merkmalsextraktion bieten. Für eine tiefere Merkmalsextraktion können Stacked- oder Deep-Autoencoder verwendet werden. Deep AEs haben eine tiefe Architektur, wie dargestellt in Abbildung 3.2. Das bedeutet, dass Encoder und Decoder aus mehreren Hidden Layers bestehen, aber beide zusammen einen großen AE darstellen, der die Eingabedaten in mehreren Encoding-Schritten zum Latenten Raum reduziert und in mehreren Decoding-Schritten wieder zur ursprünglichen Dimension führt. Da der AE im Latenten Raum eine effektive Merkmalsrepräsentation der komplexen Daten lernen kann, können sie daher auch als Vorverarbeitung für andere neuronale Netze genutzt werden, indem die Daten erst auf einem Deep AE trainiert werden und dann der Encoder Teil extrahiert wird, um Daten zu kodieren. Die gewonnenen Feature-Vektoren können dann für andere Lernmodelle genutzt werden.

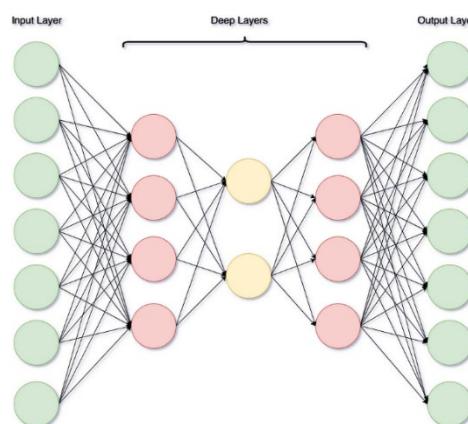
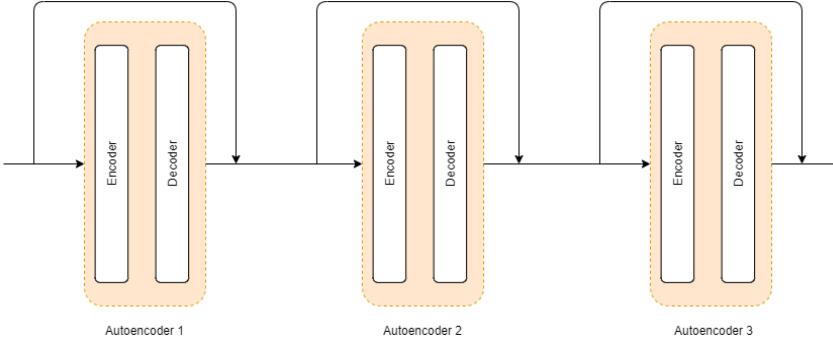


Abbildung 3.2: Darstellung der tiefen Architektur eines Deep AEs mit mehreren Hidden-Layers; Quelle: (Shinde, 2018)

---

Stack AEs, verwenden auch mehrere Encoding- und Decoding-Schritte aber nicht jeweils hintereinander. Sie stapeln mehrere AEs aufeinanderfolgend, wie zu sehen in Abbildung 3.3, wobei die Ausgabe eines Decoders als Eingabe für den nächstfolgenden AE dient. Jeder AE lernt damit eine reduzierte Darstellung der Eingabedaten (Berahmand, Daneshfar, Salehi, Li, & Xu, 2024, S. 12).



*Abbildung 3.3: Architektur eines Stacked AEs bestehend aus mehreren einzelnen AEs;*  
*Quelle: (Rajas, 2021)*

### 3.4 Regularized AE

Regularized AEs extrahieren eine komprimierte Darstellung der Eingabedaten und erzwingen dabei bestimmte Einschränkungen, indem sie die Verlustfunktion erweitern. Sparse AEs (SAE) versuchen spärliche Darstellungen der Eingabedaten zu erzeugen. Das wird erreicht, indem ein Sparsity-Loss zugefügt wird, wie die L1-Regularisierung. Dadurch bleibt nur eine begrenzte Neuronen-Anzahl im Latenten Raum aktiv. Dadurch lernen sie eine spärliche Merkmalsrepräsentation, die die ursprünglichen Daten noch kompakter erfasst. Beispielsweise zeigt die Studie (Cunningham, Ewart, Riggs, Huben, & Sharkey, 2023), dass es möglich ist die Mehrdeutigkeit der Neuronen-Aktivierungen im verwendeten Sprachmodell durch den Einsatz von Sparse AEs deutlich zu verringern. Im Gegensatz dazu verwenden Contractive AEs einen Penalty-Term, der dafür sorgt, dass ähnliche Eingabedaten auch ähnliche Rekonstruktionen erzeugen. Dadurch wird der AE dazu gezwungen robustere Darstellungen zu lernen, die unempfindlich gegenüber Variationen sind und sich nur auf wesentliche Aspekte zu konzentrieren (Berahmand, Daneshfar, Salehi, Li, & Xu, 2024, S. 16). Weil Contractive AEs die robusten Merkmale extrahieren und Variationen unterdrücken, können sie auch für Klassifizierung oder Clustering verwendet werden, wie beispielsweise in der Studie von (Diallo, et al., 2021), die Contractive AEs zum Clustern von Dokumenten angewandt haben. Oder auch in der Anomalie-Erkennung wie in der Studie (Lokman, Othman, Musa, & Abu Bakar, 2019), die einen Deep Contractive AE nutzten, um erfolgreich Anomalien in Fahrzeug-CAN-Daten zu erkennen, indem sie eine robuste Darstellung dieser Daten erlernen. Sparse und Contractive AEs gehören zu den Regularized AEs, da beide die Verlustfunktion um einen weiteren Term erweitern, die den AE zwingen sich an die vorgegebenen Regularisierung Constraints zu halten (Berahmand, Daneshfar, Salehi, Li, & Xu, 2024, S. 17).

### 3.5 Denoising AE

Denoising AEs (DAE) sind darauf ausgelegt mit Rauschen in Daten umzugehen. Im Gegensatz zum klassischen AE, der die Eingabedaten als Zieldaten hat und darauf trainiert ist, sie genau wie möglich zu rekonstruieren, wird der DAE darauf trainiert, Rauschen zu entfernen und Daten ohne Störfaktoren zu rekonstruieren. Dafür werden die Eingabedaten absichtlich mit Rauschen versehen, wobei das Modell die Original-Daten als Zieldaten nimmt. Dadurch lernt der DAE, robuste Merkmale und Mustern, die über verschiedene Rauschmuster beständig sind. (Berahmand, Daneshfar, Salehi, Li, & Xu, 2024, S. 19). Die Qualität kann dabei durch Peak to Signal Noise Ratio (PSNR) (2.4.2) beurteilt werden. DAEs werden daher bei Anwendungen eingesetzt, bei denen Daten durch Rauschen beeinträchtigt sind, wie Sprachverarbeitung und Audiodaten. Beispielsweise wurde in der Studie von (Abouzid, Chakkor, Reyes, & Ventura, 2019) eine Kombination aus Convolutional Denoising AEs verwendet, um eine effektive Methode zur Extraktion gemischter Audio-Datensätzen aus verschiedenen Quellen zu entwickeln.

## 3.6 Zusammenfassung

Wie in Abbildung 3.4 zu sehen, gibt es viele AE-Typen, die sich über die Jahre entwickelt haben, und die in verschiedenen Anwendungen und Bereichen zum Einsatz kommen, wie CAEs im Umgang mit Bildern oder VAEs zur Generierung neue Daten, wobei diese Typen kombiniert wieder neue spezialisiertere AEs bilden, wie Convolutional VAEs (CVAES), Convolutional SAEs (CSAE) oder Deep CAEs (DCAE). AEs eignen sich generell sehr gut zur nichtlinearen Dimensionsreduktion und können flexibel auf verschiedene Datentypen angepasst werden. Denn es gibt noch viele weitere Typen, wie Recurrent AEs zur Verarbeitung von Sequenzen oder Graph AEs, um komplexe Graphen zu vereinfachen (Berahmand, Daneshfar, Salehi, Li, & Xu, 2024, S. 16).

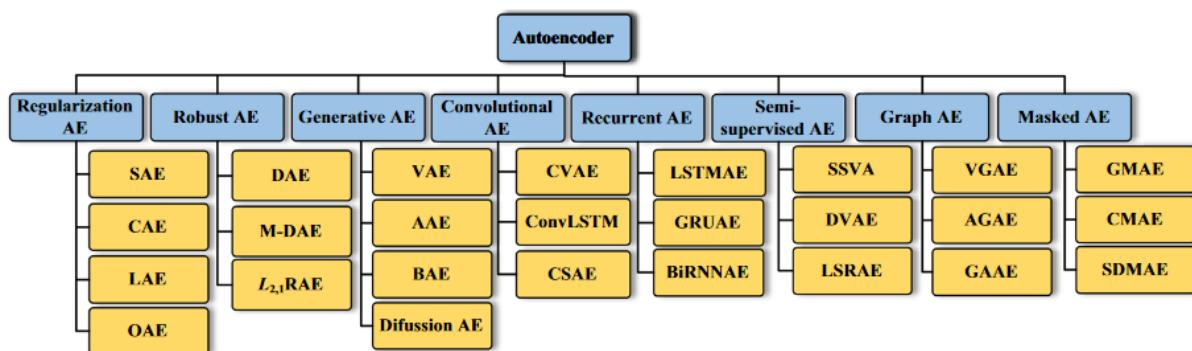


Abbildung 3.4: Übersicht verschiedener AE-Typen kategorisiert nach ihrer Architektur; Quelle: ((Berahmand, Daneshfar, Salehi, Li, & Xu, 2024, S. 16), Fig.4).

---

## **4      Methoden und Analyse**

Im Folgenden werden die Methoden und Verfahren beschrieben, die für die Lösungserstellung in Betracht gezogen wurden und wieso sie aufgrund der Projektanforderungen ausgewählt wurden.

### **4.1      Anforderungsanalyse für Lern- und Auswahlverfahren**

Die Entwicklung eines Systems zur Performanz-Abschätzung, unterliegt bestimmten Anforderungen bezüglich der Projekt-Ziele und der verfügbaren Datensätze, auf denen es angewandt werden soll.

#### **4.1.1    Anforderungen zur Merkmalsextraktion und Dimensionsreduktion**

Um die Datensätze einem Auswahlverfahren zu unterziehen und schließlich ein Ranking zu erhalten, müssen sie in eine vergleichbare Form gebracht und auf eine handhabbare Größe reduziert werden. Das System muss in der Lage sein, die wesentlichen Charakteristiken der Datensätze zu identifizieren, um sie effektiv vergleichen zu können. Wenn alle Merkmale berücksichtigt werden würden, könnte die Menge an unwichtigen Informationen dazu führen, dass sie die tatsächlich relevanten Ähnlichkeiten oder Unterschiede überlagern und damit die Aussagekraft der Vergleiche mindern. Daher ist ein geeignetes Lernverfahren zur Merkmalsextraktion und Dimensionsreduktion erforderlich. Da die vorhandenen Datensätze ungelabelte Bilddaten sowie JSON-Dateien mit Bounding-Box-Informationen für jedes Bild enthalten, wird ein unüberwachtes Lernverfahren benötigt, das mit nichtlinearen Daten umgehen und insbesondere Bildstrukturen erfassen kann.

#### **4.1.2    Anforderungen zum Auswahlverfahren und Bewertungskriterien**

Mit Hinblick auf das nachgeschaltete Auswahlverfahren, ist die Definition der Bewertungskriterien notwendig. Damit sollen Datensätze hinsichtlich ihrer grundsätzlichen Eignung für das Training von ML-Modellen bewertet werden, um für sie ein Ranking zu erstellen. Das Ranking soll dabei helfen, eine Vor-Auswahl von Datensätzen zu treffen, die aufgrund ihrer Eigenschaften als allgemein am besten geeignet erscheinen. Da durch die Vor-Auswahl Trainings, Zeit und Ressourcen gespart werden sollen, sollte auch das Auswahlverfahren nicht auf einer Predictive-Task basieren. Es soll kein Verfahren verwendet werden, das erfordert, die Datensätze einzeln zu trainieren und zu evaluieren, um ihre Eignung anhand einer separaten Performance-Score zu bestimmen. Die allgemeine Annahme „Real-(ähnliche) Daten sind wertvoll“, basiert darauf, dass Realdaten die Unvorhersehbarkeit und hohe Komplexität realer Umgebungen widerspiegeln, die so in synthetischen Daten nicht vorhanden sind. Synthetische Datensätze, die Real-Eigenschaften besser nachahmen und ihnen ähnlicher bzw. näher sind, könnten demnach als wertvoller angesehen werden als rein synthetische ohne Real-Eigenschaften. Eine weitere Anforderung ist deswegen zu erkennen, inwieweit synthetische Daten den echten ähneln. Konkret ist eine Methode notwendig, die die Ähnlichkeit von Feature-Vektoren bewerten kann.

---

## 4.2 Methoden zur Merkmalsextraktion und Dimensionsreduktion

Wie im Kapitel Dimensionsreduktion (2.1) erwähnt, gibt es Techniken wie Kernel-PCA, UMAP oder t-SNE, um hochdimensionale (nichtlineare) Daten im niedrigdimensionalen Raum darzustellen, indem sie versuchen Datenstrukturen wie Varianz (Kernel-PCA), die topologische Struktur (UMAP) oder Nachbarschaftsbeziehungen (t-SNE) zu erhalten. Alle drei nutzen mathematische Transformationen, um Muster und Cluster in den Daten zu identifizieren. UMAP und t-SNE verwenden zwar den Gradient-Descent zur Optimierung der Struktur-Darstellung im niedrigdimensionalen Raum, wie Lernprozesse in Deep Learning Modellen, lernen aber nicht direkt spezifische Bildmerkmale oder Pixelbeziehungen, wie CNNs oder CAEs.

CNNs sind darauf ausgelegt, Merkmale aus Bilddaten zu extrahieren und Muster wie Kanten und Texturen durch Convolutional-Layer zu identifizieren. Da sie jedoch hauptsächlich in überwachten Lernumgebungen eingesetzt werden – in Aufgaben wie Bildklassifizierung, Objekterkennung oder Segmentierung –, erfordern sie gelabelte Daten. Eine weitere Bedingung ist jedoch, dass das Verfahren auch mit ungelabelten Daten umgehen kann, da das System allein durch Analyse der Bildeigenschaften, eine Bewertung erstellen soll.

CAEs sind daher gut geeignet, da sie beide Schritte – die Merkmalsextraktion und Dimensionsreduktion – vereinen, mit Bilddaten umgehen können und keine gelabelten Trainingsdaten benötigen. Nach den im Stand der Technik (3) vorgestellten AE-Typen gibt es viele Möglichkeiten zur Spezialisierung. Da das Ziel ein sehr kompakter Latenter Raum ist, der jedoch die essenziellen Datenmerkmale beibehält, könnte ein Regularized AE (3.4), wie Sparse und Contractive AE, in Frage kommen, da sie eine spärliche Darstellung erzeugen, als CSAE (Convolutional Sparse Autoencoder) oder Domäneneigenschaften hervorheben könnten, als CCAE (Convolutional Contractive Autoencoder). Allerdings könnten Contractive AEs für Eingabedaten von Datensätze gleicher Domäne stark ähnliche Kodierungen erzeugen, wodurch die Datensatz-Kodierungen innerhalb einer Domäne weniger unterscheidbar werden könnten. Sparse Autoencoder könnten das gleiche Problem haben, da sie durch die Regularisierung Neuronen deaktivieren, die wenig zur Repräsentation der Daten beitragen und dadurch eher allgemeinere Merkmale hervorgehoben werden könnten, wenn der Latente Raum sehr klein ist. Klar ist jedoch, dass es sich um einen CAE-basierten AE handeln muss, um mit Bilddaten umzugehen. Ein Deep CAE, wie dargestellt in Kapitel Deep AE und Stack AE (3.3), könnte sich besonders gut eignen, da er eine tiefgehende Merkmalsextraktion durch mehrere Convolutional Layer ermöglicht, um auch komplexe Merkmale zu erfassen. Nach dem Modelltraining könnte der CAE-Encoder-Part entzogen werden und als neues Modell alleinig zur Kodierung von Daten verwendet werden, um Feature-Vektoren der Eingabedaten zu erstellen, die die wichtigsten Merkmale in wenigen Werten darstellen.

---

## 4.3 Evaluierungsmethoden der CAE-Modelle

Zur Bewertung der Modellleistung der CAEs gibt es verschiedene Evaluierungsmethoden, um die Rekonstruktionsqualität und die Effektivität der Kodierungen im Latenten Raum zu beurteilen.

SSI, PSNR, MSE und MAE sind quantitative Evaluierungs-Metriken (2.4), die die Genauigkeit der Bildrekonstruktionen beurteilen. SSI ist eine ganzheitliche Einschätzung der wahrgenommenen Bildqualität, indem die Struktur, Helligkeit und Kontrast zwischen dem Originalbild und der Rekonstruktion verglichen wird. PSNR legt den Schwerpunkt auf die Rekonstruktions-Genauigkeit im Vergleich zum ursprünglichen Signal und zeigt die Störungsfreiheit der Rekonstruktion. Während SSI und PSNR die Erhaltung der Bildstruktur und den Rauschfaktor zeigen, liefern MSE und MAE direkte Messungen des Rekonstruktionsfehlers, wobei MAE alle Fehler gleich und MSE größere Abweichungen stärker gewichtet als kleinere.

Am relevantesten ist hier der SSI, der die strukturelle Ähnlichkeit der Originale und der Rekonstruktionen misst. Datensätze auf denen Scene-Engineering oder Domain-Randomization angewandt wurde, enthalten spezifische Variationen in Eigenschaften wie Lichtverhältnisse, Kamera- oder Objektposition. Diese strukturellen Informationen können durch den SSI erfasst werden, als Maß die Domäneneigenschaften nachzubilden. PSNR wird häufig zur Bewertung der Denoising AE (3.5) eingesetzt, da diese darauf abzielen die Störfaktoren und das Rauschen der Eingabedaten zu entfernen. Die Erhaltung der Bildqualität könnte zudem mit MSE bewertet werden, da dies den Fehler der Rekonstruktion am stärksten darstellt und somit auch kleine Qualitätsunterschiede hervorgehoben werden, im Vergleich zum MAE, der alle Fehler gleichgewichtet. Aus diesem Grund eignen sich SSI und MSE zur Bewertung der Rekonstruktionsqualität und damit indirekt zur Beurteilung der Fähigkeit des CAEs effektive Kodierungen der Eingabedaten zu erstellen.

Der visuelle Vergleich zwischen Originale und Rekonstruktionen ermöglicht eine subjektive, aber leichter interpretierbare Bewertung. Neben der Bildqualität selbst kann erkannt werden, inwiefern das Modell bestimmte Bildinhalte bewahrt oder vernachlässigt. Dadurch lässt sich indirekt erkennen, ob die reduzierte Darstellung des Eingabebildes die relevanten Merkmale beibehält, oder ob die Größe des Latenten Raums zu klein gewählt wurde, wenn bestimmte Eigenschaften der Eingabedaten nicht mehr enthalten sind. Denn wenn die Kodierung gut ist, ist auch eine akzeptable Rekonstruktion möglich. Anzeichen von Overfitting - bei zu starker Dimensionsreduktion – könnten teilweise erkannt werden, falls Domäneneigenschaften in Rekonstruktionen auftreten, die nicht der Domäne entsprechen, der die Eingabedaten zugehören. Wenn ein CAE-Modell in der Lage ist, Daten verschiedener Datensätze und Domänen gut zu rekonstruieren, sollte die Verwendung des Encoder-Teils dieses Modells auch zu einer effektiven Kodierung ganzer Datensätze führen. Dadurch entstehen Feature-Vektoren als komprimierte und repräsentative Darstellungen der Bilder der Datensätze.

---

## 4.4 Analyse des Latenten Raums

Bild-Datensätze haben das Format  $(n, height, width, channels)$  wobei  $n$  die Anzahl der Bilddaten angibt,  $height$  und  $width$  die Bildabmessungen und  $channels$  die Farbkanäle und oder Maskenkanal. Wenn diese vom CAE-Encoder kodiert werden, können sie weiter analysiert werden durch Visualisierung des Latenten Raums einzelner kodierter Bilder oder mehrerer Datenpunkte durch Clustering-Verfahren. Ein kodierter Datensatz wird die Form  $(n, fm_{height}, fm_{width}, dim)$  haben, wobei  $n$  nun der Anzahl der Feature-Vektoren entspricht. Jedes Bild, das durch den Encoder kodiert wurde, erzeugt einen Feature-Vektor, welcher das Eingabebild im Latenten Raum darstellt – reduziert auf den gelernt wichtigsten Merkmalen -. Das kodierte Bild wird durch die Feature-Maps, der Größe  $fm_{height} * fm_{width}$  beschrieben. Die Anzahl der Feature-Maps, wird durch die gewählte Filteranzahl der letzten Encoder-Schicht bestimmt. Dadurch bestimmt  $dim$  - als Filteranzahl der letzten Encoder-Schicht - direkt die Dimensionalität des Latenten Raums.

### 4.4.1 Clustering-Verfahren

Die vollständig kodierten Datensätze sollten die charakteristischen Merkmale der Daten widerspiegeln. Daraus ergibt sich die Erwartung, dass Feature-Vektoren verschiedener Datensätze unterscheidbare Eigenschaften aufweisen, aber durch die zugrunde liegenden Domäneneigenschaften Gemeinsamkeiten haben. Clusterverfahren wie Kernel-PCA, UMAP und t-SNE können Gruppierungen von Vektoren identifizieren, wenn sie sich in Muster und Strukturen ähnlich sind. Wenn Datensätze aus verschiedenen Domänen stammen, könnten die Gruppierungen möglicherweise den Domänen entsprechen oder es können auch Cluster entstehen, die sich auf spezifische Eigenschaften wie Texturen, Farbmuster, Objekte, Lichtverhältnisse oder Kamerastellung beziehen. Nach Kapitel Kernel-PCA, t-SNE und UMAP (2.1.7), geht hervor, dass UMAP sich im Vergleich zu den anderen zwei Methoden am besten eignet, um die lokalen Beziehungen innerhalb der Gruppen darzustellen, aber auch um die globalen Strukturen der Domänen beizubehalten.

### 4.4.2 Heat Maps

Die Visualisierung des Latenten Raums würde direkt zeigen, wie die Kodierungen der komplexen Merkmale aussehen. Die Interpretation der Feature-Maps im Latenten Raum ist jedoch schwierig. In den ersten Schichten können sie einfache Merkmale wie Kanten oder Farben darstellen, die durch Hervorhebung durch Heat Maps visuell nachvollziehbar werden. Sie würden zeigen in welchen Bereichen die Feature-Maps am stärksten oder schwächsten aktiviert sind – als Einblick in den Kodierungsprozess -, um zu sehen, welche Bereiche für die Identifizierung bestimmter Merkmale entscheidend sind, wie der Encoder die Eingabebilder interpretiert und welche Bildaspekte er als die wichtigsten erachtet. In tieferen Schichten mit reduzierter räumlicher Dimension der Feature-Maps

---

werden die Merkmale jedoch abstrakter und komplexer, wodurch die direkte Interpretation der Aktivierungen schwerer wird. Anstatt einzelne Bilder zu betrachten, kann auch ein aggregierter Feature-Vektor visualisiert werden. Als Zusammenfassung des gesamten Datensatzes, könnte dieser Einsicht in die allgemeinen Muster geben. Das ist dann vom Vorteil, um die Menge von Datensätzen in ihrer Gesamtheit zu vergleichen und die Ähnlichkeiten und Unterschiede ihrer typischen Trends zu zeigen.

## 4.5 Aggregationsmethoden

Bei einem Datensatz mit  $n$  Bildern, der durch den CAE-Encoder kodiert wurde, entstehen  $n$  Feature-Vektoren, der Form  $(fm_{height}, fm_{width}, dim)$ , die jeweils ein kodiertes Bild beschreiben. Die  $n$  Kodierungen können durch eine Aggregation weiter reduziert werden. Dabei gibt es verschiedene Aggregationsmethoden, die unterschiedliche Aspekte und Muster des Datensatzes hervorheben. Der **Mean (Mittelwert)** würde die allgemeinste, häufigste Darstellung der Feature-Maps im Latenten Raum zeigen, also ihren typischen Aktivierungs-Zustand. Der **Median** ist wie der Mittelwert, nur robuster gegen Ausreißer und starke Abweichungen durch wenige Feature-Vektoren. Das **Maximum** und **Minimum** würde die stärkste bzw. schwächste Merkmalsaktivierung der Feature-Maps zeigen. Die **Summe** als Gesamtpräsenz, würde die Dominanz eines Merkmals über den gesamten Datensatz zeigen. Der Feature-Vektor der **Varianz** und **Standardabweichung** gibt an, wie sehr die Aktivierung eines Merkmals über die  $n$  Kodierungen variiert. Die **Skewness (Schiefe)** beschreibt die Asymmetrie der Merkmale und zeigt, ob sie gegenüber ihrem Durchschnitts-Wert verzerrt sind. Wenn sie tendenziell stärker ausgeprägt sind, ist diese Schiefe positiv und für eine schwächere Ausprägung negativ. Eine Aggregation kann helfen einen Datensatz-Repräsentant zu bestimmen, der bestimmte Aspekte des Datensatzes hervorhebt, statt  $n$  individuelle Vektoren als Kodierungen zu haben.

## 4.6 Methoden des Auswahlverfahrens

Nach den Anforderungen zum Auswahlverfahren und Bewertungskriterien (4.1.2) sollen Datensätze besser eingestuft werden, die real-ähnlicher sind. Konkret bedeutet das für die Datensatz-Repräsentanten, dass die Datensätze besser eingestuft werden sollten, dessen aggregierter Feature-Vektor ähnlicher zu dem aggregierten Feature-Vektor ist, der den Real-Datensatz repräsentiert. Zum Vektoren-Vergleich gibt es Distanz-Metriken (2.5) wie die Euklidische- oder Manhattan-Distanz. Die beiden können auf hochdimensionalen Vektoren angewandt werden, da sie die Abstände als Punkte im Raum messen, unabhängig von der Dimensionalität. Diese können jedoch in hohen Dimensionen weniger aussagekräftig sein. Nach dem Fluch der Dimensionalität (2.1.2) gleichen sich die Distanzen zwischen den Punkten in hohen Dimensionen an. Wenn die Distanzen beginnen zu konvergieren, wird es schwieriger die Unterschiede zwischen den Datensatz-Repräsentanten zu erkennen. Deshalb könnte die durch den CAE erreichte Dimensionsreduktion nicht ausreichend sein, um die Distanzen zwischen Datenpunkten klar voneinander unterscheidbar zu machen.

---

#### **4.6.1 Voraussetzung zur Anwendung der Distanz-Metriken**

Um die Distanzen auf aggregierte Feature-Vektoren anzuwenden, müssen die Daten in ein Format gebracht werden, das mit den Metriken kompatibel ist. Dazu muss eine Flatten-Operation angewandt werden, die die drei Dimensionen des Vektors ( $fm_{height}, fm_{width}, dim$ ) zusammenlegt und eindimensional macht. Das bedeutet, dass jeder Wert innerhalb der ursprünglichen dreidimensionalen Struktur in eine lineare Sequenz umgeordnet wird mit der Größe ( $fm_{height} * fm_{width} * dim$ ).

#### **4.6.2 Konsequenzen der Flatten-Operation**

Durch die Umwandlung gehen die ursprünglichen Strukturen der Feature-Maps verloren. Unterschiede in bestimmten Feature-Maps können nicht direkt identifiziert werden. Damit bleibt die Interpretation der Distanzmaße sehr allgemein, ohne dass gesagt werden kann, welche spezifischen Feature-Maps oder Pixelpositionen am meisten zu den Unterschieden beigetragen haben. Die Unterschiede oder Ähnlichkeiten werden daher ausschließlich auf Basis der flachen, eindimensionalen Repräsentationen berechnet, als Gesamtähnlichkeit.

#### **4.6.3 Distanzen auf Feature-Vektor-Level**

Die Euklidische-Distanz misst die quadratischen Unterschiede, summiert sie und berechnet die Quadratwurzel der Summe. Größere Unterschiede werden durch das Quadrieren überproportional betont. Eine große Abweichung in einer Feature-Map würde stärker in die Gesamtdistanz einbezogen werden als andere. Die Manhattan-Distanz betrachtet die Summe der absoluten Unterschiede entlang jeder Dimension und zeigt konsistente Unterschiede, als kumulativen Distanzen aller Dimensionen. Das heißt, dass große Distanzen durch stark abweichende Feature-Map genauso viel zur Gesamtdistanz beitragen wie kleine Distanzen ähnlicher Feature-Maps. Beim Vergleich der Feature-Vektoren werden alle Unterschiede über alle Feature-Maps und alle Pixelpositionen hinweg zusammen betrachtet. Das kann nützlich sein, um eine allgemeine Vorstellung von der Ähnlichkeit zweier Datensätze zu erhalten, verdeckt aber spezifische Merkmalsunterschiede einzelner Feature-Maps.

#### **4.6.4 Distanzen auf Feature-Map-Level**

Durch den Vergleich auf Feature-Map-Level kann gezeigt werden, wie ähnlich bestimmte Merkmale der Datensätze zueinander sind. Jedoch müssen auch beim einzelnen Vergleich die Feature-Maps durch eine Flatten-Operation eindimensional gemacht werden. Dadurch würden - wie bei der Anwendung auf Feature-Vektor-Level - nur die Gesamtunterschiede erfasst werden – diesmal pro Feature-Map -. Es ist aber nicht mehr klar, woher die einzelnen Unterschiede der entsprechenden Feature-Map herkommen – also in welchen Bereichen - und wie stark die Abweichungen an dieser Position sind.

---

## 5 Konzept

In diesem Kapitel wird das Konzept vorgestellt, als Plan, wie die im Kapitel Methoden und Analyse (4) erklärten Methoden und Verfahren für die Entwicklung des Performanz-Schätzungssystem angewendet werden sollen. Es wird die Architektur-Planung der CAE-Modelle beschrieben, unter welchen Konfigurationen das Training stattfinden soll, warum sie untersucht werden und wie die zu analysierenden Datensätze schließlich mit dem Encoder-Part des CAE-Modells kodiert werden sollen. Dann wird erklärt, wie das Auswahlverfahren die Ergebnisse der Datensatz-Kodierungen nutzen soll und wie die vom CAE-Encoder gewonnenen Feature-Vektoren weiterverarbeitet werden, damit sie im Auswahlverfahren verglichen werden können, um das Finale Ranking zu erstellen.

### 5.1 Planung der Modellarchitektur

Nach Kapitel Methoden zur Merkmalsextraktion und Dimensionsreduktion (4.2), sind CAEs besonders geeignet, um den Anforderungen zur Merkmalsextraktion und Dimensionsreduktion (4.1.1) gerecht zu werden. Sie vereinen die Merkmalsextraktion und Dimensionsreduktion in einem Modell und arbeiten unüberwacht ohne gelabelte Daten.

Die geplante Modellarchitektur folgt der typischen CAE-Struktur, bestehend aus zwei Submodellen – Encoder und Decoder -. Der Encoder führt Encoding-Schritte durch, um die Größe der Eingabedaten schrittweise zu verringern, bis die Zielgröße des Latenten Raums erreicht ist. Der Decoder wiederum vergrößert die komprimierten Daten des Latenten Raums mittels Decoding-Schritte, bis die ursprüngliche Größe der Eingabedaten wiederhergestellt ist. Sowohl Encoding- und Decoding-Schritte bestehen aus Convolutional-Layer (Conv-Layer), die für das Erfassen der Merkmale zuständig sind. Für beide soll deswegen ein Conv-Block implementiert werden, der als Lernblock, für das reine Feature-Learning verwendet werden kann. Für eine tiefe Merkmalsextraktion, sind mehr Conv-Layer nötig. Daher soll ein Deep-Conv-Block, bestehend aus mehreren einfachen Conv-Blöcken zum Feature-Learning dienen. Die genaue Anzahl ist ein Hyperparameter, der während des Trainings abhängig von den verfügbaren Ressourcen ermittelt werden soll.

In einem Encoding-Schritt folgt nach jedem Lernblock ein Downsampling, was die räumliche Dimension der Feature-Maps reduziert und im Decoding-Schritt ein Upsampling, dass sie wieder erhöht. Das einfachste CAE-Modell besteht daher aus einem Encoding- und Decoding-Schritt. Theoretisch könnten die Eingangsbilder direkt durch einen Encoding-Schritt auf die gewünschte Zielgröße reduziert werden, mit einem großen Stride. Allerdings könnten zu große "Sprünge", zu einem radikalen Informationsverlust führen. Deshalb soll für das Down- und Upsampling ein Stride von 2 verwendet werden. Dadurch wird die Größe der Feature-Maps beim Downsampling im Encoding-Schritt halbiert und beim Upsampling im Decoding-Schritt verdoppelt.

---

Da sich besonders ResNets durch ihre Fähigkeit bewährt haben, das Problem des Vanishing Gradient zu mildern und effektiv in tieferen Architekturen sind, sollen auch Modelle mit Residual-Connections (2.3) erstellt werden können. Daher soll in der Implementierung des Deep-Conv-Blocks ein Eingabeparameter festlegen können, ob dieser zum (Deep-Conv-)Residual Block umfunktioniert werden soll. Das bedeutet, dass eine Skip Connection vom Eingang des Deep-Conv-Blocks bis zum Ausgang gelegt wird. Dieser ResNet-basierte CAE wird im Folgenden als R-CAE bezeichnet.

Die Entscheidung für Strided Convolutions anstelle von Pooling-Funktionen im Encoder basiert darauf, dass Strided Convolutions während des Downampling-Prozesses lernen, wohingegen Pooling-Funktionen eine Subsampling-Operation ohne Lernparameter durchführen. Wie erwähnt im Kapitel Convolutional Autoencoder (CAE) (2.2.3) erzielen die ersten oft bessere Ergebnisse. Darüber hinaus kann der einfache Conv-Block durch die Verwendung von Conv2D für das Downampling recycelt werden. Dadurch kann der Conv-Block sowohl im Deep-Conv-Block als auch im Downampling verwendet werden, indem einfach der Stride angepasst wird. Für das Upsampling im Decoder wird Conv2DTranspose, auch bekannt als Deconvolution, bevorzugt. Obwohl Conv2DTranspose das Risiko von Artefakten in der Rekonstruktion birgt, wie Schachbrettmuster, wird es aufgrund seiner Fähigkeit den Lernprozess zu unterstützen und der Symmetrie willen bevorzugt.

Ein gängiger Ansatz bei CAEs ist es, dass die ReLU-Aktivierungsfunktion in allen Schichten außer der letzten Schicht des Decoders verwendet wird, weil sie nichtlinear arbeitet und dabei hilft, das Problem des Vanishing Gradient zu mindern. In der letzten Schicht des Decoders wird oft Sigmoid verwendet, um die Ausgabe des Netzwerks an den normalisierten Wertebereich [0,1] der Eingabebilder anzupassen. Um die Generalisierungsfähigkeit zu verbessern, soll auch die Batch-Normalization verwendet werden.

Die Zielgröße des Latenten Raums bestimmt, wie stark die Daten komprimiert werden und beeinflusst dadurch die Komplexität des Modells. Je kleiner die Zielgröße, desto mehr Encoding-Schritte sind nötig, um sie zu erreichen. Da im CAE eine Symmetrie zwischen Encoder und Decoder herrschen sollte, werden daher auch mehr Decoding-Schritte benötigt. Die Symmetrie ist nicht unbedingt notwendig, wird standardmäßig jedoch bevorzugt, was vermutlich daran liegt, dass es zum einen sehr intuitiv ist, wenn beide Submodelle aufeinander abgestimmt und zum anderen auch weight-sharing zwischen den Encoder und Decoder angewendet werden kann (was eine Symmetrie voraussetzt), um Rechen- und Speicherressourcen zu reduzieren wie beispielsweise in (Choi, Kim, & Ko, 2021). Auch für die Hyperparametersuche, ist es leichter sie auf einem symmetrischen Modell durchzuführen, da ein Parameter für beide Modelle ausreicht. Deshalb sollen auch die geplanten CAE-Modelle einer gewissen Symmetrie folgen, wobei eine leichte Gewichtung seitens des Encoders auch vom Vorteil sein könnte.

Für die Implementierung des Modells wäre eine symmetrische Struktur auch vom Vorteil. Die Anzahl der Encoding-Schritte beeinflusst letztendlich die Größe des Latenten Raums, und da das Ziel darin besteht, ein Modell zu finden, das die Daten stark komprimieren kann, aber dennoch gute

---

Rekonstruktionen liefert (als Indiz für eine gute Kodierung der wichtigsten Merkmale), wäre es gut, die Modelltiefe leicht anpassen zu können. Geplant ist daher, das Modell mit nur einem Parameter zu erstellen - einer Liste von Filtern. Jeder Wert in der Liste würde die Anzahl der Filter bestimmen, die in einem bestimmten Encoding-Schritt verwendet werden. Die Anzahl der Listenelemente bestimmt wiederum die Anzahl der Encoding-Schritte und somit auch, wie stark das Modell die Eingabedaten reduziert bzw. (bei einem Stride von 2) wie oft sie halbiert werden, bevor der Latente Raum erreicht wird, ausgehend von der Größe der Eingabebilder. Zusammengefasst sollen einzelne Bausteine implementiert werden, mit denen dynamisch verschiedene CAE-Modelle erstellt werden können:

- Deep-Conv-Block zum reinen Feature-Learning in jedem Encoding- /Decoding-Schritt
- Einfacher Conv-Block als Baustein im Deep-Conv-Block und im Downsampling mit Stride 2
- Einfacher De-Conv-Block als Baustein im Upsampling mit Stride 2
- Residual Block als Variante des Deep-Conv-Blocks mit Skip Connections
- Filter-Liste als Eingabeparameter zur dynamischen Erstellung der Zielgröße

## 5.2 Geplante Trainings-Konfigurationen

Durch die geplanten Bausteine können verschiedene Modellarchitekturen erstellt werden. Dabei sind die Anzahl der Encoding-Schritte, die durch die Filterliste angegeben wird, sowie die Anzahl der Conv-Blöcke im Deep-Conv-Block oder im Residual Block Hyperparameter. Es sollen verschiedene Trainingskonfigurationen getestet werden, um das optimale CAE- oder R-CAE-Modell zu ermitteln, das die Daten stark in ihrer Dimension reduzieren und dennoch gute Rekonstruktionen erzeugen kann. Wenn die Rekonstruktionen gut sind, ist das ein Indiz dafür, dass das Modell die Charakteristika der Daten im Latenten Raum einfängt und wichtige Eigenschaften beibehält. Um das zu prüfen, sollen die Trainings-Ergebnisse mit den Evaluierungsmethoden der CAE-Modelle (4.3) bewertet werden.

Es sollen verschiedene Konfigurationen für die Trainings getestet werden: Zunächst soll die Anzahl der möglichen Conv-Blöcke im Deep-Conv-Block ermittelt werden ausgehend der verfügbaren Ressourcen. Anschließend sollen verschiedene Filterlisten untersucht werden. Die Filterlisten variieren in ihrer Länge und jedes Element der Filterliste gibt die Filteranzahl für den entsprechenden Encoding-Schritt an. Das Ziel ist es, die ideale Zielgröße des Latenten Raums zu finden. Nachdem die optimale Zielgröße ermittelt wurde, soll auch der Einfluss der Residual-Connections untersucht werden, inwiefern sie die Modellleistung verbessern. Das heißt, CAE- und R-CAE-Modelle werden mit der gleichen Zielgröße direkt verglichen. Auch sollen Modelle mit weniger Real-Trainingsdaten getestet werden.

Danach soll noch die Filter-Strategie betrachtet werden, indem Modelle mit einer kontinuierlich wachsenden oder abnehmenden Filteranzahl verglichen werden. Das Modell mit einer niedrigeren Anzahl an Filtern erhöht diese sukzessive in den tieferen Schichten. Die Strategie konzentriert sich darauf, zunächst grundlegende und dann zunehmend komplexere und spezifischere Merkmale aus den

---

Eingabedaten zu extrahieren. Das Modell, das mit einer hohen Filteranzahl beginnt, reduziert sie nach und nach in den tieferen Schichten. Die anfänglich hohe Filterzahl sollte eine breite Palette von Merkmalen frühzeitig erfassen, wobei die anschließende Reduktion der Filteranzahl das Netzwerk dazu zwingt, die erfassten Informationen auf die wesentlichsten Merkmale zu reduzieren.

Neben den Modellspezifischen Konfigurationen, sollen auch die Datensätze mit zwei Ansätzen trainiert werden. Da Datensätze existieren, die aus bestimmten Domänen stammen, soll untersucht werden, ob das CAE-Modell die Charakteristiken der Domänen besser im sequenziellen oder kombinierten Training lernen kann.

### 5.3 Weiterverarbeitung der Ergebnisse

Nach der Bestimmung des optimalen Modells, soll der Encoder-Part des CAEs verwendet werden, um einen ganzen Datensatz mit  $n$  Bildern zu kodieren, wodurch  $n$  Feature-Vektoren entstehen. Obwohl die Originaldaten reduziert werden, stellen  $n$  Feature-Vektoren noch immer eine hohe zu vergleichende Menge dar. Aus dem Grund müsste eine weitere Reduktion stattfinden mithilfe der Methoden beschrieben in Aggregationsmethoden (4.5).

Da jedoch nicht die Pixelpositionen der Feature-Maps zusammengelegt werden sollten, um die Rauminformation zu wahren und auch nicht alle Feature-Maps, um die individuellen Merkmale zu erhalten, könnten nur die  $n$  Datenpunkte zusammengefasst werden. Die Aggregation über  $n$  Feature-Vektoren, würde in einen einzigen resultieren, der aus allen individuellen Bildkodierungen im Latenten Raum eine Gesamtrepräsentation des Datensatzes erzeugt. Wenn Repräsentanten für alle verfügbaren Datensätze bestimmt wurden, können die aggregierten Vektoren im Auswahlverfahren weiter verglichen werden. Da es verschiedene Methoden gibt, die verschiedene Aspekte der Daten betonen, ist es ein Ziel, eine Aggregationsmethode zu finden, die die Daten so zusammenfasst, dass die zugrunde liegenden Muster und Unterschiede zwischen den einzelnen Datensätzen und ihrer Domänen beim Vergleich am deutlichsten hervorgehoben werden. Es wird angenommen, dass „Mean“ bzw. „Median“ besonders aussagekräftig sein wird, da sie den typischen Zustand der Feature-Maps zeigen.

### 5.4 Planung des Rankings durch Auswahlverfahren

Wie im Kapitel Methoden des Auswahlverfahrens (4.6) erklärt wird, ist für das Auswahlverfahren keine Predictive Task geeignet, die für jeden Datensatz einzeln eine direkte Performance-Score ausgibt, wie etwa die Genauigkeit eines Objekterkennungsmodells. Ein solcher Ansatz würde ein überwachtes Lernmodell erfordern, das für jeden Datensatz separat trainiert und ausgewertet werden müsste. Da ein Ziel des Projekts jedoch darin besteht, den Trainingsaufwand durch eine Vor-Auswahl zu minimieren, wird ein anderer Ansatz verfolgt: Die Performanz soll durch die Analyse der Ähnlichkeiten der Datensatz-Repräsentanten (gewonnen aus der Aggregation) abgeschätzt werden – also wie nahe oder

---

ähnlich sich die kodierten Darstellungen der Datensätze im Merkmalsraum sind. Dabei sollen Datensätze, die dem Real-Datensatz nahekommen, höher bewertet werden.

Die Annahme - Je mehr Real-Eigenschaften die Datensätze haben, desto wertvoller sind sie - stützt sich zudem auf vorhandene Performance-Scores von vier Datensätzen einer Objekterkennungsaufgabe, die als Referenz dienen. Diese Datensätze – `adidas_real`, `adidas_syn_se`, `adidas_syn_simple` und `adidas_syn_dr` – repräsentieren die Domänen Real, Scene Engineering (SE), Simple und Domain Randomization (DR). Die Reihenfolge ihrer Performance-Scores, von der höchsten zur niedrigsten, zeigt, dass die Objekterkennungsaufgabe auf dem Real-Datensatz am effizientesten durchgeführt wurde, gefolgt von den synthetischen Datensätzen. Im weiteren Verlauf werden die genannten vier Datensätze als „Basis-Datensätze“ bezeichnet. Die restlichen verfügbaren Datensätze sind Varianten der synthetischen Basis-Datensätze und gehören zu einen der drei Domänen – SE – Simple – DR – an, wobei der Real-Datensatz der einzige der Real-Domäne ist. Zum Beispiel ist der Datensatz „`adidas_syn_simple_real_texture`“ eine Variante des Simple-Basis-Datensatzes „`adidas_syn_simple`“. Die Variante mit realen Texturen ist somit Teil der Simple-Domäne.

Mit den Basis-Datensätzen als Referenz-Punkte, sollen alle Varianten-Datensätze durch das Auswahlverfahren bewertet und in eine Rangfolge gebracht werden. Ausgehend davon soll das Auswahlverfahren, die Zugehörigkeit zu ihren jeweiligen Domänen identifizieren können. Dadurch kann ein vorläufiges Ranking erstellt werden, indem die Datensätze erst entsprechend ihrer Domänen gruppiert werden. Diese Zuordnung soll als grobe Vorhersage der zu erwartenden Leistung dienen. Ein synthetischer Datensatz, der einen der drei synthetischen Domänen – SE, Simple, DR - eindeutig zugeordnet werden kann, wird voraussichtlich eine ähnliche Leistung wie der Repräsentant dieser Domäne aufweisen. Wenn Datensatz „`adidas_syn_simple_real_texture`“ als am ähnlichsten zum Simple-Basis-Datensatz erkannt wird, dann wird auch eine ähnliche Performanz erwartet.

Die Fähigkeit des Auswahlverfahrens, Datensätze anhand ihrer Eigenschaften richtig zuzuordnen, hängt natürlich stark davon ab, ob die aggregierten Feature-Vektoren überhaupt unterscheidbare Merkmale enthalten. Da alle Datensätze aus einer der vier Domänen stammen, ist anzunehmen, dass die charakteristischen Merkmale jeder Domäne auch in gewissem Maße in den entsprechenden Datensätzen vorhanden sind und sich in den (aggregierten) Feature-Vektoren widerspiegeln. Die Qualität und Aussagekraft dieser Feature-Vektoren hängt daher direkt von der Leistung des trainierten CAE-Modells ab, das zur Kodierung aller Datensätze verwendet wurde und von der Aggregation zum Datensatz-Repräsentant.

#### **5.4.1 Distanzbasierter Ansatz**

Wie im Kapitel Methoden des Auswahlverfahrens (4.6) erwähnt, gibt es Distanz-Metriken, um die Ähnlichkeit von Vektoren zu messen, wie die Euklidische- und Manhattan Distanz. Es ist nicht klar, ob

---

eine direkte Korrelation zwischen der Nähe der Datensätze zur realen Domäne – gemessen an den Distanzen – und den Performance-Scores der Objekterkennungsaufgabe besteht. Das zu ermitteln, ist ein Ziel der Untersuchung. Da die Manhattan-Distanz besser im Umgang hochdimensionaler Daten geeignet ist, wird sie der euklidischen Distanz vorgezogen. Die Anwendung beider, soll jedoch für das distanzbasierte Auswahlverfahren möglich sein.

An dieser Stelle ist anzumerken, dass das auf Domänen und Distanzen basierende Vorgehen, seine Grenzen hat. Die hier bereits erwähnten Domänen und spezifischen Datensätze dienen nur zum Verständnis. In der Anwendung auf andere Datensätze sind diese womöglich nicht eindeutig zu Domänen zuordbar oder es besteht kein Wissen über die Zugehörigkeiten oder Existenz von Domänen. Zudem könnten innerhalb der Domänen die Unterschiede zwischen den Datensätzen so groß sein, dass allein auf Grundlage der Distanz keine eindeutige Zuordnung stattfinden kann. Selbst wenn ein Datensatz offensichtlich nach visueller Betrachtung oder durch das vorhandene Domänen-Wissen eine Variante des domänenrepräsentierenden Basis-Datensatzes ist. Beispielsweise ist durch Domänen-Wissen bekannt, dass "adidas\_syn\_se\_random\_light" eine Variante des SE-Basis-Datensatz ist. Es ist jedoch unklar, ob das Auswahlverfahren genug SE-Eigenschaften in seinem aggregierten Feature-Vektor – also dem Datensatz-Repräsentant - erkennt, um ihn richtig zuzuordnen. Deswegen wird neben dem Ziel ein Ranking zu erstellen, parallel auch das Ziel verfolgt, generell Ähnlichkeiten zwischen Datensätzen zu erkennen. Denn es könnten weitere Gruppen existieren, die die Domänen-Grenzen überschreiten, oder Subgruppen innerhalb der Domänen, die sich durch spezifische Merkmale unterscheiden. Zu dem Zweck, sollen nicht nur die Distanzen zwischen allen Datensatz-Repräsentanten zu den Basis-Datensatz-Repräsentanten gemessen und analysiert werden, sondern auch alle Distanzen der Datensatz-Repräsentanten untereinander, um natürliche Gruppen bilden zu lassen. Im Falle das kein Domänen-Wissen besteht, könnte das System erst dazu verwendet werden so Gruppierungen zu identifizieren und dann einen Repräsentanten für die Gruppe zu bestimmen.

Da bekannt ist, dass es Varianten-Datensätze gibt, die sich in einem bestimmten Aspekt unterscheiden, wie Texturen, die *real* oder *random* sein können, könnten diese als Subgruppen innerhalb der Domänen oder sogar über Domänen-Grenzen hinweg identifiziert werden. Zum Beispiel zeigen die Varianten "adidas\_syn\_simple\_real\_texture" und "adidas\_syn\_dr\_real\_texture" beide reale Texturen. Es ist möglich, dass diese Texturvariation als separate Gruppe über die Domänen-Grenzen hinweg erkannt wird. Jedoch ist es wahrscheinlicher, dass Subgruppen innerhalb derselben Domäne erfasst werden, wie "adidas\_syn\_simple\_real\_texture" und "adidas\_syn\_simple\_random\_texture", falls die Texturinformation aus denselben Feature-Maps oder einer Kombination weniger stammt. Denn die Domänen-Eigenschaft könnte einen zu großen Einfluss auf die Merkmale haben, als dass die Gruppe über Domänen-Grenzen hinweg bestehen bleibt. Genau wie eine oder wenige Feature-Maps Informationen zur Textur enthalten können, könnten wenige oder mehrere Feature-Maps spezifische Domänen-Eigenschaften beinhalten. Das bedeutet, selbst wenn die Texturinformation von

---

„adidas\_syn\_simple\_real\_texture“ und „adidas\_syn\_dr\_real\_texture“ nur in einer Feature-Map an bestimmten Pixelpositionen dargestellt wird und diese beim direkten Vergleich sehr ähnlich sind, könnten sie sich in anderen Bereichen derselben Feature-Map zu stark unterscheiden. Dadurch könnte der Gesamtabstand zwischen beiden Feature-Maps größer werden.

Doch für die Erstellung des Rankings sollen die Basis-Datensätze als Referenzpunkte verwendet werden. Obwohl die Performance-Scores der Objekterkennungsaufgabe darauf hindeuten, dass Datensätze bestimmter Domänen (z.B. Simple) generell besser abschneiden als solche aus anderen Domänen (z.B. DR), könnten die tatsächlichen Distanzen der synthetischen Basis-Datensätze zum Real-Datensatz das nicht widerspiegeln. Das heißt, es ist nicht auszuschließen, dass sich die Domänen überlappen. Deshalb kann nicht grundsätzlich davon ausgegangen werden, dass das distanzbasierte Bewertungssystem ein Datensatz aus der Simple-Domäne in jedem Fall besser bewertet als ein Datensatz aus der DR-Domäne. Zum Beispiel könnte es einen DR-Datensatz geben, mit vielen realen Eigenschaften. Gleichzeitig könnte auch ein Simple-Datensatz existieren, der gar keine realen Merkmale enthält und vielleicht sogar andere Eigenschaften, die ihn noch weiter vom Real-Datensatz entfernen. Dadurch könnte der DR-Datensatz ähnlicher zum Real-Datensatz sein als der Simple-Datensatz, obwohl die Performance-Scores der Objekterkennungsaufgabe zeigen, dass Simple-Datensätze im Allgemeinen bessere Leistungen erzielen als DR-Datensätze.

Um mögliche Überlappungen zwischen den Domänen zu verhindern, könnte eine feste Reihenfolge der Domänen im Ranking eingeführt werden, z.B. dass im Ranking zuerst alle Datensätze der Real-Domäne, dann der SE-Domäne, anschließend der Simple-Domäne und zuletzt der DR-Domäne aufgeführt werden. Der Ansatz orientiert sich an den Ergebnissen der Objekterkennungsaufgabe. Innerhalb der Domäne sollen die Datensätze dann entsprechend ihrer (Real-)Eigenschaften neu geordnet werden. Ein klarer Vorteil der Vermeidung der Domänen-Überlappung ist, dass die Vergleichbarkeit dadurch einfacher wird und durch die Neuordnung innerhalb der Domäne, die Performanz genauer beurteilt werden kann als alleinig auf Basis der Domäne.

Ein bedeutender Nachteil ist jedoch, dass das Ranking dadurch unflexibel wird. Durch die erzwungene Domänen-Reihenfolge besteht die Gefahr, dass Domänen bevorzugt behandelt werden und dadurch eine Verzerrung der Ergebnisse entstehen kann. Beispielsweise, wenn „adidas\_syn\_dr\_real\_texture“ trotz geringerer Distanz zum Real-Datensatz (durch seine Real-Eigenschaften) aufgrund seiner Zugehörigkeit zur DR-Domäne nie besser bewertet werden kann als ein Simple-Datensatz. Demnach wäre die Einordnung nicht fair, da der Datensatz durch seine Domänen-Zugehörigkeit benachteiligt wird, obwohl er in anderen Aspekten, besser abschneidet. Ob eine feste Domänen-Reihenfolge noch ein aussagekräftiges Ranking darstellt, wird im Kapitel Grenzen des Auswahlverfahrens (8.2) erläutert.

Das heißt: Im ersten Schritt des Auswahlverfahrens, sollen die Distanzen der synthetischen Basis-Datensatz-Repräsentanten zum Real-Datensatz gemessen werden, um die Domänen-Reihenfolge zu

---

ermitteln. Im zweiten Schritt sollen die Distanzen zwischen allen Datensatz-Repräsentanten zu den Basis-Datensatz-Repräsentanten gemessen werden, und dies für alle verfügbaren Datensätze, um sie ihrer entsprechenden Domäne zuzuordnen. Danach soll eine dritte Bewertung stattfinden, die die Datensätze nach ihrer Ähnlichkeit zum Real-Datensatz innerhalb der Domäne neu einordnet. Beispielsweise ist nach Domänen-Wissen bekannt, dass die Datensätze „adidas\_syn\_dr“ und „adidas\_syn\_dr\_real\_texture“ zur selben Domäne angehören.

Wird dies erkannt und ist auch anhand der Real-Nähe zu sehen, dass „adidas\_syn\_dr\_real\_texture“ durch die „realen Texturen“ eine niedrigere Distanz zum Real-Datensatz hat, so soll dieser Varianten-Datensatz besser eingestuft werden innerhalb der DR-Domäne als der DR-Basis-Datensatz, der die Domäne repräsentiert. Dazu können die Datensatz-Repräsentanten auch auf Feature-Map-Level untersucht werden, indem beispielsweise die Distanz zwischen Feature-Map  $k$  von Datensatz „adidas\_syn\_simple\_real\_texture“ und der Feature-Map  $k$  aller Basis-Datensätze gemessen wird. Wird dies für alle Feature-Maps gemacht, kann analysiert werden, welche Features zu welchen Basis-Datensätzen am ähnlichsten sind und dadurch auch, welche für das Lernen von Real-Eigenschaften zuständig sind.

Es ist zu erwarten, dass die meisten Feature-Maps am ähnlichsten zu dem Basis-Datensatz sind, dessen Domäne sie angehören. Jedoch könnten bestimmte dafür zuständig sein, eine reale Eigenschaft zu lernen, wobei dann für solche Feature-Maps eine höhere Ähnlichkeit zum Real-Datensatz erwartet wird. Zum Beispiel wird angenommen, dass der Datensatz „adidas\_syn\_simple\_real\_texture“, aufgrund der realen Texturen, besser bewertet wird als der Simple-Basis-Datensatz „adidas\_syn\_simple“, der die Simple-Domäne repräsentiert. Denn man würde erwarten, dass es Feature-Maps gibt, die die realen Texturen erfassen und damit die Distanz zum Real-Datensatz verringern. Der Einfluss einzelner Feature-Maps würde beim direkten Vergleich auf Feature-Vektor-Level nicht erkennbar sein.

Zusammenfassend sollen mit dem Auswahlverfahren erst die großen Unterschiede zwischen den Domänen erfasst werden durch Vergleich auf Feature-Vektor-Level und anschließend auf Feature-Map-Level spezifische Merkmale der Datensätze analysieren, die in einer allgemeineren Bewertung möglicherweise unterdrückt wurden. Das entstehende Ranking wird ordinal sein, da keine Predictive Task durchgeführt wird, um eine Performance-Score für jeden Datensatz zu generieren. Stattdessen dienen die vorhandenen Performance-Scores der Objekterkennungsaufgabe als Referenzwerte. Durch das Vorwissen der Domänen-Zugehörigkeiten kann überprüft werden, ob das Auswahlverfahren die Datensätze auf Grundlage der Distanzen korrekt zuordnen kann. Zusätzlich gibt es auch Performance-Scores zu einer kleinen Auswahl der Varianten-Datensätze, um zu überprüfen, ob das entstehende ordinale Ranking die Ergebnisse der Objekterkennungsaufgabe widerspiegelt.

---

#### **5.4.2 Clusteranalyse und SVM als Alternativen**

Neben dem distanzbasierten Ansatz soll parallel auch eine Clusteranalyse der vom CAE-Encoder gewonnenen und aggregierten Feature-Vektoren durchgeführt werden. Dabei wird untersucht, ob Verfahren wie UMAP in der Lage sind, die Feature-Vektoren aller Datensätze entsprechend zu gruppieren und ob diese Gruppierungen den Domänen entsprechen. Das dient zum einen als Alternative, falls die Distanzen allein nicht aussagekräftig genug sind, um die Zuordnung zu den Domänen vorzunehmen. Zum anderen ermöglicht es eine zusätzliche Überprüfung, ob die Ergebnisse der Clusteranalyse mit den berechneten Distanzen übereinstimmen.

Eine weitere Möglichkeit ist die Anwendung einer Support Vector Machine (SVM). Nachdem die Basis-Datensätze vom CAE-Encoder kodiert wurden, soll der SVM auf den gewonnenen Feature-Vektoren der Basis-Datensätze trainiert werden. Diese sollen entsprechend ihrem Basis-Datensatz gelabelt werden, um ihre Domäne zu repräsentieren. Nachdem die SVM trainiert wurde, soll versucht werden, die Domänen-Zugehörigkeit der Varianten-Datensätze vorherzusagen. Zum Beispiel sollte für die Feature-Vektoren der Simple-Variante „adidas\_syn\_simple\_real\_texture“, vorhergesagt werden, dass dieser Datensatz zur Domäne „Simple“ gehört. Das würde zwar ein weiteres Training bedeuten, jedoch ist das SVM-Training auf Basis-Datensätzen immer noch ressourcenschonender als ein separates Modell für jeden einzelnen Datensatz zu trainieren.

Zusammengefasst, soll versucht werden, die Ähnlichkeit der vom CAE gewonnenen Feature-Vektoren mit den drei Ansätzen – Distanzen, Clusteranalyse, SVM - zu erfassen, um die Domänen-Zuordnung zu bekommen und demnach eine grobe Schätzung der zu erwartende Performanz. Ob diese Ansätze eine effektive Abschätzung der Performanz und ein aussagekräftiges Ranking ermöglichen, soll in der weiteren Implementierung und Evaluierung geprüft werden.



---

## **6      Implementierung**

In diesem Kapitel wird die praktische Umsetzung beschrieben, von der Vorverarbeitung der Daten, zur Implementierung und Anwendung des Modells und des Auswahlverfahrens.

### **6.1      Datenvorbereitung**

Die Trainingsdaten und die Datensätze, die mit dem Encoder-Modell kodiert werden sollen, müssen vorverarbeitet werden.

#### **6.1.1      Verfügbare Datensätze**

Es stehen insgesamt 17 Datensätze mit Farbbildern der ursprünglichen Dimension  $512 \times 682$  zur Verfügung, wobei alle synthetischen Datensätze 14.000 PNG-Bilder enthalten. Der Real-Datensatz „adidas\_real“, enthält 5.452 PNG-Bilder. Für jedes Bild gibt es eine zugehörige JSON-Datei, mit Bounding-Box-Informationen.

Die Bilder im Real-Datensatz zeigen eine große Box mit einem oder mehreren Adidas-Schuhkartons in verschiedenen Positionen. Die Kamera ist dabei von oben auf die offene Box gerichtet und zentriert sie im Bild. Diese Grundszenen werden auch in den synthetischen Datensätzen nachgebildet. Für den SE-Basis-Datensatz "adidas\_syn\_se" wurde Scene Engineering angewandt, um eine realistische Darstellung von Texturen, Beleuchtung und Positionen der Boxen zu erzeugen, die eine möglichst authentische Simulation der realen Szenen darstellen. Der Simple-Basis-Datensatz "adidas\_syn\_simple" hingegen zeigt eine minimalistische und abstrakte Darstellung der Grundszenen. Hier werden einfache Rechtecke und einheitliche Farben verwendet, um die Szene zu repräsentieren. Im „adidas\_syn\_dr“ wurde hingegen Domain Randomization angewandt. Die Bilder dieses Datensatzes zeigen Variationen der Realdaten durch unvorhersehbare Szenarien, vielfältige Umgebungen und zufällige Veränderungen in Bezug auf Hintergründe, Texturen, Beleuchtung und Positionen der Objekte.

Wie bereits im Kapitel Planung des Rankings durch Auswahlverfahren (5.4) erwähnt, werden diese drei synthetischen Datensätze [adidas\_syn\_se, adidas\_syn\_simple, adidas\_syn\_dr] als Basis-Datensätze bezeichnet, weil die restlichen synthetischen Datensätze auf ihnen basieren und Variationen davon darstellen. Die Varianten unterscheiden sich beispielsweise in Lichtverhältnissen, Kameraposition, Hintergrund oder Texturen. Aus diesem Grund werden diese synthetischen Basis-Datensätze als Repräsentanten der Domänen – SE – Simple – DR - gewählt. Der Real-Datensatz ist dabei der Repräsentant der Real-Domäne. Für das CAE-Modell, das die Eigenschaften aller Datensätze möglichst gut in den Kodierungen erfassen soll, werden deshalb Trainingsdaten aus den Basis-Datensätzen verwendet. Als Domänen-Repräsentanten decken diese vier Basis-Datensätze die Vielfalt aller Datensätze ab, die auf ihnen basieren.

---

## 6.1.2 Padding

Standardmäßig nehmen CAEs Bilder der Form (*height, width, channels*) entgegen. Die verfügbaren Farbbilder haben eine Originalgröße von  $512 \times 682$  Pixeln. Da große Bilder als Eingabedaten die Anzahl der zu verarbeitenden Parametern erhöhen, was zu einer größeren Rechenlast und längeren Trainingszeiten führt, werden die Bilder zuerst verkleinert. Durch Halbierung der Höhe und Breite auf  $256 \times 341$  Pixel wird das Bildverhältnis beibehalten, um Verzerrungen zu vermeiden. Allerdings wird, um die Verarbeitung im CAE zu optimieren, ein explizites Padding hinzugefügt, was die Breite mit Pixeln konstanter Farbe auf  $256 \times 384$  Pixel erweitert. Diese Pixel entsprechen dem Grauwert der Ränder des Bildes. Durch die Anpassung kann die Bildabmessung fortlaufend halbiert werden ohne das ungerade Werte beim DownSampling auftreten:

$$256 \times 384 - 128 \times 192 - 64 \times 96 - 32 \times 48 - 16 \times 24 - 8 \times 12 - 4 \times 6 - 2 \times 3$$

Dadurch kann die räumliche Dimension maximal zu  $2 \times 3$  reduziert werden. Bei einer ungeraden Größe von  $256 \times 341$  würde das Ergebnis auf die nächsthöhere ganze Zahl gerundet werden. Schon nach der ersten Halbierung der Breite 341 zu 170.5 wird die Breite auf 171 erhöht, was erneut eine ungerade Breite erzeugt, die bei der nächsten Halbierung wieder eine Rundung benötigt. Durch das explizite Padding für eine Größe von  $256 \times 384$  wird die Verarbeitung im Netz vereinfacht, weil dadurch sichergestellt wird, dass die räumliche Dimension der Feature-Maps bei jedem DownSampling exakt halbiert wird, ohne Rundungsprobleme zu verursachen. Generell sind diese Rundungsprobleme nicht schwerwiegend, aber im Falle vom CAE-Modell kann es zu Komplikationen führen, da die Architektur des Encoders und des Decoders aufeinander abgestimmt sein muss, um mit dem Decoder die Eingabegröße zu erzeugen. Wenn die Größe der Feature-Maps beim DownSampling ungerade wird und dann beim Upsampling versucht wird die ursprüngliche Größe wiederherzustellen, würde das Runden auf die nächsthöhere Zahl bei ungeraden Dimensionen im Encoder eine falsche Basis für den Decoder liefern. Damit ist gemeint, dass bei der Verwendung von  $256 \times 341$  und 5 DownSamplings folgende Abmessungen nach den Halbierungen entstehen:

$$256 \times 341 - 128 \times 171 - 64 \times 86 - 32 \times 43 - 16 \times 22 - 8 \times 11$$

Die Feature-Maps im Latenten Raum würden somit eine Abmessung von  $8 \times 11$  haben, was der Decoder als Basis nimmt. Da der Symmetrie Willen, bei 5 DownSamplings im Encoder auch 5 Upsamplings im Decoder erfolgen, würden folgende Abmessungen nach den Verdopplungen entstehen:

$$8 \times 11 - 16 \times 22 - 32 \times 44 - 64 \times 88 - 128 \times 176 - 256 \times 352$$

Damit hätten die Ausgabedaten mit  $256 \times 352$  nicht dieselbe Form wie die Eingabedaten der Form  $256 \times 341$ . Nachdem die Bildgröße für alle Bilder aller Datensätze verkleinert und das explizite Padding hinzugefügt wurde, wurden alle Datensätze von  $(n, 512, 682, 3)$  zu  $(n, 256, 384, 3)$  umgewandelt.

---

### **6.1.3 Masken**

Es gibt für jedes PNG-Bild eine zugehörige JSON-Datei, die Informationen zu den Bounding-Boxen enthält. Es ist jedoch problematisch, mehrere Masken für ein Bild zu verwenden, weil die Eingabegrößen für das Netz konsistent sein müssen. Aus diesem Grund wird keine herkömmliche Einzelmaskenstruktur erstellt, die für jedes einzelne Objekt im Bild eine Maske erstellt. Stattdessen werden alle Bounding-Box-Informationen verwendet, um eine einzige Maske zu erstellen, die alle Objekte im Bild mit weißen Pixeln markiert und den Rest des Hintergrunds mit schwarzen Pixeln füllt.

Im Training könnten die Masken dem CAE helfen, die relevantesten Bereiche des Bildes zu identifizieren, indem das Modell lernt, sich auf die von den Masken hervorgehobenen Bildbereiche zu konzentrieren und den Hintergrund als nicht so wichtig zu erachten. Dadurch könnte das Modell effektiver Merkmale der Objekte extrahieren.

Da sich die JSON-Files auf die Original-Abmessungen von  $512 \times 682$  beziehen und das CAE-Modell Daten mit Abmessung  $256 \times 384$  verarbeiten soll, müssen die Masken entsprechend der Bildgröße angepasst werden. Das heißt, die Masken werden erst für die Original-Bilder erstellt und dann gleichermaßen auf  $256 \times 341$  verkleinert und am Rand mit schwarzen Pixeln aufgefüllt, um die Größe  $256 \times 384$  zu bekommen. Dadurch wird ein Maskenkanal erzeugt der Größe  $(256, 384, 1)$ . Dieser wird als vierter Kanal zum Farbbild zugefügt. Das erfolgt durch Stapeln (Stacking) der Maskeninformationen auf die Bilddaten als Numpy-Array. Dadurch werden die Eingabedaten des Modells zu  $(256, 384, 4)$ . Die Ausgabedaten, bzw. die Rekonstruktionen des CAE-Modells, haben daher auch die Form  $(256, 384, 4)$ . Um die Rekonstruktionen visuell zu untersuchen, müssten die Kanäle dann wieder getrennt werden, damit sowohl das Farbbild als auch das Maskenbild wiederhergestellt werden kann.

## **6.2 AE- Modelle**

Nachdem die Daten vorverarbeitet wurden, indem sie von ihrer ursprünglichen Form  $(512, 682, 3)$  auf das Format  $(256, 384, 4)$  umgewandelt wurden, kommt es zur Implementierung der CAE-Modelle.

### **6.2.1 Modell Bausteine**

Das Modell wurde so entwickelt, dass dynamisch verschiedene Modellvarianten erstellt werden können. Es besteht aus zwei Hauptkomponenten: Dem Encoder- und Decoder-Teil. Die zwei setzen sich aus den im Kapitel Planung der Modellarchitektur (5.1) geplanten Bausteinen zusammen, die je nach Eingabeparameter verwendet werden können.

### 6.2.1.1 Conv-Block

Der Convolutional-Block (Conv-Block) ist ein grundlegender Baustein, der die Conv2D-Faltung, eine Batch-Normalisierung und eine LeakyReLU-Aktivierungsfunktion kombiniert, wie zu sehen in der Abbildung 6.1. Conv2D wendet eine Faltung auf die Eingabedaten an durch einen Kernel der Größe (3,3). Durch den Stride-Eingabeparameter wird angegeben, ob der Conv-Block mit (1,1) als Teil des Deep-Conv-Blocks für das reine Feature Learning agieren soll, oder mit Stride (2,2) als Downsampling-Block, wie in Abbildung 6.2.

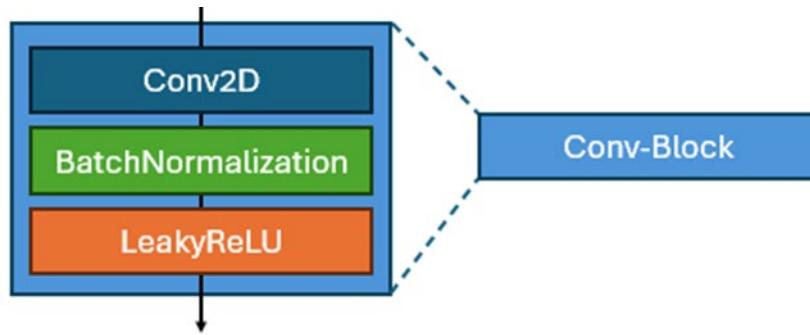


Abbildung 6.1: Conv-Block (Standard Stride = (1,1), bestehend aus einer Conv2D, Batch Normalization und LeakyReLU-Aktivierung).



Abbildung 6.2: Conv-Block (Downsampling Stride = (2,2)

Durch die Batch Normalisierung nach dem Conv-Layer bleibt die Verteilung der Eingaben stabil, während sie durch das Netzwerk fließen. Normalisierte Eingaben helfen, die Gradienten während der Backpropagation stabil zu halten, was besonders bei tiefen Netzwerken das Lernen beschleunigt und dem Problem des Vanishing Gradient mildert. LeakyReLU wird dem klassischen ReLU vorgezogen, um die Gefahr des „Neuronensterbens“ zu vermeiden, wie erwähnt in LeakyReLU (2.2.4.1). Durch den kleinen konstanten Gradienten für negative Eingabewerte verhindert LeakyReLU, dass Neuronen inaktiv werden und nicht mehr lernen.

Wie im Kapitel Padding (6.1.2) erklärt, wurde explizites Padding verwendet, um die kontrollierte Reduktion und Erhöhung der räumlichen Dimension der Feature-Maps sicherzustellen. Während jedem Conv-Layer ist ebenfalls ein Padding notwendig um sicherzustellen, dass die räumlichen Dimensionen erhalten bleiben. Beim Anwenden eines Kernels auf die Bilddimension können die Ränder des Bildes nicht vollständig abgedeckt werden, was dazu führen würde, dass die Größe der Feature-Map mit jeder Convolution verringert wird. Um sicherzustellen, dass die Ausgabe die gleiche Dimension wie die Eingabe hat, wird "padding='same'" verwendet, was die Ränder des Bildes mit zusätzlichen Pixeln auffüllt, wodurch auch die Informationen an den Bildrändern nicht verloren gehen.

### 6.2.1.2 DeConv-Block

Der Deconvolution-Block (DeConv-Block) kann als Gegenstück zum Conv-Block mit Stride (2,2) für das Upsampling angesehen werden. Er enthält eine Batch Normalisierung gefolgt vom LeakyReLU, jedoch davor eine Deconvolution mit dem Conv2DTranspose-Layer, wie zu sehen in Abbildung 6.3. Sie arbeitet gegensätzlich zu Conv2D und verdoppelt die Bilddimension bei einem Stride von (2,2).

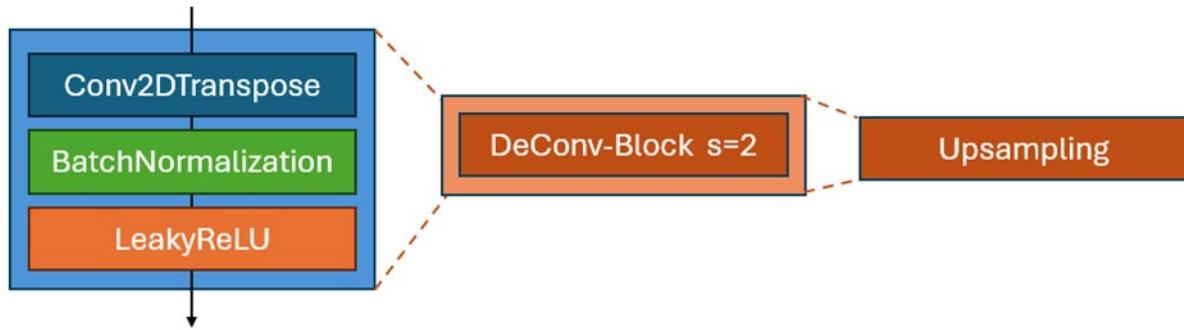


Abbildung 6.3: DeConv-Block zum Upsampling mit Stride = (2,2), bestehend aus einem Conv2DTranspose-Layer, Batch Normalization und LeakyReLU-Aktivierung.

### 6.2.1.3 Deep-Conv-Block

Der Deep-Convolutional-Block (Deep-Conv-Block) besteht aus mehreren einfachen Conv-Blöcken, wie zu sehen in Abbildung 6.4. Jeder Encoding- und Decoding-Schritt besteht aus einem Deep-Conv-Block gefolgt von einem Downsampling bzw. Upsampling. Der Grund, warum im Decoding-Prozess auch einfache Conv-Blöcke und nicht ausschließlich Conv2DTranspose verwendet werden, liegt darin, dass die Funktionen Conv2DTranspose und Conv2D zu gleichen Ergebnissen führen, wenn ein Stride von (1,1) und padding='same' verwendet wird (Dumoulin & Visin, 2016, S. 16,22).

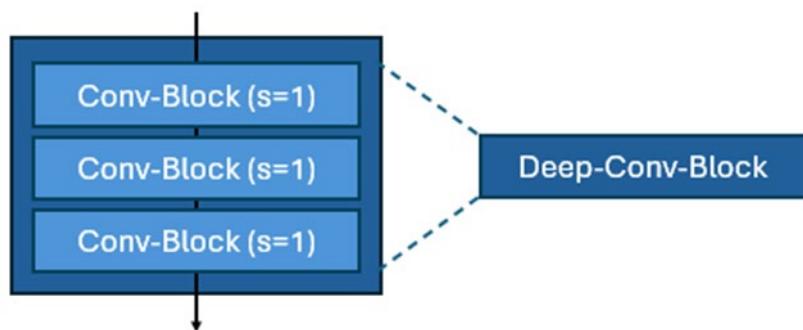


Abbildung 6.4: Deep-Conv-Block mit 3 Conv-Blöcken, zum reinen Feature Learning

Für eine tiefe Merkmalsextraktion werden drei Conv-Blöcke nacheinander angewandt. Die Erklärung, wieso die Anzahl auf drei festgelegt wurde, ist im Kapitel Ressourcenbeschränkung (6.3.1) gegeben.

#### 6.2.1.4 Residual Block

Der Residual Block ist ein Deep-Conv-Block, dem eine Residual-Connection hinzugefügt wurde, wie dargestellt in Abbildung 6.5. Diese Skip Connection leitet die Eingabedaten um den Deep-Conv-Block herum und addiert sie zur Ausgabe, die dann mit der LeakyReLU aktiviert wird. Die Skip Connection ergibt einen direkten Pfad für den Rückfluss von Gradienten während der Backpropagation. Und die Skip Connection als Identitätsfunktion ermöglicht es, dass das Modell nur bei Bedarf zusätzliche Transformationen durch den Deep-Conv-Block durchführen kann. Falls die durchgeleiteten Daten bereits ausreichend sind – den Gradienten nicht ausreichend beeinflussen würden –, können die Daten den Deep-Conv-Block „überspringen“.

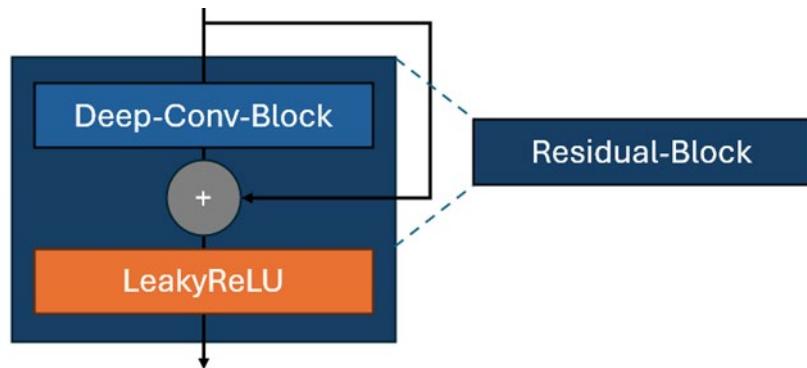


Abbildung 6.5: Residual-Block als Deep-Conv-Block mit Residual-Connection

#### 6.2.1.5 Encoding- und Decoding-Schritt

Ein Encoding-Schritt besteht aus einem Deep-Conv-Block oder Residual-Block mit anschließendem Downsampling-Block, wie in Abbildung 6.6. Ein Decoding-Schritt besteht aus einem Deep-Conv-Block oder Residual-Block mit anschließendem Upsampling-Block, dargestellt in Abbildung 6.7.



Abbildung 6.6: Encoding-Schritt mit Residual-Block für R-CAE (links) oder Deep-Conv-Block für CAE (rechts)



Abbildung 6.7: Decoding-Schritt mit Residual-Block für R-CAE (links) oder Deep-Conv-Block für CAE (rechts))

### 6.2.1.6 Output-Layer mit Sigmoid-Aktivierung

Der letzte Decoding-Schritt erzeugt wieder die Bilddimension der Eingabedaten, mit einer Anzahl von Feature-Maps, die durch das erste Filterelement angegeben wurde. Der Output-Layer des CAE nutzt einen weiteren Conv2D-Layer, jedoch mit der Sigmoid-Aktivierung, um die Tiefe der Feature-Maps auf die vier Kanäle der ursprünglichen Eingabedaten zu bringen. Sie wird deshalb verwendet, weil sie die finalen Pixelwerte auf den Wertebereich zwischen [0,1] normiert, was für die Rekonstruktion der Farbkäne und dem Maskenkanal notwendig ist.

### 6.2.2 Eingabeparameter und Datenfluss

Ob in den Encoding- und Decoding Schritte die normalen Deep-Conv-Blocks oder die Residual Blocks verwendet werden sollen, kann mit dem Eingabeparameter *autoencoder\_type* angegeben werden. Damit wird festlegt, ob das Modell ein CAE oder R-CAE sein wird. Des Weiteren wurde das Modell so aufgebaut, dass es mit einem Eingabeparameter *filters* möglich ist, die Anzahl der Encoding- und Decoding-Schritte sowie die zu verwendende Filteranzahl für jeden Schritt zu bestimmen. Für jeden angegeben Filter  $k$  in der Filters-Liste wird ein Encoding-Schritt durchgeführt, wobei das letzte Listenelement die Dimensionstiefe – die Anzahl der Feature-Maps - im Latenten Raum angibt. Dadurch regelt der Parameter direkt die Tiefe des Modells und damit die Zielgröße des Latenten Raums. Danach wird die Liste einfach rückwärts durchlaufen, um die entsprechenden Decoding-Schritte symmetrisch zu den Encoding-Schritten zu halten. Wie im Kapitel Datenvorbereitung (6.1) erläutert, wurde die Größe der Eingabedaten bewusst in die Form (256,384,4) gebracht, um das Down- und Upsampling aufeinander abzustimmen. Dadurch sind sieben Encoding-Schritte möglich, die jeweils die räumliche Dimension der Feature-Maps halbieren, wie dargestellt in Abbildung 6.8.

$$256 \times 384 - 128 \times 192 - 64 \times 96 - 32 \times 48 - 16 \times 24 - 8 \times 12 - 4 \times 6 - 2 \times 3$$

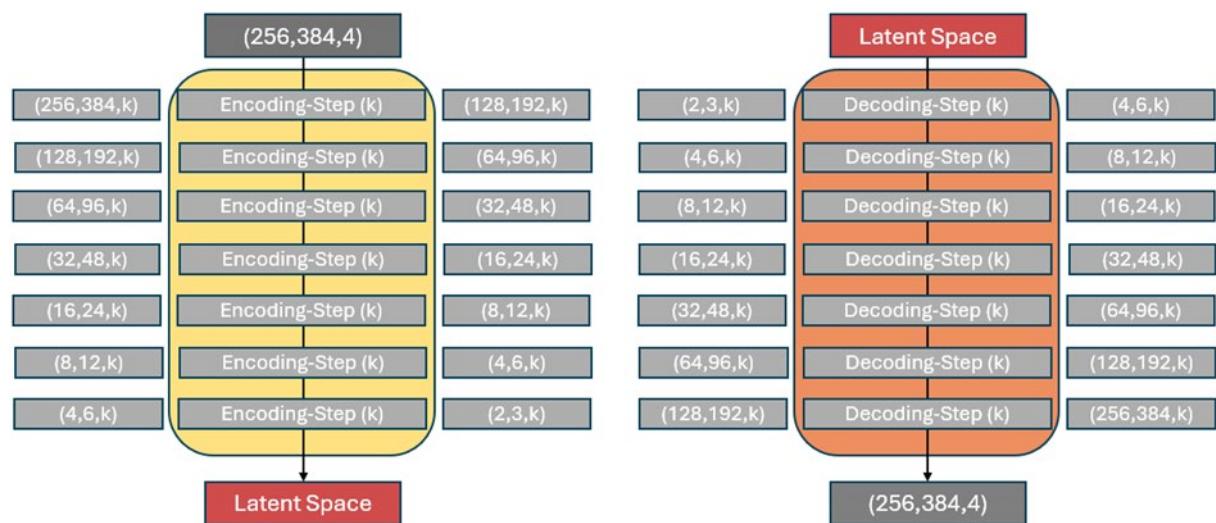


Abbildung 6.8: Aufbau des CAE/R-CAE mit Submodellen Encoder (links) und Decoder (rechts)

Angenommen, es gibt eine Konfiguration mit vier Filtern [256,128,64,32] und die Eingabebilder sind bekannt nach der Vorverarbeitung mit der Form (256,384,4). Da vier Filter in der Liste angegeben werden, finden auch vier Encoding- und Decoding-Schritte statt. Der erste Encoding-Schritt würde 256 Filter nutzen, entsprechend dem ersten Filterelement. Das bedeutet, er nutzt 256 Filter im Deep-Conv-Block zum Feature-Learning und auch im Downsampling mit Stride (2,2), die die Dimension von (256,384,256) auf (128,192,256) halbiert. Der zweite Encoding-Schritt wird 128 Filter verwenden und reduziert die Dimension von (128,192,128) auf (64,96,128). Der dritte reduziert sie mit 64 Filter von (64,96,64) weiter auf (32,48,64) und der letzte mit 32 Filtern erzeugt ausgehend von (32,48,32) mit dem letzten Downsampling den Latent Raum der Größe (16,24,32). Ein Bild, das mit dem Encoder kodiert wurde, würde somit in 32 Feature-Maps der Größe 16x24 beschrieben werden. Dann würden die Daten vom Latenten Raum mit dem Decoder wieder vergrößert werden. Für die Decoding-Schritte wird die Filter-Liste rückwärts durchlaufen. Der erste Decoding-Schritt bringt die Daten von (16,24,32) auf (32,48,32), der zweite von (32,48,64) auf (64,96,64), der dritte von (64,96,128) auf (128,192,128) und der letzte von (128,192,256) auf (256,384,256). Nach dem letzten Decoding-Schritt wird, durch den Conv-Layer mit Sigmoid-Aktivierung, die Ausgabe zurück auf die ursprüngliche Form der Eingabedaten (256,384,4) gebracht.

### 6.2.3 Erweiterung mit Dense Layer

Ähnlich wie *filters* ist *dense\_units* eine Liste, die die Anzahl der Units in den zu verwendenden Dense Layer angibt. Sie wird für die Encoding-Schritte in der angegebenen Reihenfolge verwendet und rückwärts für die Decoding-Schritte. *dense\_units* ist für die Erstellung eines Klassischen AEs mit Dense Layer gedacht. Es besteht jedoch auch die Möglichkeit, Mischformen zu erstellen, sodass erst eine Reihe von Conv-Encoding-Schritte stattfinden, um zu einer bestimmten Zielgröße zu kommen.

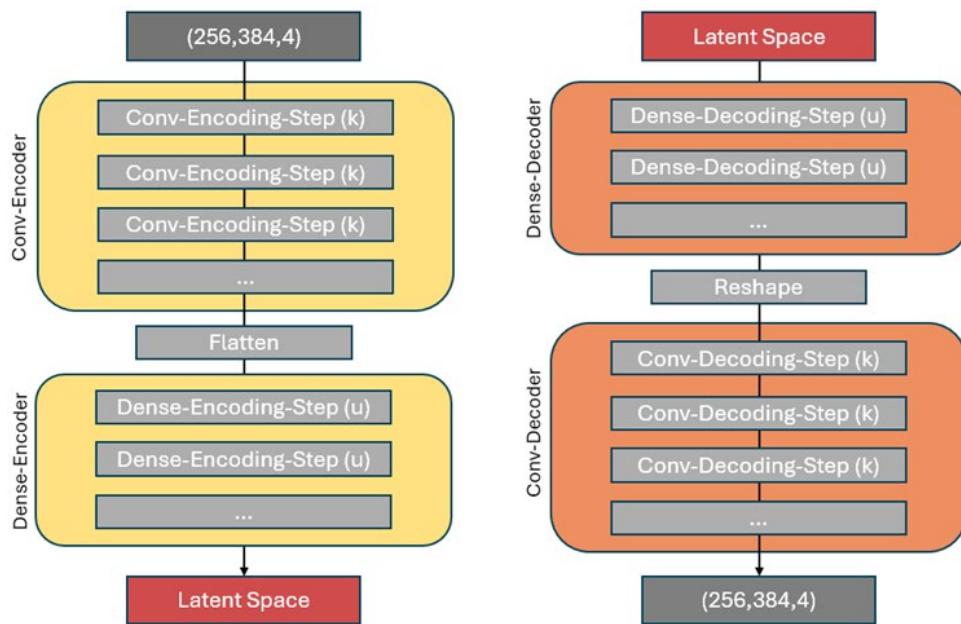


Abbildung 6.9: Mischform eines Autoencoders mit Conv- und Dense- Encoder und Decoder

---

Danach könnten die Daten durch eine Flatten-Operation in eindimensionale Vektoren umgewandelt werden, um die Dense Layer zu nutzen, wie dargestellt in Abbildung 6.9 durch:

$$x = \text{Flatten}(X) = \text{Reshape}\left(X, (n, fm_{height} * fm_{width} * dim)\right) \quad (6.1)$$

Die Units  $u$ , die in der Liste angegeben wurden, werden dann für die Dense-Encoding-Schritte verwendet. Die letzte Unit-Anzahl der Liste würde die Größe der Latenten Dimension festlegen. Symmetrisch dazu würde die Liste, während der Dense-Decoding-Schritte, rückwärts durchlaufen werden, um wieder die Größe des eindimensionalen Vektors zu erreichen, die mit der Flatten-Operation erzeugt wurde. Jetzt kann die Dimension mit einem Reshaping wieder in die ursprüngliche dreidimensionale Form zurückgeführt werden:

$$X_{reshaped} = \text{Reshape}\left(x, (n, fm_{height}, fm_{width}, dim)\right) \quad (6.2)$$

Anschließend wird die Filterliste für die Conv-Decoding-Schritte wieder rückwärts durchlaufen. Falls ausschließlich Dense Units verwendet werden sollen, um einen Klassischen AE zu erzeugen, müsste demnach einfach eine leere Filterliste mitgegeben werden. Dann würde die ursprüngliche Dimension sofort durch die Flatten-Operation eindimensional werden. Für Mischformen, wie in der Grafik beschrieben, wären eine Filterliste und eine Unit-Liste notwendig. Praktisch gesehen, sind Dense-Layer jedoch sehr ressourcenbeanspruchend. Möglichen Konfigurationen werden im Kapitel Ermittlung der Grenzen (6.3.2) beschrieben.

## 6.2.4 Leicht stärkerer Encoder

Wie in Planung der Modellarchitektur (5.1) beschrieben wird meist die Symmetrie zwischen Encoder und Decoder erhalten. Jedoch könnte ein leicht stärkerer Encoder vorteilhaft sein, wenn das Ziel die starke Kompression der Eingabedaten ist. Ein zu starker Encoder, könnte dazu führen, dass die vom Encoder erzeugten Kodierungen nicht gut rekonstruierbar sind. Aus diesem Grund sind die Anzahl der Downsamplings und Upsamplings wie aus Abbildung 6.8 symmetrisch. Allerdings ist nach dem letzten Downsampling im letzten Encoding Schritt, ein weiterer (Residual-)Deep-Conv-Block, wodurch sich eine leicht asymmetrische Gewichtung seitens dem Encoder-Modell ergibt. Das kann in Kapitel Kodierung der Datensätze (6.4) an der Abbildung 6.10 gesehen werden. Das bedeutet, dass beispielsweise bei 5 Encoding-Schritten bereits die Zielgröße des Latenten Raums erreicht wurde, danach noch ein Lernblock zum Feature-Learning kommt, quasi als Schritt „im Latenten Raum“, wobei der tatsächliche Latente Raum – als Encoder-Output -, dann die letzte Schicht dieses Schritts darstellt.

---

## 6.3 Modell-Training

In folgenden Abschnitten werden die getesteten Trainings-Konfigurationen vorgestellt nach Kapitel Geplante Trainings-Konfigurationen (5.2).

### 6.3.1 Ressourcenbeschränkung

Alle Trainings der in diesem Kapitel beschriebenen Modell-Konfigurationen wurden unter bestimmten Ressourceneinschränkungen durchgeführt. Die verfügbare GPU war eine NVIDIA Corporation GP106GL [Quadro P2000] mit 5 GB GDDR5-Speicher. In einer Vorstudie wurde versucht zu ermitteln, wann die Modell-Komplexität die Grenzen der verfügbaren Ressourcen übersteigt, unter Berücksichtigung, dass die Modelle untereinander vergleichbar sein sollen.

### 6.3.2 Ermittlung der Grenzen

Wie im Kapitel Modell Bausteine (6.2.1) beschrieben, ist das CAE-Modell modular, wobei durch alleinige Angabe der zu verwendenden Filtern und oder Units das Modell erstellt werden kann: Als Klassischer AE mit Dense Layer, CAE mit Conv-Layer, R-CAE mit erweiterten Residual-Connections oder Mischformen mit Conv-Layer und Dense-Layer. In der Theorie sind Mischformen wie in Erweiterung mit Dense Layer (6.2.3) beschrieben, möglich. Jedoch waren diese praktisch nicht testbar, da Dense Layer besonders ressourcenintensiv sind. Als Fully-Connected-Layer verbinden sie alle Neuronen einer Schicht mit allen Neuronen der vorigen Schicht.

Die (R-)CAE-Modelle enthalten Deep-Conv- oder Residual-Blöcke, die aus mehreren einfachen Conv-Blöcken bestehen. Es wurde schrittweise versucht, die Anzahl der Conv-Blöcke innerhalb der Deep-Conv-Blöcke zu erhöhen, unter Berücksichtigung der Zielgrößen des Latenten Raums, die verglichen werden sollen. Die Anzahl der Filterelemente in der Filter-Liste gibt an, wie viele Encoding- (und damit auch Decoding-) Schritte stattfinden, bis die Latente Dimension erreicht wird und welche Filteranzahl im entsprechenden Schritt verwendet werden soll. Mit jedem weiteren angegebenen Filter erhöht sich damit die Komplexität des Modells. Folglich gilt: Je kleiner die Zielgröße des Latenten Raums, desto komplexer das Modell, da mehr Encoding-Schritte erforderlich sind, um diese zu erreichen.

Je kleiner die Zielgröße, also je mehr Encoding-Schritte stattfinden und je stärker die Originaldaten damit komprimiert werden, desto mehr Merkmale gehen verloren. Wenn zu viele Merkmale wegfallen, würden auch die Rekonstruktionen schlechter werden. Nach der Anforderungsanalyse für Lern- und Auswahlverfahren (4.1) soll die Zielgröße für das nachgeschaltete Auswahlverfahren so klein wie möglich sein. Denn die durch die Kodierung gewonnenen Feature-Vektoren eines Datensatzes sollten nur die wichtigsten Charakteristiken enthalten, damit sie im direkten Vergleich mit anderen klar unterscheidbar sind. Wenn die Zielgröße zu groß gewählt wird, könnten die Feature-Vektoren viele

---

unwichtige und redundante Merkmale enthalten, die nicht repräsentativ genug sind und wenn sie zu klein gewählt wird, könnten wichtige Merkmale verloren gehen, wodurch die prägnantesten, die den Datensatz ausmachen nicht mehr erfasst werden. Um die Latente Dimension so klein wie möglich zu halten und dennoch gute Rekonstruktionen zu bekommen, wurde innerhalb der Vorstudie versucht die optimale Zielgröße zu finden. Ein Kompromiss zwischen „so klein wie möglich“ und „akzeptabler Rekonstruktion“. Das Maß an „akzeptabler Rekonstruktion“ wird durch die Evaluierungsmethoden der CAE-Modelle (4.3) bewertet und die Ergebnisse in der Evaluierung der Vorstudie (7.1.1) erläutert.

Durch die Festlegung auf eine Halbierung der Feature-Maps in jedem Encoding-Schritt und die Form der Eingabedaten von (256,384,4), steht fest welche Größen der Feature-Maps mit jedem Encoding-Schritt erreicht werden können:

$$256 \times 384 - 128 \times 192 - 64 \times 96 - 32 \times 48 - 16 \times 24 - 8 \times 12 - 4 \times 6 - 2 \times 3$$

Dabei gibt das letzte Filterelement der Filterliste in Kombination mit der Feature-Map-Größe nach der letzten Encoder-Schicht an, welche Größe der Latente Raum haben wird. Wenn das letzte Filterelement eine Filteranzahl von 64 hat und es insgesamt sieben Filter in der Liste gibt, bedeutet das, dass mit sieben Halbierungen im Latenten Raum 64 Feature-Maps der Größe 2x3 entstehen. Dadurch hätte der Latente Raum eine Dimension von (2,3,64) bzw. als flacher Vektor 384 Werte. Die zu vergleichenden Zielgrößen - als flache eindimensionale Vektoren - sind:

$$12.288 - 6.144 - 3.072 - 1.536 - 768 - 384 - 192$$

Die Tabelle 6.1 zeigt den Zusammenhang zwischen den Zielgrößen und ihren entsprechenden Kombinationen aus Feature-Map-Größe und Filteranzahl nach dem letzten Encoding-Schritt. Dabei wird verdeutlicht, wie sich die ursprünglichen Daten der Form (256,384,4) mit einer eindimensionalen Vektorgröße von 393.216 in Bezug auf die Zielgrößen reduzieren.

Latenter Raum	Filterliste	Zielgröße	Reduktion in %
(16,24,32)	[32,32,32,32]	12.288	96,88%
(8,12,64)	[64,64,64,64,64]	6.144	98,44%
(8,12,32)	[32,32,32,32,32]	3.072	99,22%
(4,6,64)	[64,64,64,64,64,64]	1.536	99,61%
(4,6,32)	[32,32,32,32,32,32]	768	99,80%
(2,3,64)	[64,64,64,64,64,64]	384	99,90%
(2,3,32)	[32,32,32,32,32,32]	192	99,95%

Tabelle 6.1: Zielgrößen des Latenten Raums und Reduktionsstärke in %

Es wurden nur Kombinationen mit konstanter Filteranzahl 32 und 64 getestet. Theoretisch gibt es auch andere Kombinationen, bei denen eine höhere Filteranzahl verwendet werden könnte, um die gleiche Zielgröße zu erreichen. Beispielsweise kann 6.144 auch durch 1.024 Feature-Maps der Größe 2x3

---

erreicht werden oder die Filteranzahl kann schrittweise erhöht oder verringert werden. Wie im Kapitel Convolutional Autoencoder (CAE) (2.2.3) erwähnt, gibt es zwei Ansätze:

- Wachsend: Beginnt mit wenigen Filtern im ersten Encoding-Schritt und verdoppelt die Anzahl schrittweise in den Encoding-Schritten und halbiert sie in den Decoding-Schritten.
- Abnehmend: Beginnt mit vielen Filtern im ersten Encoding-Schritt und halbiert die Anzahl schrittweise in den Encoding-Schritten und verdoppelt sie in den Decoding-Schritten.

Beide Strategien können für Undercomplete AEs mit dem Ziel der Dimensionsreduktion verwendet werden. Um faire Vergleichsbedingungen zu schaffen, ist es wichtig, dass die Filteranzahl für alle Encoding-Schritte nach einem System ausgewählt wird. Zum Beispiel, dass bei jeder Halbierung der räumlichen Dimension die Anzahl der Filter verdoppelt oder halbiert wird oder, dass die Filteranzahl in allen Schritten konstant gehalten wird und nur im letzten Encoding-Schritt verdoppelt oder halbiert wird. Ressourcenbedingt wurde eine konstante Filteranzahl von 32 und 64 für die Modelle gewählt. Es wurden mehrere Konfigurationen ausprobiert: Bei einem Conv-Block pro Deep-Conv-Block sind zwar verschiedene Filteranzahlen möglich, jedoch nicht für alle Zielgrößen der Latentnen Dimension.

Beispielsweise ist die Zielgröße von 6.144 mit verschiedenen Kombinationen erreichbar: (8,12,64), (4,6,256), (2,3,1024). Um eine Feature-Map-Größe von 4x6 im Latentnen Raum zu bekommen, müssten sechs 6 Downsamplings stattfinden, also 6 Filter angegeben werden. Schon die Verwendung von 256 Filtern in einer Schicht übersteigt die Ressourcenkapazität und so auch für 1.024.

Bei einer Zielgröße von 12.288 sind möglich: (16,24,32), (8,12,128), (4,6,512), (2,3,2048). Die letzten beiden können aus derselben Begründung wie zuvor ausgeschlossen werden. Bei einer Zielgröße von (8,12,128), ist es möglich mit 5 Downsamplings und 128 Filter im Latentnen Raum zu enden mit der Kombination [64,64,64,64,128]. Für die Zielgröße von 768 durch (2,3,128) gilt dies jedoch nicht. Hier würden durch die steigende Komplexität der zwei weiteren Encoding-Schritte die Konfiguration [64,64,64,64,64,128] zum Out-of-Memory führen, wobei kleinere Zielgrößen auch nicht mehr möglich wären, da die Feature-Map-Größe nicht weiter reduziert werden können, sondern nur die Filteranzahl. So ist die Konfiguration mit konstanter Filteranzahl von 64 Filtern mit 2x3 als Feature-Map-Größe möglich und ergibt eine Zielgröße von 384. Versuche mit 64 Filtern in der Latentnen Dimension zu enden, aber dennoch verschiedene Filteranzahlen zu verwenden, scheitern deshalb auch, da die Konfiguration [128,64,64,64,64,64], mit nur einer größeren Filteranzahl die Ressourcen übersteigt. Daher ist es schon bei einem Conv-Block pro Deep-Conv-Block nicht mehr möglich eine Filteranzahl zu verwenden, die gleich oder größer 128 ist und dieses Muster einheitlich für alle zu testenden Dimensionsgrößen beizubehalten. Bei zwei Conv-Blöcken pro Deep-Conv-Block sind die Grenzen natürlich viel schneller erreicht, dennoch können alle Zielgrößen mit einer konstanten Filteranzahl von 32 und 64 erreicht werden, so auch für drei Conv-Blöcke pro Deep-Conv-Block. Die Grenze war hier bei vier, weshalb die Entscheidung getroffen wurde, als Hyperparameter die Anzahl

---

der Conv-Blöcke auf drei zu wählen. Da die R-CAE Modelle Residual-Blocks verwenden, die genauso wie die Deep-Conv-Blocks aufgebaut sind, nur mit der Erweiterung einer Residual-Connection, gelten diese Grenzen auch für die R-CAE Modelle gleicher Zielgröße.

### 6.3.3 Hyperparameter

Hier eine kurze Beschreibung der gewählten Hyperparameter, die für alle Trainings gelten: Durch die Ermittlung der Grenzen (6.3.2) wurde aus Vergleichbarkeitsgründen die komplexere Variante des Deep-Conv-Blocks bzw. Residual Blocks mit drei Conv-Blocks gewählt. Ebenfalls ressourcenbedingt, wurde die Batch-Size auf 4 festgelegt. Die Form der Eingabe- und Ausgabedaten ist (256, 384, 4) nach der Datenvorbereitung (6.1). Zur Optimierung wurde *Adam* verwendet und *MSE* als Loss-Funktion.

### 6.3.4 Trainings-, Validierungs- und Testdaten

Um für das folgende Modell-Training einen fairen Vergleich zu ermöglichen, werden die Daten vorab in Trainings-, Validierungs- und Testdaten aufgeteilt. Wie im Kapitel Masken (6.1.3) erwähnt, wurden die Daten durch das Hinzufügen der Maskeninformation in die Form (256,384,4) gebracht und als Numpy-Arrays abgespeichert. Alle synthetischen Datensätze enthalten somit jeweils 14.000 Numpy-Arrays und der Real-Datensatz 5.452. Die Trainingsdaten und Anzahl, die in der Vorstudie verwendet wurden, sind nicht dieselben, wie die die in der Hauptstudie verwendet wurden. Für die Vorstudie wurde auf 6 Datensätzen trainiert, mit jeweils 2.500 Trainingsdaten. In der Hauptstudie wurde auf den 4 (Basis-) Datensätze trainiert, mit jeweils 3.500 Trainingsdaten. Die Daten werden nach der Standard-Ratio 0.2 in Train und Val aufgeteilt. Bei 2.500 Trainingsdaten in der Vorstudie bedeutet das, dass weitere 500 Validierungsdaten jedes Datensatzes notwendig sind. Für die Trainings der Hauptstudie mit 3.500 Trainingsdaten werden 700 Validierungsdaten gebraucht. Da Daten von mehreren Datensätzen für die Trainings verwendet werden, ist der Anteil an verbliebenden Testdaten pro Datensatz sehr hoch. In beiden Fällen – der Vorstudie und den nachfolgenden Trainings – wurden die entsprechende Anzahl an Daten zufällig gewählt und dann konstant für alle Trainings verwendet.

### 6.3.5 Konfigurationen der Vorstudie

Die getesteten Konfigurationen des Kapitels Ermittlung der Grenzen (6.3.2) waren Teil einer Vorstudie, die zum Ziel hatte, die optimale Latente Dimensionsgröße zu finden, die eine maximale Datenkompression unter Bewahrung der Rekonstruktionsqualität liefert. Für einen fairen Vergleich wurde festgelegt, dass die Filteranzahl in jedem Encoding-Schritt konstant bei 32 oder 64 liegt. Für die Vorstudie wurden jeweils 2.500 Trainingsdaten und 500 Validierungsdaten der folgenden sechs Datensätze verwendet, wodurch auf insgesamt 15.000 Daten trainiert wurde:

- 
1. adidas\_syn\_dr\_real\_texture
  2. adidas\_syn\_simple
  3. adidas\_syn\_simple\_real\_texture
  4. adidas\_syn\_se\_random\_light
  5. adidas\_syn\_se
  6. adidas\_real

Die Trainings der Vorstudie waren sequenziell. Das heißt, dass das Modell nacheinander auf den Trainingsdaten der sechs Datensätze trainiert wurde, bevor zum nächsten übergegangen wird. Insgesamt wurden die Modelle der Vorstudie für 300 Epochen trainiert, 50 Epochen pro Trainingsdaten eines Datensatzes. Der Validation Loss der Modelle der Vorstudie können im Anhang Validation Loss (E) nachvollzogen werden. Dadurch passieren schrittweise Modellanpassungen auf Grundlage jedes spezifischen Datensatzes bzw. ihrer Domänen-Eigenschaften. Jedoch besteht die Gefahr, dass das sequenziell trainierte Modell dazu neigen könnte, die Merkmale des zuletzt gelernten Datensatzes zu bevorzugen. Da die Annahme besteht, dass R-CAEs eine bessere Leistung als CAEs erzielen, wurden in der Vorstudie bereits alle Modelle als R-CAE definiert. Die Tabelle 6.1 zeigt die für die Vorstudie verwendeten Filterkonfiguration und die resultierenden Zielgrößen in insgesamt sieben Modellen. Je kleiner der Latente Raum, desto stärker die Kompression. Es ist daher zu erwarten, dass die Qualität der Rekonstruktionen abnimmt, mit Abnahme der Zielgröße. Die Evaluierung der Vorstudie (7.1.1) zeigt den direkten Vergleich der sieben Modelle, mit 3.072 und 6.144 als weiter zu untersuchende Zielgrößen.

### **6.3.6 Konfigurationen der Hauptstudie**

Im Rahmen der Hauptstudie wurden weitere Modelle trainiert, basierend auf den optimalen Zielgrößen aus der Vorstudie. Nun mit Trainingsdaten der vier Basis-Datensätze:

[adidas\_syn\_dr, adidas\_syn\_simple, adidas\_syn\_se, adidas\_real]

Die sequenziell trainierten Modelle wurden für 120 Epochen trainiert – 30 Epochen pro 3.500 Trainingsdaten eines Basis-Datensatzes. Die kombiniert trainierten Modelle mit insgesamt 14.000 Daten wurden für 30 Epochen trainiert. In der Evaluierung der Hauptstudie (7.1.2) werden die Ergebnisse erklärt und bewertet. Der Validation Loss der Modelle der Hauptstudie ist auch im Anhang Validation Loss (E) hinterlegt.

#### **6.3.6.1 Sequenziell: CAE vs. R-CAE**

Zuerst wurden vier Konfigurationen getestet für einen direkten Vergleich zwischen jeweils zwei CAE- und R-CAE Modellen mit denselben Zielgrößen, um den Einfluss der Residual-Connections auf die Modellleistung zu untersuchen. Die ermittelten optimalen Zielgrößen der Vorstudie waren (8,12,32) – 3.072 mit der Filterliste [32,32,32,32,32] und (8,12,64) – 6.144 mit Filterliste [64,64,64,64,64]. Die Reihenfolge des sequenziellen Trainings war - DR - Simple - SE – Real -. Beginnend mit DR wird die

---

Vielfalt simuliert, dann durch Simple auf grundlegende Strukturen fokussiert, über SE detailliertere Szenarien eingeführt, bis hin zu realen Daten. Alle folgenden Trainings-Konfigurationen beziehen sich ausschließlich auf R-CAE Modelle.

### **6.3.6.2 Sequenziell: Datensatz-Reihenfolge**

Danach wurden weitere zwei Konfigurationen getestet, die sich auf die Datensatz-Reihenfolge des sequenziellen Trainings beziehen. Es werden dieselben R-CAE Modelle – mit Größen 3.072 und 6.144 – mit Modellen verglichen, die auf der umgekehrten Reihenfolge von - Real – SE – Simple – DR – trainiert wurden und – Simple – DR – SE – Real –, um zu untersuchen, welchen Einfluss die Reihenfolge der Domänen hat und ob das Modell dazu fähig ist, die Simple-Domäne als eigene Domäne zu erfassen oder diese nur als Ausgangspunkt nutzt und sie später mit komplexeren Merkmalen überschreibt.

### **6.3.6.3 Kombiniert: R-CAE mit kleinen Zielgrößen**

Als nächstes wurden die Konfigurationen auf kombinierten Daten untersucht. Das heißt, alle Trainingsdaten der vier Basis-Datensätze wurden gemischt und auf Batches von kombinierten Daten trainiert. Bei der Batch-Größe von 4 enthält jede Batch jeweils ein Datenpunkt eines Basis-Datensatzes. Nach der Evaluierung der Vorstudie (7.1.1) wurde die optimale Größe des Latenten Raums auf 6.144 und 3.072 geschätzt. Es wurden die kleineren Zielgrößen – 1.536 – 768 – 368 – 192 – nochmals getestet mit denselben Filterkonfigurationen der entsprechenden Zielgröße der Vorstudie. Zum einen als direkter Vergleich zu den sequenziell trainierten Modellen. Zum anderen, da auch die Modelle mit visuell schlechterer Rekonstruktionsqualität zur Kodierung getestet werden, um zu prüfen, ob unterscheidbare Merkmale in den Datensatz-Kodierungen vorhanden bleiben, auch wenn sie in den Rekonstruktionen nicht erkennbar sind.

### **6.3.6.4 Kombiniert: Abnahme der Real-Trainingsdaten**

Parallel wurde ausgehend von den optimalen Zielgrößen 3.072 und 6.144 der Vorstudie versucht die Anzahl an Real-Trainingsdaten schrittweise zu reduzieren von – 3.500 – 2.500 – 1.500 – 500 –. Es soll getestet werden, wie stark der Anteil an Realdaten reduziert werden kann, damit die Real-Domäne dennoch gut erfasst wird bzw. wie das die Rekonstruktionsqualität der Realdaten beeinflusst.

### **6.3.6.5 Kombiniert: Filteranzahl-Strategie**

Zuletzt wurden auch im kleinen Stil versucht den Einfluss der Filteranzahl-Strategie zu testen. Es wurde die Zielgröße 1.536 mit der absteigenden Filterkonfiguration [64,64,32,32,16] getestet und die wachsende Filterkonfiguration [16,16,32,32,64,64]. Die Zielgröße ist gleich, jedoch ist die Form des Latenten Raums beim Ansatz „absteigend“ (8,12,16), da 5 Encoding-Schritte stattfinden und bei „wachsend“ (4,6,64), da 6 Encoding-Schritte stattfinden.

## 6.4 Kodierung der Datensätze

Die Kodierung aller 17 Datensätze kann mit allen Modellen aus der Vor- und Hauptstudie durchgeführt werden. Zur Veranschaulichung wie die Kodierungen weiterverwendet wurden, werden im Folgenden die Datensatz-Kodierungen von „r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500“ verwendet. Das Modell wurde auf kombinierten Daten trainiert mit Real-Anteil von 3.500 und Zielgröße von (8,12,32), die durch 5 Encoding-Schritte erreicht wurde mit konstanter Filter-Konfiguration [32,32,32,32,32].

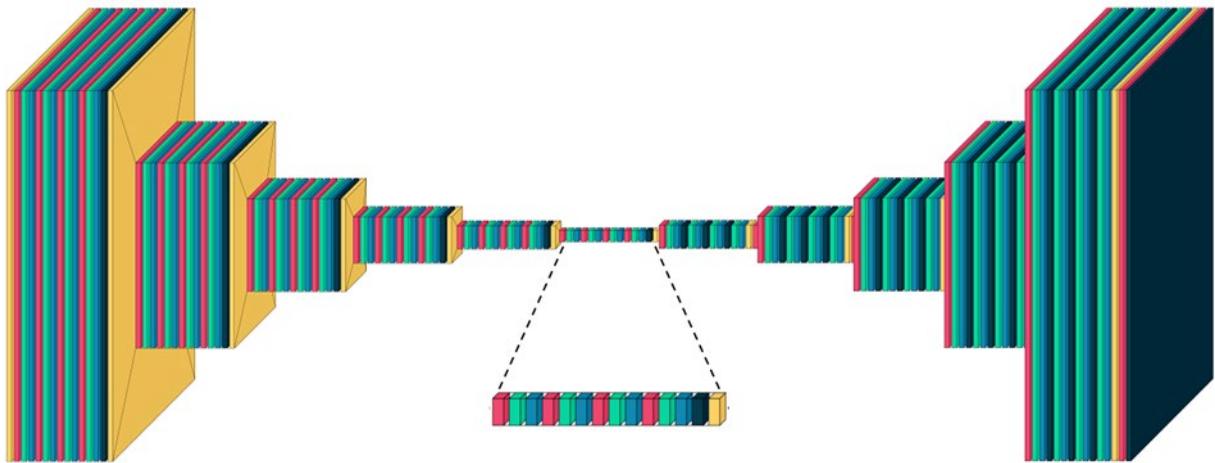


Abbildung 6.10: R-CAE Modell „r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500“ mit 5 Downsamplings im Encoder zum Latentten Raum gefolgt von 5 Upsamplings im Decoder zu der ursprünglichen Dimension der Eingabedaten:  
 $256 \times 384 - 128 \times 192 - 64 \times 96 - 32 \times 48 - 16 \times 24 - 8 \times 12 - 16 \times 24 - 32 \times 48 - 64 \times 96 - 128 \times 192 - 256 \times 384$

Zur Kodierung wird nur das Encoder-Submodell benötigt. Der hervorgehobene Bereich in Abbildung 6.10 zeigt die Daten nach dem letzten Encoding-Schritt. Hier haben die Daten bereits die Form (8,12,32). Der Latente Raum wird mit der letzten Aktivierungsschicht nach dem Lernblock im Latentten Raum erreicht (als Teil des Encoders), bevor das erste Upsampling durch den Decoder passiert. Das bedeutet, wenn Datensätze der Form ( $n, 256, 384, 4$ ) zu ( $n, 8, 12, 32$ ) kodiert werden, werden für die  $n$  Bilder  $n$  Feature-Vektoren erzeugt, wobei je ein Feature-Vektor 32 Feature-Maps der Größe  $8 \times 12$  hat.

In Abbildung 6.11 wird ein Sample vom Real-Datensatz mit den Kodierungen gezeigt. Da sie nach der Vorverarbeitung als  $(256, 384, 4)$  Numpy-Array vorliegen, werden sie in ihre ursprünglichen drei Farbkanäle RGB und den Maskenkanal aufgeteilt.

Die Feature-Maps im Latentten Raum enthalten die wichtigsten Informationen über die Eigenschaften der Eingabebilder. Dass die Merkmale mit jedem Encoding-Schritt abstrakter und komplexer werden, ist auch in den Heat Maps erkennbar. Anfangs sind die Eigenschaften, die die Feature-Maps repräsentieren, wie Kanten, Texturen oder das Logo deutlicher zu erkennen, wie in Abbildung 6.12. Im Latentten Raum sind sie jedoch nicht mehr so einfach interpretierbar, da die Struktur und Inhalte der Eingabebilder, sich als Kombination der einfachen Merkmale zusammensetzen und nun komplexe Merkmale sind, wie zu sehen in Abbildung 6.13.

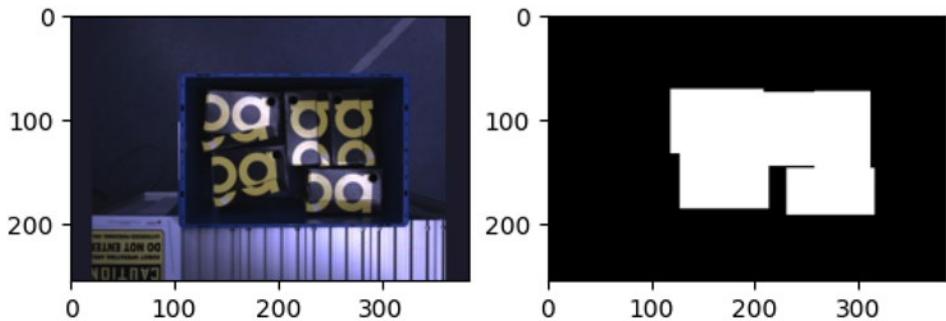


Abbildung 6.11: Sample vom Datensatz "adidas\_real" (preprocessed\_adidas\_real\_109)

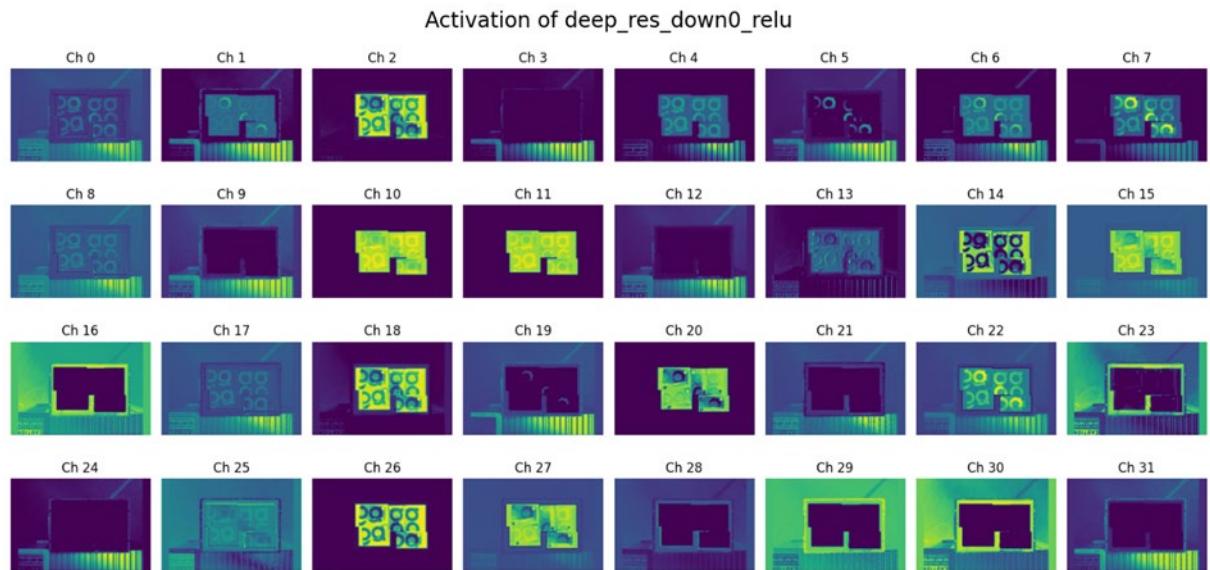


Abbildung 6.12: Aktivierung nach dem ersten Encoding-Schritt, vor dem Downsampling

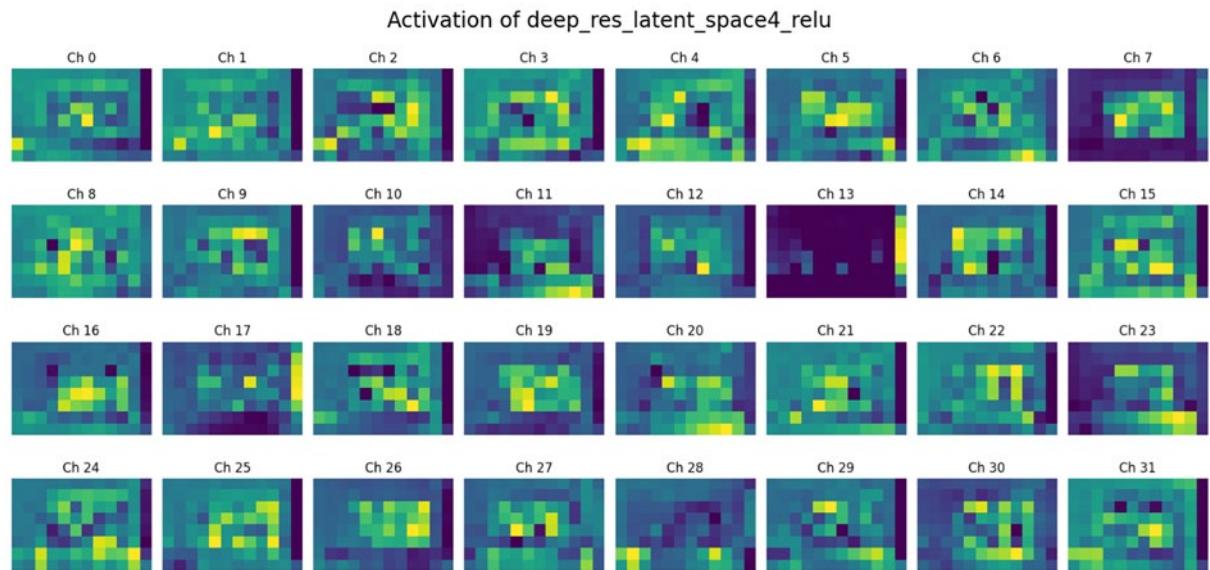


Abbildung 6.13: Aktivierungen des Latenten Raums

Dies war nur ein Sample aus dem Real-Datensatz. Es wurden alle Bilder aller Datensätze mit diesem Modell kodiert und somit 17 Datensätze von  $(n, 256, 384, 4)$  zu  $(n, 8, 12, 32)$  kodiert.

## 6.5 Aggregation der Feature-Vektoren

Nach der Kodierung gibt es für jedes Original-Bild einen Feature-Vektor mit 32 Feature-Maps. Zur Datensatz-Analyse und zur Erstellung eines Datensatz-Repräsentanten wird eine Aggregation über die Feature-Vektoren durchgeführt, um die Informationen aller Datenpunkte zu verdichten. Bei einem kodierten Datensatz der Form  $(n, 8, 12, 32)$  bezieht sich die erste Achse ( $Axis = 0$ ) auf die  $n$  Feature-Vektoren, die zweite und dritte ( $Axis = 1, 2$ ) auf die Pixelposition in einer Feature-Map und die vierte Achse ( $Axis = 3$ ) auf die Anzahl der Feature-Maps. Zur Erstellung der Datensatz-Repräsentanten wird über ( $Axis = 0$ ) aggregiert. Dadurch wird der Durchschnitts-Wert für jeden Pixel, angegeben durch Index  $ij$ , in jeder Feature-Map  $k$  errechnet und das über alle  $n$  Feature-Vektoren von Datensatz  $X$ :

$$mean_{fv}(i, j, k) = \frac{1}{n} \sum_{l=1}^n X_{l, i, j, k} \quad (6.3)$$

Dadurch ergibt sich ein Durchschnitts-Feature-Vektor der Form  $(8, 12, 32)$ . Wird ein Durchschnitts-Feature-Vektor für jeden Datensatz erstellt, enthält er den typischen Aktivierungszustand jeder Feature-Map im gesamten Datensatz. Damit können die Datensatz-Repräsentanten im Auswahlverfahren miteinander verglichen werden. Diese Repräsentanten können auch durch Heat Maps visualisiert werden. Es wird erwartet, dass ähnliche Datensätze auch ähnliche Muster in ihren Durchschnitts-Feature-Vektoren zeigen. Wie beschrieben im Kapitel Weiterverarbeitung der Ergebnisse (5.3) gibt es neben dem Durchschnitt weitere Aggregationsmethoden, die verschiedene Aspekte der Daten hervorheben würden. Auch relevant zur Visualisierung ist die Standardabweichung, da sie zeigt, wie stark die Aktivierungen auf einer Feature-Map variieren und in welchem Bereich. Deswegen wird auch für jeden Datensatz ein Std-Dev-Feature-Vektor erstellt:

$$stdDev_{fv}(i, j, k) = \sqrt{\frac{1}{n} \sum_{l=1}^n (X_{l, i, j, k} - mean_{fv}(i, j, k))^2} \quad (6.4)$$

Um die Varianz für jedes Merkmal zu berechnen, wird über die Pixelpositionen ( $Axis = 1, 2$ ) und  $n$  Feature-Vektoren ( $Axis = 0$ ) aggregiert. Dadurch wird die räumliche Struktur jeder Feature-Map zusammengeführt, und die Varianz über alle Pixelpositionen  $ij$  einer Feature-Map  $k$  berechnet. Die Variabilität eines Features kann als Wichtigkeit angesehen und für die Erstellung des Rankings Variance-Penalty-Reward (6.6.6) verwendet werden. Für jede Feature-Map entsteht ein Varianz-Wert:

$$Var_k = \frac{1}{n} \sum_{l=1}^n \sum_{i=1}^{fm\_height} \sum_{j=1}^{fm\_width} (x_{lijk} - \mu_k)^2 \quad (6.5)$$

---

## 6.6 Distanzbasiertes Auswahlverfahren

Nach der Kodierung und Aggregation werden alle Durchschnitts-Feature-Vektoren als Datensatz-Repräsentanten in einem dreistufigen Auswahlverfahren miteinander verglichen und in ein Ranking gebracht. Nach der Evaluierung der Hauptstudie (7.1.2), geht hervor, dass das Modell „r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500“ trotz der starken Reduktion von 99,22% noch einen guten SSI-Wert für die getesteten Datensätze hat und damit die Bildstrukturen erfassen kann. Das ist wichtig, denn die Anwendung des Auswahlverfahrens setzt voraus, dass die Ergebnisse der Kodierung und Aggregation aussagekräftig genug sind, damit sie noch die charakteristischen Eigenschaften des Datensatzes mit ihren Domänen- und spezifischen Eigenschaften enthalten.

### 6.6.1 Grundlage der Bewertung

Wie in Kapitel Planung des Rankings durch Auswahlverfahren (5.4) erwähnt, existieren Performance-Scores für bestimmte Datensätze einer bereits durchgeföhrten Objekterkennungsaufgabe.

Dataset	adidas real	adidas syn se	adidas syn simple	adidas syn dr
mAP	88,99	75,53	28,34	10,73
Std Dev	0,37	1,15	0,79	2,24

*Tabelle 6.2: Performance-Scores der Basis-Datensätze einer Objekterkennungsaufgabe*

Die Ergebnisse der Tabelle 6.2 zeigen eine deutliche Abstufung der Objekterkennungs-Genauigkeit. Der Real-Datensatz erzielt die beste Leistung, gefolgt vom realimitierenden SE-Datensatz. Zum minimalistisch-abstrakten Simple-Datensatz ist ein großer Leistungsunterschied erkennbar. Der randomisierte DR-Datensatz zeigt die schlechteste Leistung. Da diese vier die Basis-Datensätze sind und als Repräsentanten ihrer entsprechenden Domäne – Real – SE – Simple – DR – gelten, werden diese Referenzwerte für die Bewertung der Datensätze und dem Ranking verwendet, mit dem Wissen, wie sie ordinal zueinanderstehen. Da für Basis- und Varianten-Datensätze Repräsentanten erstellt wurden, kann verglichen werden, inwiefern die Varianten-Repräsentanten den Basis-Repräsentanten ähneln, um auf Grundlage ihrer Domänenzugehörigkeit ihre grobe Performanz zu schätzen. Ist ein Varianten-Datensatz sehr ähnlich zu einem Basis-Datensatz, wird eine ähnliche Performanz erwartet.

Domäne	Basis-Datensätze (Domänen-Repräsentanten)	Varianten-Datensätze
Real	adidas_real	-
SE	adidas_syn_se	adidas_syn_se_color_texture adidas_syn_se_random_texture adidas_syn_se_random_light adidas_syn_se_static_light adidas_syn_se_random_camera adidas_syn_se_no_lc_table
Simple	adidas_syn_simple	adidas_syn_simple_real_texture adidas_syn_simple_random_texture adidas_syn_simple_real_light adidas_syn_simple_random_light adidas_syn_simple_real_camera
DR	adidas_syn_dr	adidas_syn_dr_color_texture adidas_syn_dr_real_texture

Tabelle 6.3: Domänenzugehörigkeiten der Varianten-Datensätze und Basis-Datensätze als Domänenrepräsentanten

Nach Domänen-Wissen ist bekannt, dass die übrigen 13 Varianten-Datensätze zu einen der drei synthetischen Domänen angehören. Die Zugehörigkeiten können in Tabelle 6.3 nachvollzogen werden.

## 6.6.2 Aufstellung der Distanzmatrix und Normalisierung

Zur Berechnung der Ähnlichkeit zwischen den Datensatz-Repräsentanten wurde ein Distanzbasierter Ansatz (5.4.1) implementiert. Der Wertebereich der Feature-Vektoren wird durch die LeakyReLU-Aktivierung der letzten Encoder-Schicht bestimmt. LeakyReLU erlaubt es, dass für negative Eingabewerte ein kleiner, nicht-null Gradient existiert, abhängig vom Leck-Parameter, der standardmäßig auf 0,01 belassen wurde. Das bedeutet, dass die Ausgaben des Encoders leicht negative Werte für bestimmte Aktivierungen annehmen können, während positive Eingaben unverändert bleiben. Nach der Aggregation zu den Durchschnitts-Feature-Vektoren als Datensatz-Repräsentanten, werden diese normalisiert, damit sie für den Vergleich miteinander in einem konsistenten Wertebereich liegen. Dafür wird der Robust-Scaler verwendet, der den Median entfernt und die Daten anhand des Quantil-Bereichs skaliert. Die Distanzmatrix wird daher auf den vom Robust-Scaler normalisierten Datensatz-Repräsentanten erstellt.

Die Distanzmatrix für die Datensatz-Repräsentanten  $D_{rep} = \{d_1, d_2, \dots, d_m\}$  auf Feature-Vektor-Level:

$$D = \begin{pmatrix} D_{11} & D_{12} & \cdots & D_{1m} \\ D_{21} & D_{22} & \cdots & D_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ D_{m1} & D_{m2} & \cdots & D_{mm} \end{pmatrix} \quad (6.5)$$

---

Dann wird auf Feature-Map-Level die Distanzmatrix für jede Feature-Map aufgestellt, wobei  $D_{ij}^{(k)}$  die Distanz zwischen den Feature-Maps  $k$  der Repräsentanten  $d_i$  und  $d_j$  angibt:

$$D^{(k)} = \begin{pmatrix} D_{11}^{(k)} & D_{12}^{(k)} & \dots & D_{1m}^{(k)} \\ D_{21}^{(k)} & D_{22}^{(k)} & \dots & D_{2m}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ D_{m1}^{(k)} & D_{m2}^{(k)} & \dots & D_{mm}^{(k)} \end{pmatrix} \quad (6.6)$$

Mit dem Eingabeparameter *distance\_type* kann bestimmt werden, ob die Euklidische Distanz oder die Manhattan Distanz als Distanz-Funktion *dist* für die Berechnung gewählt werden soll. Beide Distanz-Metriken arbeiten auf flachen Vektoren. Das heißt, dass die Datensatz-Repräsentanten der Form (8,12,32) in eindimensionale Vektoren mit 3.072 Werten transformiert werden. Beim Vergleich zweier flacher Datensatz-Repräsentanten, wird jeweils die Distanz der Werte berechnet, die an derselben Position in beiden Vektoren – der sequenziellen Liste – stehen. Das heißt, die Werte beziehen sich zwar auf eine spezifische Pixelposition einer spezifischen Feature-Map, jedoch wird durch die summierte Gesamtdistanz nicht mehr ersichtlich, welche der 8x12 Pixelpositionen und welche der 32 Feature-Maps am meisten beitragen, da kein Bezug mehr zur räumlichen Struktur besteht.

Für den ersten Vergleich ist das jedoch ein Vorteil. Die Distanzmessung auf Feature-Vektor-Level wird als ganzheitliches Maß für Unterschiedlichkeit oder Ähnlichkeit zwischen zwei Datensatz-Repräsentanten gewählt, um Gruppen und Domänen zu erkennen, die durch isolierte Betrachtung einzelner Feature-Maps nicht eindeutig identifizierbar wären. Es wird zwar angenommen, dass die meisten Feature-Maps die Domänen-Eigenschaften des Datensatzes in irgendeiner Form enthalten, jedoch ist jede Feature-Map die Darstellung eines komplexen Merkmals. Es könnte sich daher auch um ein Merkmal handeln, welches über Domänen hinweg konsistent ist, was dadurch nicht unbedingt zur Unterscheidung beitragen würde. Denn die Kombination aller Merkmale macht letztendlich den Datensatz mit seiner Domäne und Gruppeneigenschaften (wie Lichtverhältnisse, Kameraposition, etc.) identifizierbar. Daher würden die Distanzen aller Merkmalsunterschiede mehr zur Unterscheidung beitragen. Die Distanzen auf Feature-Vektor-Level als Gesamtdistanzen sind damit ein Maß zur Gesamtähnlichkeit zwischen zwei Repräsentanten.

### 6.6.3 Erster Schritt: Domänen-Reihenfolge

Die Referenzwerte der Objekterkennungsaufgabe zeigen eine Abstufung der Performanz in der Reihenfolge – Real, SE, Simple, DR -. Mit der Bestimmung der Domänen-Reihenfolge soll überprüft werden ob Datensätze, die tendenziell besser abschneiden, näher zum Real-Datensatz sind bzw. ob die mit schlechteren Performance-Scores weiter entfernt sind – also ob es eine Korrelation mit der beobachteten Leistungsabstufung mit der Real-Nähe gibt.

---

Die Basis-Datensätze werden definiert als  $B = \{s_1, s_2, \dots, s_i, r\}$  mit  $s_i$  als Synthetischer-Basis-Datensatz-Repräsentant und  $r$  als Real-Repräsentant. Ausgehend von der aufgestellten Distanzmatrix auf Feature-Vektor-Level, werden die Distanzen der synthetischen Basis-Datensätze  $S = \{s_1, s_2, \dots, s_i\}$  zum Real-Datensatz  $r$  genommen:

$$dist_i = dist(s_i, r), \text{ für } i \in S \quad (6.7)$$

Konkret zeigen die Distanzen, welcher der drei synthetischen Basis-Datensätze, als  $s_{se}$ ,  $s_{simple}$ ,  $s_{dr}$  dem Real-Datensatz  $r$  am nächsten ist, der zweitnächste und drittstärkste. Diese bekommen dann ein vorläufiges ordinales Ranking ausgehend von ihrer Real-Nähe:

$$Rank_i = Rang(dist_i), \text{ für } i = 1, 2, \dots, i \quad (6.8)$$

Demnach würde der Real-Datensatz immer den höchsten Rank haben. Allerdings kann angegeben werden, nach welcher Datensatz-Nähe die Reihenfolge bestimmt werden soll. So könnte nach Anwendungsfall auch die Nähe zum SE-Datensatz als Ausgangspunkt genommen werden. Im Kapitel Distanzbasierter Ansatz (5.4.1) wurde auch angedeutet, dass die Distanzen zum Real-Datensatz nicht der erwarteten Abstufung der Objekterkennungsaufgabe entsprechen könnten. Ist dies der Fall, so kann die Reihenfolge auch explizit angegeben werden, um die Domänen-Reihenfolge zu erzwingen.

#### 6.6.4 Zweiter Schritt: Domänen-Zugehörigkeit

Im zweiten Schritt werden die Distanzen der Varianten-Repräsentanten  $V = \{v_1, v_2, \dots, v_j\}$  zu den vier Basis-Repräsentanten  $B$  genommen, um ihre Domänen-Zugehörigkeit zu bestimmen. Für eine eindeutige Zuordnung muss ein Schwellenwert  $\alpha$  festgelegt werden, ab dem der Varianten-Datensatz als ausreichend ähnlich genug zum Domänenrepräsentierenden Basis-Datensatz angesehen wird, um eine Zuordnung zu rechtfertigen:

$$v_j = \begin{cases} Rank v_j = Rank b_k, & dist(v_j, b_k) \leq \alpha \\ Undefined, & dist(v_j, b_k) > \alpha \end{cases} \quad (6.9)$$

Der Schwellenwert wird standardmäßig auf  $\alpha = 0,3$  festgelegt, kann jedoch individuell angepasst werden, um eine strengere Zuordnung zu ermöglichen oder mehr Flexibilität. Die Wahl des Schwellenwerts wird genauer im Kapitel Evaluierung der Domänen-Zuordnung (7.3.4) beschrieben. Die Varianten-Repräsentanten, die eine Distanz unter diesem Schwellenwert zu einem der vier Basis-Repräsentanten haben, werden zur entsprechenden Domäne zugeordnet. Da für diese eine ähnliche Performanz erwartet wird, bekommen sie in dem Schritt dasselbe ordinale Ranking, welches nach Erster Schritt: Domänen-Reihenfolge (6.6.3) ermittelt wurde. Die, die über dem Schwellenwert liegen, können nicht eindeutig zugeordnet werden und bekommen daher die Zuordnung *undefined* und würden im

---

Domänenbasierten Ranking nicht weiter berücksichtigt werden. Idealerweise sollte die Domänen-Zugehörigkeit aller Varianten-Datensätze wie in Tabelle 6.3 bestimmt werden.

Alternativ wurde auch der Ansatz implementiert natürliche Domänen bilden zu lassen ohne die Basis-Repräsentanten als Referenzwerte. Dafür werden alle Distanzen der Distanzmatrix auf Feature-Vektor-Level verglichen und die Datensätze gruppiert, die unter dem angegebenen Schwellenwert liegen. Dadurch sollen potenzielle Subgruppen innerhalb der Domänen oder über Domänen-Grenzen hinweg identifiziert werden.

### 6.6.5 Dritter Schritt: Real-Nähe

Nach dem zweiten Schritt haben alle Datensätze einer Domäne dieselbe Wertigkeit im ordinalen Ranking. Um eine genauere Abschätzung zu bekommen, werden die Datensätze innerhalb ihrer Domäne nach ihrer Real-Nähe neu eingestuft, wobei wieder angegeben werden kann, ob diese tatsächlich nach der Real-Datensatz-Nähe oder nach einer anderen Datensatz-Nähe geordnet werden sollen.

### 6.6.6 Variance-Penalty-Reward

Neben dem Domänenbasierten Ansatz und der Berücksichtigung der Real-Nähe können auch einzelne Feature-Maps analysiert werden, indem wichtige Features stärker in ihrer Real- und DR-Ähnlichkeit berücksichtigt werden. Jeder Datensatz-Repräsentant  $d_i$  bekommt eine  $Score_i$ , als Importance Score.

Es wird der Varianz-Wert genommen, der durch die Aggregation der Feature-Vektoren (6.5) für jede Feature-Map eines Datensatzes bestimmt wurde. Da die Varianz die Variabilität des durch die Feature-Map repräsentierten Merkmals misst, kann eine hohe Varianz als Wichtigkeit eines Features betrachtet werden. Weil das Ranking von der Real-Wichtigkeit ausgeht, werden die Varianz-Werte der Feature-Maps des Real-Datensatzes  $Var_{Real} = [Var_1, Var_2, \dots, Var_{dim}]$  genutzt, um ein Ranking zu erstellen. Dabei werden Features mit höherer Varianz besser eingestuft. Daraus entsteht eine Rangliste der Features von  $[f_1, f_2, \dots, f_{dim}]$ , als Feature-Importance-Ranking, wobei  $f_i$  der Rang des Features mit der  $i$ -höchsten Varianz ist. Bei 32 Feature-Maps ( $dim = 32$ ) des Real-Repräsentanten, könnte zum Beispiel Feature-Map 14 die höchste Variabilität aufweisen und dadurch auf dem ersten Platz sein. Wie für die Domänen-Reihenfolge kann jedoch auch direkt angegeben werden, welcher Datensatz-Repräsentant mit seinen Feature-Maps, als Basis für das Feature-Importance-Ranking verwendet werden soll. Das Feature-Importance-Ranking dient als Grundlage zur Gewichtung der Distanzen auf Feature-Map-Level. Für jede Feature-Map eines Varianten-Datensatzes wird die Distanz zur Feature-Map des Real- und DR-Basis-Datensatzes aus der entsprechenden Feature-Map-Distanzmatrix genommen. Wenn die Distanz der Varianten-Feature-Map zur Real-Feature-Map kleiner ist als die Distanz zur DR-Feature-Map, soll der Datensatz dafür belohnt werden, und umgekehrt bestraft, wenn die Distanz zur DR-Feature-Map kleiner ist. Die Belohnung und Bestrafung wird dabei durch den Rang

---

der Feature-Map aus dem Feature-Importance-Ranking gewichtet. Das bedeutet, dass wichtigere Features (gemessen an Variabilität) einen größeren Einfluss auf die Importance Score haben, sowohl positiv als auch negativ. Der Importance-Score eines Repräsentanten  $d_i$  wird berechnet durch:

$$Score_{new} = Score_{old} \pm \sum_{k=1}^{dims} \frac{|D_i^{(k)}(d_i, r) - D_i^{(k)}(d_i, s_{dr})|}{f_k} \quad (6.10)$$

Jeder Datensatz startet mit einem initialen Score von Null. Der Score wird für jede Feature-Map  $k$  entsprechend angepasst.  $D_i^{(k)}(d_i, r)$  ist die Distanz der Feature-Map  $k$  des Datensatzes  $d_i$ , dessen Importance-Score berechnet werden soll, zur Feature-Map  $k$  des Real-Datensatzes  $r$ .  $D_i^{(k)}(d_i, s_{dr})$  ist die Distanz zur Feature-Map  $k$  vom DR-Basis-Datensatz  $s_{dr}$ . Die Differenz der beiden wird als Betrag berechnet, sodass sie immer positiv ist und anschließend wird das Ergebnis durch den Feature-Importance-Rank des Features  $k$  geteilt, als Gewichtungsfaktor. Bei einer Belohnung wird das Ergebnis auf den aktuellen Score aufaddiert und bei einer Bestrafung davon abgezogen. Am Ende ergibt sich so für jeden Datensatz ein Importance-Score, der als Bewertungskriterium für das Finale Ranking verwendet werden kann. Datensätze, die in wichtigen Features dem Real-Datensatz ähnlicher sind, werden dadurch stärker belohnt als unwichtigere und die, die dem DR-Datensatz in wichtigen Features ähnlicher sind werden stärker bestraft als die unwichtigen.

### 6.6.7 Visualisierung mit MDS

Um die Abstände zwischen den 17 Datensatz-Repräsentanten und den einzelnen Feature-Maps darzustellen, wird Multidimensional Scaling (MDS) (2.1.6.3) verwendet. Dadurch wird die aufgestellte Distanzmatrix in einen Raum mit weniger Dimensionen überführt, um eine übersichtliche Darstellung der Distanzbeziehungen zu bekommen. Unterschiede und Ähnlichkeiten der Datensätzen oder Feature-Maps können damit leichter erkannt werden. Jeder Punkt im resultierenden Koordinatensystem, repräsentiert einen Datensatz-Repräsentant auf Feature-Vektor-Level oder auch auf Feature-Map-Level, dann spezifisch für die Distanzen einer Feature-Map  $k$ .

---

## 6.6.8 Clusteranalyse mit UMAP

Parallel zum distanzbasierten Auswahlverfahren wird eine Clusteranalyse mit Uniform Manifold Approximation and Projection (UMAP) (2.1.6.2) auf den Datensatz-Repräsentanten durchgeführt. Ziel ist damit zu überprüfen, ob die Domänen erfasst werden können, wie sie mit den Distanzen bestimmt werden. Die Ergebnisse dessen sind im Kapitel Auswertung der Clusteranalyse (7.3.9) zu finden.

Ein Clustering für alle Feature-Vektoren aller Datensätze durchzuführen, würde zu einem großen Rechenaufwand führen, denn bei 14.000 Feature-Vektoren für die 16 synthetischen Datensätze und 5.452 Feature-Vektoren des Real-Datensatz, gibt es insgesamt 229.452 Feature-Vektoren der Form (8,12,32). Zur Anwendung von UMAP werden diese auch in eindimensionale flache Vektoren umgewandelt, sodass die Clusteranalyse als Input einen Datenvektor der Form (229.452, 3.072) bekommt. Die Clusteranalyse mit den reduzierten Feature-Vektoren kann allerdings im Verhältnis zur Datenmenge gesehen werden, die beim Vergleich der Original-Daten entstehen würde. Bei ursprünglichen Bilddaten der Form (256,384,4) und einer flachen Darstellung von 393.216 Werten pro Bild, würde dies zu einem Input von (229.452, 393.216) führen. Eine Reduktion von 99,22%. Die Aggregation zu den Datensatz-Repräsentanten wäre eine weitere Reduktion von 99,99%.

## 6.6.9 SVM-Training

Auch wurde eine SVM auf den Feature-Vektoren der Basis-Datensatz-Kodierungen trainiert. Ähnlich wie beim R-CAE-Modells bekommt die SVM jeweils 3.500 Feature-Vektoren eines Basis-Datensatzes als Trainingsdaten. Da die SVM ein überwachtes Lernverfahren ist, benötigt sie gelabelte Daten. Die Feature-Vektoren der Basis-Datensätze können einfach gelabelt werden, da sie bereits als Repräsentanten ihrer jeweiligen Domänen bekannt sind. Konkret bedeutet das, dass alle Feature-Vektoren des Basis-Datensatzes folgendermaßen gelabelt werden:

- „adidas\_syn\_dr“ mit „DR“
- „adidas\_syn\_simple“ mit „Simple“
- „adidas\_syn\_se“ mit „SE“
- „adidas\_real“ mit „Real“

Die SVM wird darauf trainiert, Muster in den Daten zu erkennen, die den unterschiedlichen Domänen entsprechen, um so auch die Domänen-Zugehörigkeit ungewisser Feature-Vektoren vorhersagen zu können. Die Vorhersagen sind im Kapitel Auswertung der SVM-Ergebnisse (7.3.8) dargestellt.



---

## 7 Ergebnisse und Evaluierung

In diesem Kapitel werden die Ergebnisse des Modell-Trainings, der Kodierung, der Aggregation und des Auswahlverfahrens beschrieben und ausgewertet.

### 7.1 Evaluierung der Modell-Trainings

Zur Evaluierung der trainierten Modelle wurden die Evaluierungsmethoden der CAE-Modelle (4.3) verwendet. Die Qualität der Rekonstruktionen wurde einerseits visuell durch den Vergleich von Originalbildern mit ihren Rekonstruktionen beurteilt. Andererseits kamen die quantitativen Metriken SSI und MSE zum Einsatz. Diese wurden auf jeweils 300 Testdaten aus acht verschiedenen Datensätzen ausgewertet. Ein Teil dieser Testdaten stammt aus den gleichen Datensätzen, aus denen Trainingsdaten für die Modelltrainings der Vor- und Hauptstudien entnommen wurden.

#### 7.1.1 Evaluierung der Vorstudie

Das Hauptziel der Vorstudie lag darin, die ideale Größe des Latenten Raums zu ermitteln. Gesucht wurde eine Größe, die einerseits eine starke Kompression der Daten ermöglicht und andererseits noch eine akzeptable Qualität bei der Rekonstruktion der Daten gewährleistet. Konkret ging es darum, herauszufinden, ab welcher Größe der Latente Raum zu klein wird. Ein zu kleiner latenter Raum könnte die strukturellen Merkmale der Daten aus verschiedenen Domänen nicht mehr ausreichend erfassen, was eine effektive Kodierung erschwert. Dafür wurden sieben sequenziell trainierten Modelle mit ihren Zielgrößen – 12.288 – 6.144 – 3.072 – 1.536 – 768 – 384 – 192 – aus Kapitel Konfigurationen der Vorstudie (6.3.5) verglichen.

##### 7.1.1.1 Visuelle Rekonstruktionen

Zur Veranschaulichung der Rekonstruktionsqualität wird ein Bild aus „adidas\_syn\_dr“ genommen, dargestellt in Abbildung 7.1. Dasselbe Bild wurde von den sieben Modellen rekonstruiert mit der Eingabegröße (256,384,4) und anschließend in die entsprechenden Kanäle aufgeteilt.

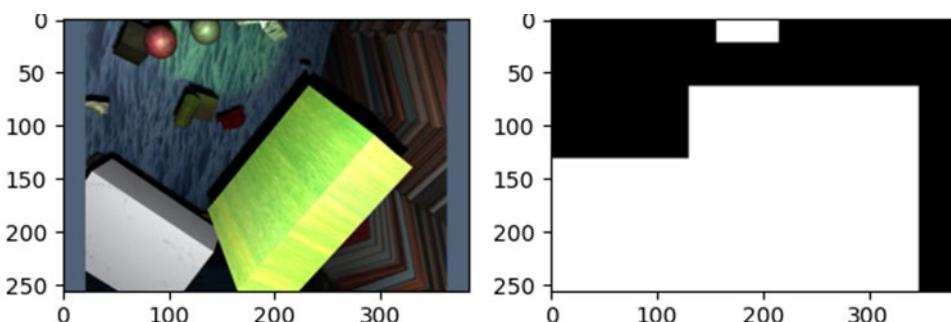
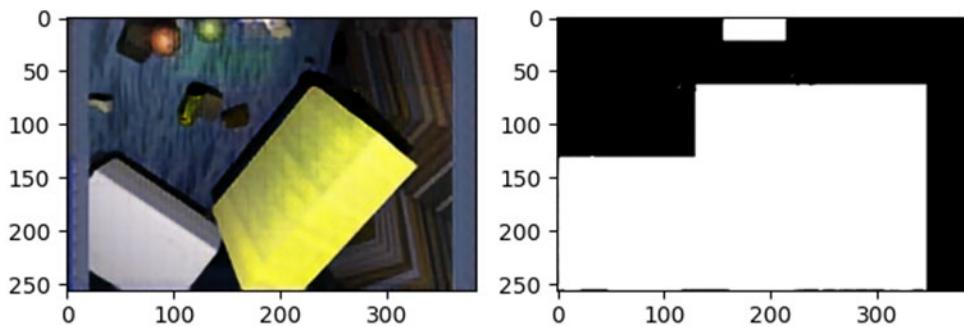
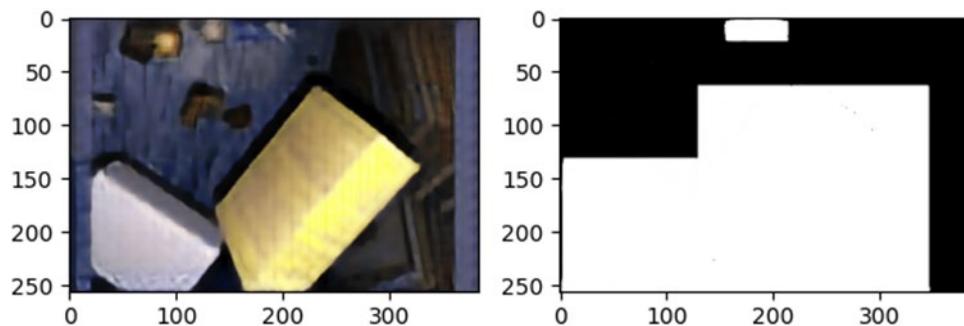


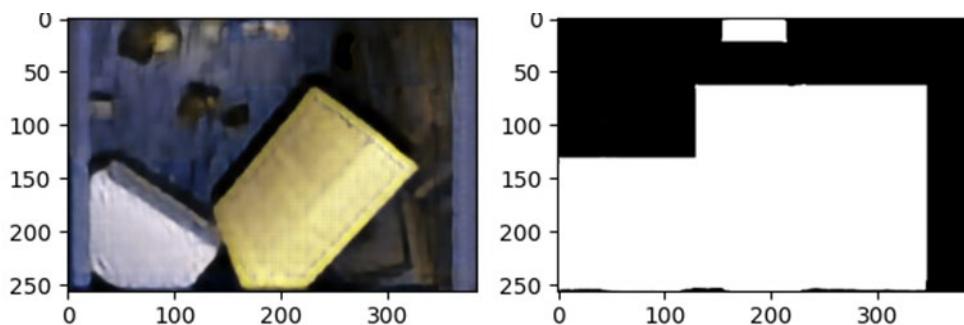
Abbildung 7.1: Original-Bild (links) und -Maske (rechts) als Sample von Datensatz "adidas\_syn\_dr" mit Original-Dimension von  $(256,384,4) = 393.216$



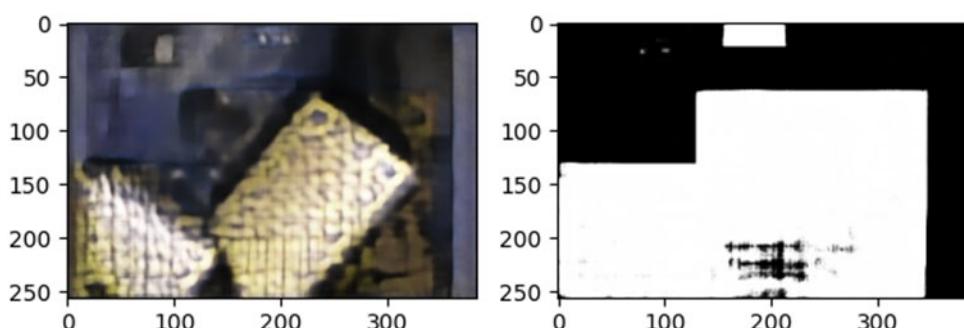
*Abbildung 7.2: Rekonstruktion durch Modell "r\_cae\_fm16x24\_dim12288"  
mit Latenter Dimension (16,24,32) = 12.288 und Reduktion von 96,88%*



*Abbildung 7.3: Rekonstruktion durch Modell "r\_cae\_fm8x12\_dim6144"  
mit Latenter Dimension (8,12,64) = 6.144 und Reduktion von 98,44%*



*Abbildung 7.4: Rekonstruktion durch Modell "r\_cae\_fm8x12\_dim3072"  
mit Latenter Dimension (8,12,32) = 3.072 und Reduktion von 99,22%*



*Abbildung 7.5: Rekonstruktion durch Modell "r\_cae\_fm4x6\_dim1536"  
mit Latenter Dimension (4,6,64) = 1.536 und Reduktion von 99,61%*

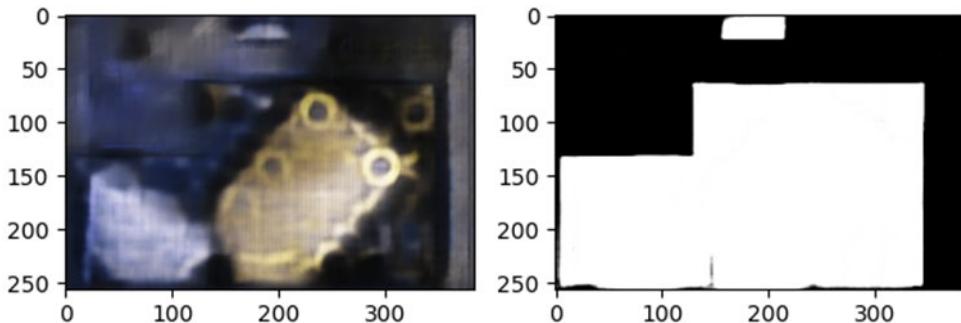


Abbildung 7.6: Rekonstruktion durch Modell "r\_cae\_fm4x6\_dim768" mit Latenter Dimension (4,6,32) = 768 und Reduktion von 99,80%

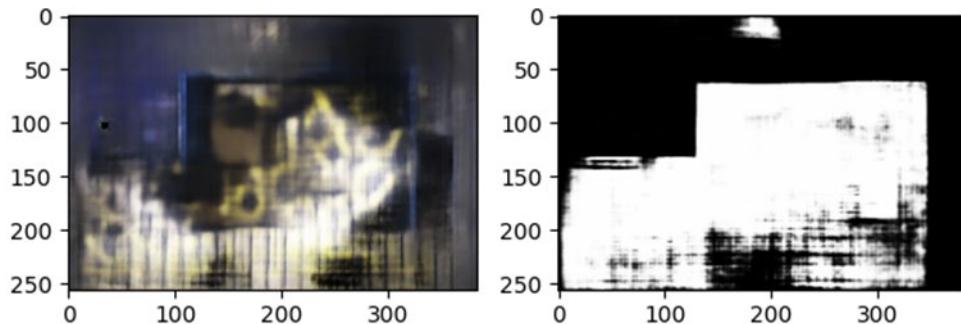


Abbildung 7.7: Rekonstruktion durch Modell "r\_cae\_fm2x3\_dim384" mit Latenter Dimension (2,3,64) = 384 und Reduktion von 99,90%

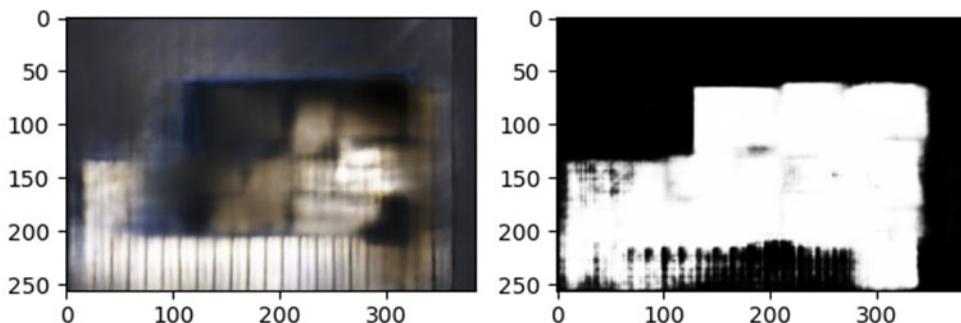


Abbildung 7.8: Rekonstruktion durch Modell "r\_cae\_fm2x3\_dim192" mit Latenter Dimension (2,3,32) = 192 und Reduktion von 99,95%

Der visuelle Vergleich der Rekonstruktionen verschiedener Modellgrößen zeigt erwartungsgemäß, dass mit der Verkleinerung des Latenten Raums die Rekonstruktionsqualität abnimmt. Insbesondere bei kleineren Dimensionen sind Anzeichen von Dataset-Imbalance erkennbar. Die Modelle neigen dazu, Szenen zu reproduzieren, die in den Trainingsdaten überrepräsentiert sind, insbesondere solche aus Domänen außerhalb der DR-Domäne. Trotz erkennbarer Artefakte bleiben bestimmte Elemente wie die Umrisse einer blauen Box, Schuhkartons in dieser Box und ein einheitlicher Hintergrund sichtbar, wie in Abbildung 7.8 zu sehen. Das liegt wahrscheinlich daran, dass das Modell auf Daten von sechs Datensätzen trainiert wurde, wobei nur einer davon aus der DR-Domäne stammt. Die restlichen fünf stammen aus dem Real-Datensatz, wobei jeweils zwei Datensätze aus den SE- und Simple-Domänen stammen. Obwohl sich diese untereinander im direkten Vergleich unterscheiden, haben sie eine Gemeinsamkeit im Vergleich zu den DR-Datensätzen: In allen Bildern ist die größte Variabilität zentriert, da sie im Wesentlichen Schuhboxen innerhalb einer großen mittig positionierten Box zeigen. Auch die minimalistisch abstrakten Simple-Datensätze zeigen diese Struktur, wenn auch mit

---

einheitlichem schwarzem Hintergrund und einfachen Farben. Daher lernten die Modelle, diese zentralisierte Szene und Positionierung zu priorisieren, was die Rekonstruktion von Objekten außerhalb des Bildzentrums erschwerte, besonders bei den kleinsten Zielgrößen von 192 und 384. Bei der Dimension 768 beginnt das Modell, die Struktur und Position der Boxen besser zu erfassen, allerdings mit Anzeichen von Overfitting, wie dem Versuch, das Adidas-Logo zu rekonstruieren, obwohl es im Original nicht vorhanden war. Das deutet darauf hin, dass das Modell möglicherweise die Darstellungen des Real-Datensatzes, des SE-Datensatzes und der Simple-Variante "adidas\_syn\_simple\_real\_texture", priorisiert hat, da diese die realen Texturen mit Logo enthalten. Interessanterweise erkennt das Modell mit Zielgröße 768 zumindest, dass das Logo, um den Bereich der Boxen zu platzieren ist – also, dass die Textur zu den Boxen gehört. Bei der Größe von 1.536 ist deutlicher zu erkennen, dass das Logo genauer in den Bereich der Boxen platziert wird, jedoch sind immer noch starke Artefakte zu sehen. Ab der Zielgröße von 3.072 verbessert sich die Rekonstruktion allmählich. Die Domänenmerkmale werden klarer erkennbar, obwohl die Farbinformationen noch nicht perfekt erfasst sind und die Bildqualität minderwertig ist. Die Zielgröße von (8,12,32) – 3.072 ist immer noch eine starke Kompression von 99,22% zur Original-Dimension von (256,384,4) – 393.216. Eine schlechtere Bildqualität ist daher bei jeder Zielgröße zu erwarten, die kleiner als die Original-Dimension ist. Auch die Masken-Rekonstruktionen zeigen Verbesserungen, wobei bereits bei sehr kleinen Zielgrößen die grobe Form korrekt erfasst wird, was darauf hindeutet, dass die Modelle möglicherweise zunächst den vierten Maskenkanal isoliert betrachten. Da jedoch auch die Artefakte, einschließlich der großen Box, im Bereich der Maske nachgebildet wurden, lernt das Modell womöglich den Zusammenhang, dass die zu rekonstruierenden Boxen im Bereich der nachgebildeten Masken liegen. Jedoch werden bei den kleinen Dimensionsgrößen nicht genügend Merkmale beibehalten, um die Position für die DR-Rekonstruktion richtig zu erfassen. Die Modelle liefern auch für die Rekonstruktionen der Trainingsdaten eine geringe Qualität, was auf Underfitting hindeutet. Dies könnte darauf zurückzuführen sein, dass die Modelle zu restriktiv sind und der Latente Raum einfach zu wenig Kapazität hat, um alle Domänen-spezifischen Eigenschaften zu berücksichtigen. Aber das sequenzielle Training selbst kann dies noch begünstigen. Da auf dem DR-Datensatz „adidas\_syn\_dr\_real\_texture“ zuerst trainiert wurde und dann die Daten der restlichen fünf Datensätze, könnten die Domänen-Eigenschaften dieser DR-Variante vergessen worden sein, bzw. nicht priorisiert gelernt oder überschrieben werden. Vor allem da nur 2.500 Daten der DR-Domäne im Vergleich zu 5.000 der SE-, 5.000 der Simple- und 2.500 der Real-Domäne verwendet wurden. Daher könnte, um dem entgegenzuwirken der Anteil an DR-Daten erhöht bzw. proportional zu den anderen Domänen gewichtet werden oder die Reihenfolge des sequenziellen Trainings angepasst werden. Allerdings besteht die Vermutung, dass auch mit diesen Änderungen die Dimensionsgröße noch zu klein ist, da die Rekonstruktionen von Testbildern aus den Datensätzen, auf denen trainiert wurden, auch starke Artefakte, Verzerrungen und ähnliche Hintergründe trotz verschiedener Domänen-Zugehörigkeit aufweisen bis zur Zielgröße 1.536.

### 7.1.1.2 Metriken

Neben dem visuellen Vergleich wurden die Metriken SSI und MSE zur Evaluierung der Performanz auf acht Datensätzen verwendet. Von diesen acht wurden sechs für die Trainings der sieben Modelle der Vorstudie verwendet. „adidas\_syn\_simple\_random\_texture“ und „adidas\_syn\_dr“ waren nicht beteiligt. Die SSI-Scores der Vorstudie können in Tabelle 7.1 nachvollzogen werden.

	adidas_real_ssi	adidas_syn_se_ssi	adidas_syn_simple_ssi	adidas_syn_dr_ssi	adidas_syn_dr_real_texture_ssi	adidas_syn_se_random_texture_ssi	adidas_syn_simple_real_texture_ssi	adidas_syn_simple_random_texture_ssi	mean_ssi
r_cae_fm2x3_dim192	0,9090	0,7042	0,6288	0,5668	0,5899	0,6131	0,6155	0,5705	0,6497
r_cae_fm2x3_dim384	0,9257	0,7097	0,6295	0,5860	0,6117	0,6161	0,6163	0,5867	0,6602
r_cae_fm4x6_dim768	0,9285	0,7583	0,6877	0,6528	0,6708	0,6606	0,6754	0,6529	0,7109
r_cae_fm4x6_dim1536	0,9453	0,7904	0,7568	0,6803	0,7029	0,6803	0,7422	0,7133	0,7514
r_cae_fm8x12_dim3072	0,9561	0,8711	0,8817	0,7463	0,7669	0,7662	0,8722	0,8465	0,8384
r_cae_fm8x12_dim6144	0,9668	0,8983	0,8920	0,8036	0,8183	0,8092	0,8831	0,8716	0,8679
r_cae_fm16x24_dim12288	0,9733	0,9314	0,8863	0,8090	0,8163	0,8719	0,8834	0,8853	0,8821

Tabelle 7.1: SSI-Scores der sequenziell trainierten Modelle mit absteigender Dimensionsgröße der Vorstudie

Dabei ist zu sehen, dass Modelle mit größeren Feature-Maps und Zielgrößen wie 6.144 und 12.288 durchweg höhere SSI-Scores erzielen, was auf eine bessere Rekonstruktionsqualität hindeutet. Das ist zu erwarten, denn eine höhere Dimensionsgröße bedeutet ein geringerer Informationsverlust. Der Real-Datensatz zeigt insgesamt die höchsten SSI-Scores, was darauf hindeutet, dass die Modelle echte Daten besser rekonstruieren können als synthetische. Jedoch kann dies eine Konsequenz des sequenziellen Trainings sein. Da auf dem Real-Datensatz zuletzt trainiert wurde, könnten sich die Modelle am stärksten an die zuletzt gelernten Merkmale erinnern. Dieses Phänomen ist als „Catastrophic Forgetting“ bekannt und beschreibt die Tendenz von neuronalen Netzen, früher gelernte Informationen zu vergessen, wenn sie auf neuen Daten trainiert werden. Die Tatsache, dass auch andere Datensätze hohe SSI-Scores aufweisen, deutet darauf hin, dass die Modelle mit größerer Latenten Dimension einige der zuerst gelernten Merkmale behalten haben. Das zeigt, dass nicht alle Modelle vollständig dem Catastrophic Forgetting unterliegen, sondern nur die, die zu wenig Kapazität im Latenten Raum haben, um die Menge an früher gelernten Informationen zu bewahren und sie nicht vollständig mit neuen Domänen-Informationen zu überschreiben.

Unter den synthetischen Datensätzen scheint der SE-Basis-Datensatz tendenziell höhere SSI-Scores zu erreichen, was darauf hindeuten könnte, dass die in diesem Datensatz enthaltenen synthetischen Daten – als Realimitationen – gut von den Modellen gelernt und rekonstruiert werden, womöglich, aufgrund ihrer Real-Nähe. Die Tatsache, dass keine Trainingsdaten der Datensätze "adidas\_syn\_dr" und "adidas\_syn\_simple\_random\_texture" verwendet wurden, könnte erklären, warum die SSI-Scores für die Testdaten dieser Datensätze niedriger ausfallen als für die Testdaten der Datensätze, die im Training enthalten waren für die Modelle bis zur Größe 768. Die Modelle, die höhere SSI-Scores für die beiden Datensätze erzielen, obwohl sie nicht im Training enthalten waren, deuten auf eine stärkere Generalisierungsfähigkeit hin. Größere Modelle sind besser in der Lage, unbekannte Daten zu rekonstruieren. Der Mean Score bietet eine Übersicht über die Gesamtperformanz über alle getesteten Datensätze. Modelle mit Zielgrößen von 6.144 und 12.288 zeigen erwartungsgemäß im Durchschnitt die beste Leistung, auch wenn kein großer Unterschied zwischen ihren SSI-Scores besteht. Visuell betrachtet, hat natürlich die Rekonstruktion vom Modell mit Größe 12.288 eine bessere Bildqualität, aber beide sind nach dem SSI-Score dazu in der Lage die Struktur des Bildes gut nachzubilden, auch wenn das Modell mit 6.144 einen größeren Qualitätsverlust hat. Das Modell mit Zielgröße 3.072 zeigt eine visuell deutlich erkennbare Verschlechterung der Bildqualität im Vergleich zu Modellen mit größeren Dimensionen, behält aber die Struktur im Vergleich zu den kleineren Dimensionen besser bei.

	adidas_real_mse	adidas_syn_se_mse	adidas_syn_simple_mse	adidas_syn_dr_mse	adidas_syn_dr_real_texture_mse	adidas_syn_se_random_texture_mse	adidas_syn_simple_real_texture_mse	adidas_syn_simple_random_texture_mse	mean_mse
r_cae_fm2x3_dim192	132,63	586,12	878,33	1631,67	1404,83	1344,11	957,77	955,47	986,37
r_cae_fm2x3_dim384	93,07	527,13	799,38	1427,98	1143,37	1215,56	894,13	866,21	870,85
r_cae_fm4x6_dim768	57,76	317,62	342,28	705,80	630,77	764,91	382,48	360,13	445,22
r_cae_fm4x6_dim1536	39,01	246,21	224,96	486,17	388,27	698,69	270,94	294,12	331,05
r_cae_fm8x12_dim3072	20,59	115,37	124,86	237,70	220,56	376,11	141,85	131,02	171,01
r_cae_fm8x12_dim6144	13,29	86,34	119,92	204,39	187,33	321,75	129,17	111,16	146,67
r_cae_fm16x24_dim12288	14,26	49,28	486,58	181,19	215,31	191,23	486,95	350,33	246,89

Tabelle 7.2: MSE-Scores der sequenziell trainierten Modelle mit absteigender Dimensionsgröße der Vorstudie

Auch die MSE-Scores in Tabelle 7.2 zeigen, dass der Fehler größer wird mit der Abnahme der Dimensionsgröße, wobei das Modell mit Zielgröße 12.228 überraschenderweise die Simple-Datensätze nicht so gut rekonstruieren kann und einen überproportionalen Fehler für diese erzeugt im Vergleich zu den Modellen 6.144 zu 786. Da es nicht das Ziel ist die R-CAEs zur Rekonstruktion oder Generierung neuer Bilder zu verwenden, wozu oft VAEs eingesetzt werden, ist ein Verlust der Bildqualität jedoch nicht schlecht sofern die Struktur und die Domäneneigenschaften gut erfasst werden. Denn das Ziel ist

---

es komplette Datensätze zu kodieren, um eine möglichst kleine komprimierte Darstellung, mit den wichtigsten Merkmalen und Charakteristiken, für den Vergleich im Auswahlverfahren zu erhalten. Aus diesem Grund wurden im weiteren Verlauf Modelle der Zielgröße 3.072 und 6.144 weiter untersucht.

### 7.1.2 Evaluierung der Hauptstudie

In der Hauptstudie wurden hauptsächlich Modelle mit Dimensionsgrößen von 3.072 und 6.144 verwendet. Im Vergleich zur Vorstudie, wurde nur auf Daten der vier Basis-Datensätze trainiert. Die Gründe dafür waren: Die vier Basis-Datensätze repräsentieren ihre spezifische Domäne – Real – SE – Simple – DR -, wobei alle anderen Datensätze als Varianten dieser erstellt wurden. Durch das Training wird getestet, ob sie ausreichend Domänenvielfalt bieten, um auch Bilder der Varianten-Datensätze zu rekonstruieren, obwohl sie spezifische Eigenschaften aufweisen, wie verschiedene Lichtverhältnisse, Kameraposition, Texturvariationen, auf denen nicht explizit trainiert wurde. Ein weiterer Grund ist, dass diese vier als Referenzwerte im Auswahlverfahren dienen. Indem für die spätere Kodierung ein Modell gewählt wird, welches ausschließlich auf den Basis-Datensätzen trainiert wurde, ist der spätere Vergleich „fair“, da das Modell keine Daten der Varianten-Datensätze kennt. Des Weiteren werden in der Hauptstudie die Domänen auch gleichmäßig vertreten. Um die Anzahl an Trainingsdaten auszugleichen, wurde sie für die Hauptstudie pro Datensatz auf 3.500 erhöht, was insgesamt 14.000 Trainingsdaten gibt, im Vergleich zu den 15.000 der Vorstudie.

#### 7.1.2.1 Sequenziell: CAE vs. R-CAE

Die ersten Trainingskonfigurationen auf den Basis-Datensätzen dienten für einen direkten Vergleich zwischen den CAE und R-CAE Modellen, um den Einfluss der Residual-Connections zu untersuchen. Interessanterweise erzielte das R-CAE-Modell mit einer Zielgröße von 3.072 in allen Datensätzen durchweg höhere SSI-Scores als das CAE-Modell mit 6.144, wie in Tabelle 7.3 zu sehen.

	<b>adidas_real_ssi</b>	<b>adidas_syn_se_ssi</b>	<b>adidas_simple_ssi</b>	<b>adidas_syn_dr_ssi</b>	<b>adidas_syn_dr_real_texture_ssi</b>	<b>adidas_syn_se_random_texture_ssi</b>	<b>adidas_simple_real_texture_ssi</b>	<b>adidas_random_texture_ssi</b>	<b>mean_ssi</b>
<b>cae_on_base_seq_fm8x12_dim3072_real3500</b>	0,9049	0,7343	0,6651	0,6389	0,6517	0,6478	0,6528	0,6247	0,6900
<b>cae_on_base_seq_fm8x12_dim6144_real3500</b>	0,9256	0,7549	0,6749	0,6667	0,6893	0,6532	0,6603	0,6290	0,7067
<b>r_cae_on_base_seq_fm8x12_dim3072_real3500</b>	0,9494	0,8454	0,7705	0,7382	0,7572	0,7403	0,7605	0,7434	0,7881
<b>r_cae_on_base_seq_fm8x12_dim6144_real3500</b>	0,9667	0,8909	0,8681	0,7946	0,8115	0,7944	0,8588	0,8274	0,8516

Tabelle 7.3: SSI-Scores der sequenziell trainierten Modelle CAE und R-CAE der Zielgrößen 3.072 und 6.144

Das zeigt, dass die Architektur mit Residual-Connections eine bessere Leistung bietet, selbst bei Halbierung der Dimension. Diese Modelle wurden ebenfalls sequenziell trainiert, in der Reihenfolge - DR - Simple - SE - Real -. Wie in der Vorstudie, sind auch an den SSI-Scores Anzeichen von „Catastrophic Forgetting“ zu erkennen. In diesem Fall führte es dazu, dass die zuletzt trainierten Domänen (SE und Real) tendenziell besser gelernt wurden und Datensätze dieser Domänen dadurch höhere SSI-Scores aufweisen, während die Domäne DR, auf der als erstes trainiert wurde, niedrigere SSI-Scores zeigt, auch in dem Varianten-Datensatz „adidas\_syn\_dr\_real\_texture“. Die Modelle, die auf den vier Basis-Datensätzen trainiert wurden, zeigten im Vergleich zu denen der Vorstudie mit sechs Datensätzen leicht niedrigere SSI-Scores bei gleicher Dimensionsgröße. Das könnte vor allem daran liegen, dass die Modelle in der Vorstudie bei mehr Datensätzen (und 1.000 Daten mehr) eine größere Vielfalt an Merkmalen gelernt haben. Obwohl eine Verschlechterung erwartet wurde, ist sie nicht besonders signifikant, weshalb das Training auf den vier Basis-Datensätzen weiter untersucht wurde.

Auch an den MSE-Scores der Tabelle 7.4 ist zu sehen, dass das CAE-Modell mit Zielgröße 6.144 einen deutlich stärkeren Fehler aufweist als das R-CAE Modell mit halbierter Kapazität im Latenten Raum.

	adidas_real_mse	adidas_syn_se_mse	adidas_syn_simple_mse	adidas_syn_dr_mse	adidas_syn_dr_real_texture_mse	adidas_syn_se_random_texture_mse	adidas_syn_simple_real_texture_mse	adidas_syn_simple_random_texture_mse	mean_mse
cae_on_base_seq_fm8x12_dim3072_real3500	138,22	463,12	473,41	970,16	853,23	1028,00	543,85	503,73	621,71
cae_on_base_seq_fm8x12_dim6144_real3500	76,90	436,96	422,46	1018,50	761,11	1183,32	487,88	449,13	604,53
r_cae_on_base_seq_fm8x12_dim3072_real3500	25,73	147,11	245,17	325,18	292,24	466,53	262,97	172,24	242,15
r_cae_on_base_seq_fm8x12_dim6144_real3500	14,04	98,27	84,36	214,52	206,31	335,72	97,06	102,92	144,15

Tabelle 7.4: MSE-Scores der sequenziell trainierten Modelle CAE und R-CAE der Zielgrößen 3.072 und 6.144

Wobei auch auffällt, dass die DR-Datensätze und „adidas\_syn\_simple\_real\_texture“ generell von allen Modellen den größten MSE zeigen, was vermutlich am sequenziellen Training liegt, bei dem auf den DR-Trainingsdaten zuerst trainiert wurde.

### 7.1.2.2 Sequenziell: Datensatz-Reihenfolge

Danach wurden Konfigurationen untersucht, bei denen die Reihenfolge der trainierten Daten variierte. Neben der vorigen Reihenfolge - DR - Simple - SE - Real - wurden die Reihenfolge - Real - SE - Simple - DR - sowie - Simple - DR - SE - Real - getestet. Die SSI-Scores sind in Tabelle 7.5 zu sehen.

	adidas_real_ssi	adidas_syn_se_ssi	adidas_syn_simple_ssi	adidas_syn_dr_ssi	adidas_syn_dr_real_texture_ssi	adidas_syn_se_random_texture_ssi	adidas_syn_simple_real_texture_ssi	adidas_syn_simple_random_texture_ssi	mean_ssi
r_cae_on_base_seq_fm8x12_dim3072_real3500_dr_si_se_re	0,9494	0,8454	0,7705	0,7382	0,7572	0,7403	0,7605	0,7434	0,7881
r_cae_on_base_seq_fm8x12_dim3072_real3500_si_dr_se_re	0,9600	0,8725	0,7948	0,7739	0,7897	0,7726	0,7814	0,7781	0,8154
r_cae_on_base_seq_fm8x12_dim3072_real3500_re_se_si_dr	0,8612	0,8543	0,9171	0,8163	0,8346	0,7834	0,9063	0,8872	0,8575
r_cae_on_base_seq_fm8x12_dim6144_real3500_dr_si_se_re	0,9667	0,8909	0,8681	0,7946	0,8115	0,7944	0,8588	0,8274	0,8516
r_cae_on_base_seq_fm8x12_dim6144_real3500_si_dr_se_re	0,9513	0,8861	0,8461	0,7823	0,7955	0,7925	0,8374	0,8252	0,8395
r_cae_on_base_seq_fm8x12_dim6144_real3500_re_se_si_dr	0,8929	0,8996	0,9175	0,8554	0,8715	0,8399	0,9082	0,8916	0,8846

Tabelle 7.5: SSI-Scores der sequenziell trainierten Modelle mit unterschiedlicher Trainings-Reihenfolge

Auch zu sehen ist, dass die Modelle mit größerer Dimension 6.144 mit Reihenfolge „\_dr\_si\_se\_re“ und „\_si\_dr\_se\_re“ durchweg besser abschneiden als die mit Zielgröße 3.072, derselben Reihenfolge. Allerdings hat das Modell mit Zielgröße 3.072 und Reihenfolge „\_re\_se\_si\_dr“ einen leicht besseren Mean-SSI. Es ist auffällig, dass bei gleicher Zielgröße, die Rekonstruktionsqualität der Testdaten des erst-trainierten Datensatzes unterschiedlich stark vom Catastrophic Forgetting betroffen ist:

- „r\_cae\_on\_base\_seq\_fm8x12\_dim3072\_real3500\_dr\_si\_se\_re“ erst auf „adidas\_syn\_dr“
- „r\_cae\_on\_base\_seq\_fm8x12\_dim3072\_real3500\_si\_dr\_se\_re“ erst auf „adidas\_syn\_simple“
- „r\_cae\_on\_base\_seq\_fm8x12\_dim3072\_real3500\_re\_se\_si\_dr“ erst auf „adidas\_real“.

Datensatz „adidas\_syn\_dr“ mit einem SSI von 0,7382 ist am meisten betroffen, gefolgt von „adidas\_syn\_simple“ mit 0,7948 und „adidas\_real“ mit 0,8612 (jeweils ausgehend vom Modell des erst-trainierten Datensatzes.)

	<b>adidas_real_mse</b>	<b>adidas_syn_se_mse</b>	<b>adidas_syn_simple_mse</b>	<b>adidas_syn_dr_mse</b>	<b>adidas_real_texture_mse</b>	<b>adidas_random_texture_mse</b>	<b>adidas_simple_texture_mse</b>	<b>adidas_random_texture_mse</b>	<b>mean_score</b>
<b>r_cae_on_base_seq_fm8x12_dim3072_real3500_dr_si_se_re</b>	25,73	147,11	245,17	325,18	292,24	466,53	262,97	172,24	242,15
<b>r_cae_on_base_seq_fm8x12_dim3072_real3500_si_dr_se_re</b>	17,72	114,28	229,26	268,45	260,94	395,00	254,77	202,73	217,89
<b>r_cae_on_base_seq_fm8x12_dim3072_real3500_re_se_si_dr</b>	170,82	130,04	64,47	140,54	114,43	303,16	90,41	89,46	137,92
<b>r_cae_on_base_seq_fm8x12_dim6144_real3500_dr_si_se_re</b>	14,04	98,27	84,36	214,52	206,31	335,72	97,06	102,92	144,15
<b>r_cae_on_base_seq_fm8x12_dim6144_real3500_si_dr_se_re</b>	32,17	108,23	153,93	239,56	240,09	333,28	167,63	140,50	176,93
<b>r_cae_on_base_seq_fm8x12_dim6144_real3500_re_se_si_dr</b>	143,93	89,93	45,81	102,42	81,09	216,70	65,85	54,25	100,00

Tabelle 7.6: MSE-Scores der sequenziell trainierten Modelle mit unterschiedlicher Trainings-Reihenfolge

Die MSE-Scores beider Modelle mit Reihenfolge „re\_si\_si\_dr“ in der Tabelle 7.6 zeigen durchweg kleinere Fehler als die anderen Modelle, was darauf hinweist, dass trotz halbierter Dimensionsgröße das Modell mit dieser Reihenfolge besser die Domänen-Eigenschaften erhält als die anderen. Die zuerst trainierten Real-Trainingsdaten scheinen zwar den größten MSE zu haben, jedoch deutlich kleinere für die restlichen Testdaten. Auch bei visueller Betrachtung fällt auf, dass die Dekodierung des DR- und Simple-Samples am schlechtesten ist, wie in Abbildung 7.9. Dies liegt daran, dass sie von den Modellen dekodiert wurden, die zuerst mit den Trainingsdaten desselben Datensatzes trainiert wurden. Da die Merkmale der Simple-Domäne tendenziell einfacher sind im Vergleich zu den komplexeren Merkmalen in den anderen Domänen, könnte das Modell zunächst abstraktere Merkmale lernen. Diese abstrakten Merkmale könnten dann durch die komplexeren Merkmale der nachfolgenden Datensätze überschrieben werden, was zu einer Verschlechterung der Rekonstruktion der Simple-Daten führt. Des Weiteren fällt auf, dass die Rekonstruktion des Real-Bildes durch das Modell mit Reihenfolge „\_si\_dr\_se\_re“ die beste Qualität der drei Real-Rekonstruktionen aufweist. Das könnte daran liegen, dass das Modell Simple eben als Ausgangspunkt nimmt, um weiterführende Merkmale und Komplexität zu erlernen, die in den nachfolgenden Datensätzen vorkommen. Nach dem dekodierten Real-Sample von Modell „\_si\_dr\_se\_re“ und dem SSI-Score von 0,8612 sind die Real-Daten zwar auch vom „Vergessen“ betroffen, wenn auf diesen als erstes trainiert wurde, aber nicht so stark wie der Simple- und DR-Datensatz. Bei einer größeren Dimension von 6.144 sind jedoch auch die SSI-Scores und Rekonstruktionen besser, da wieder mehr Kapazität im Latenten Raum zur Verfügung steht, die die Domäneneigenschaften des erst-trainierten Datensatz behalten kann.

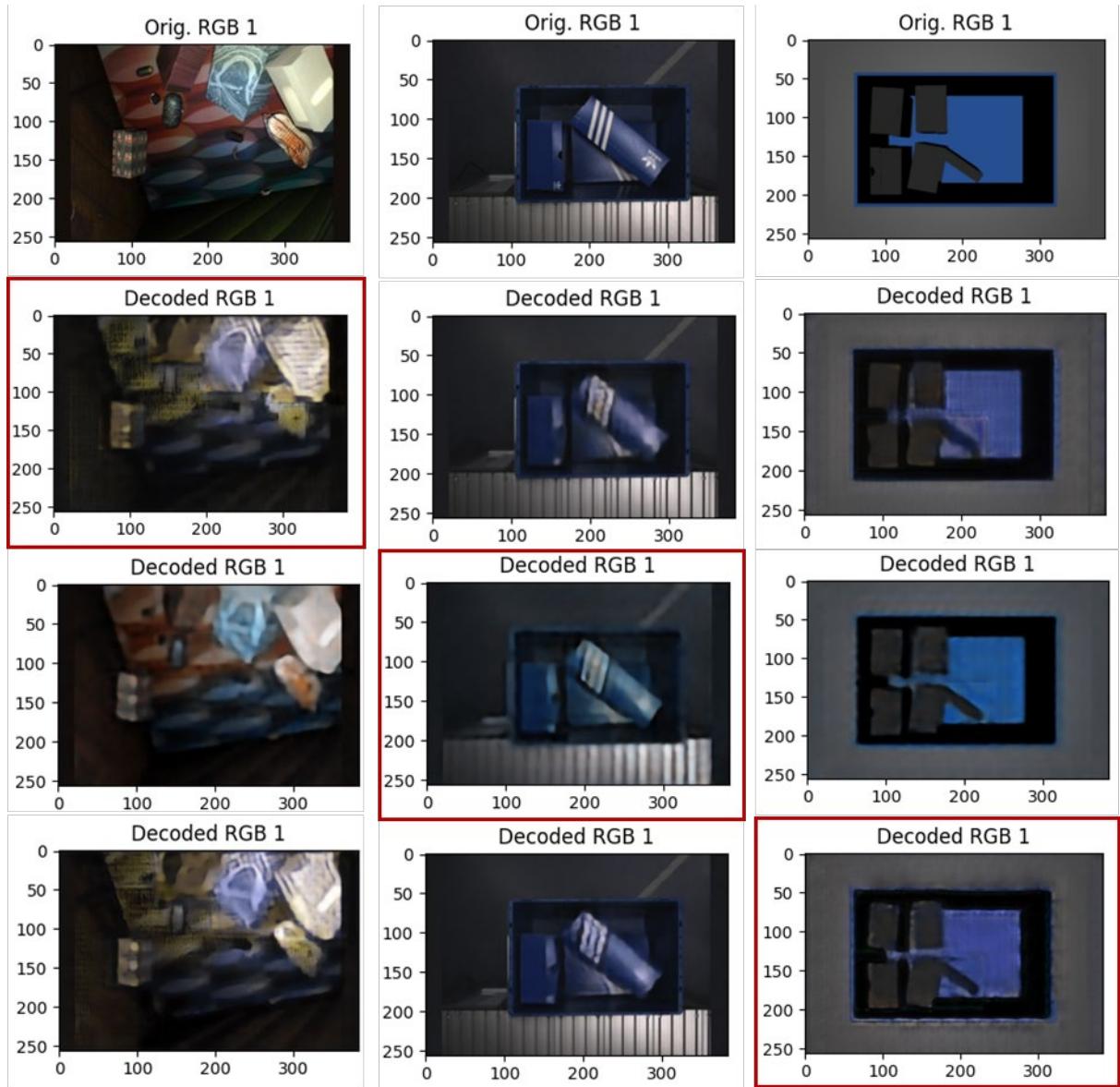


Abbildung 7.9: Die Grafiken zeigen die Rekonstruktionen von jeweils einem DR-, Simple- und Real-Sample, zur Veranschaulichung, wie diese von der Trainings-Reihenfolge betroffen sind. Original-Sample, darunter gefolgt von Dekodierungen durch „dr\_si\_se\_re“, „re\_se\_si\_dr“, „si\_dr\_se\_re“. Die rot hervorgehobenen Bilder, stellen die Samples dar, die zum Datensatz gehören, der als erstes im Modell-Training verwendet wurde.

### 7.1.2.3 Kombiniert: R-CAE mit kleinen Zielgrößen

Um dem „Catastrophic Forgetting“ entgegenzuwirken, wurden Modelle auf „kombinierten“ Daten trainiert. Das heißt, in allen Batches (der Größe vier) kommt jeweils ein Datenpunkt der vier Basis-Datensätze vor, wodurch das Modell nicht nach und nach auf die neusten Daten konditioniert wird, sondern regelmäßig alle sieht. Im Gegensatz zu den sequenziellen Modellen, zeigen die SSI-Scores der kombiniert-trainierten Modellen in unterer Tabelle 7.7 höhere Werte über alle Datensätze hinweg. Der Unterschied zwischen den Zielgrößen ist nach der SSI-Metrik nicht sehr signifikant.

	adidas_real_ssi	adidas_syn_se_ssi	adidas_syn_simple_ssi	adidas_syn_dr_ssi	adidas_syn_dr_texture_ssi	adidas_syn_random_texture_ssi	adidas_syn_simple_texture_ssi	adidas_syn_random_texture_ssi	mean_ssi
r_cae_on_base_comb_fm8x12_dim6144_real3500	0,9477	0,9446	0,9523	0,8374	0,8515	0,8587	0,9448	0,8787	0,9019
r_cae_on_base_comb_fm8x12_dim3072_real3500	0,9370	0,9331	0,9381	0,8044	0,8203	0,8309	0,9287	0,8612	0,8817
r_cae_on_base_comb_fm4x6_dim1536_real3500	0,9313	0,9158	0,9719	0,7594	0,7771	0,7798	0,9577	0,8666	0,8699
r_cae_on_base_comb_fm4x6_dim768_real3500	0,9093	0,8614	0,9493	0,7356	0,7523	0,7317	0,9346	0,8394	0,8392
r_cae_on_base_comb_fm2x3_dim384_real3500	0,9100	0,8340	0,9465	0,6920	0,7091	0,6955	0,9299	0,8328	0,8187
r_cae_on_base_comb_fm2x3_dim192_real3500	0,8895	0,8159	0,9292	0,6785	0,6958	0,6855	0,9119	0,8017	0,8010

Tabelle 7.7: SSI-Scores der kombiniert trainierten Modelle mit absteigender Zielgröße von 6.144 bis 192

Abbildung 7.10 betrachtend, ist jedoch klar, dass das Modell mit Zielgröße von 6.144 aufgrund der größeren Kapazität im Latenten Raum mehr Informationen über das Eingabebild kodieren kann. Das bedeutet, dass neben strukturellen Informationen wie die Objektpositionen, andere Aspekte wie Textur und Farbe „mehr Platz“ einnehmen können, wodurch die Bildqualität besser wird. Das kleinere Modell mit Zielgröße 3.072 hat eine geringere Kapazität, um feinere Details zu kodieren. Dadurch müssen Kompromisse eingegangen werden, bei denen die strukturellen Informationen über genauere Texturinformationen gestellt werden.

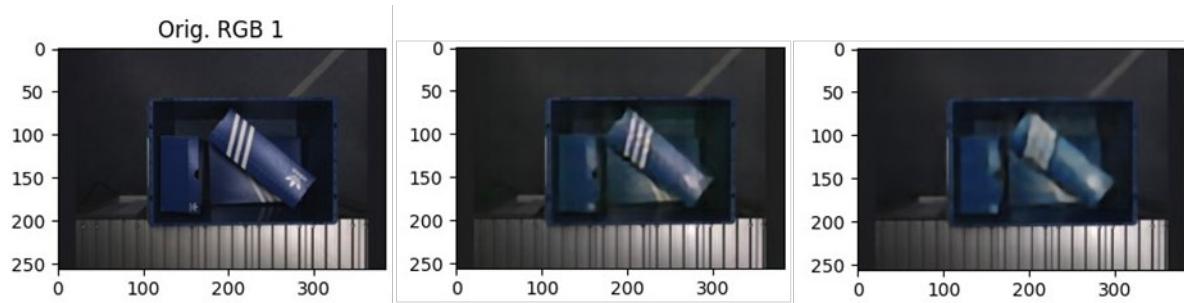


Abbildung 7.10: Vergleich der Rekonstruktionen eines Real-Samples (links) mit Modellen:  
*r\_cae\_on\_base\_comb\_fm8x12\_dim6144\_real3500* (mittig) &  
*r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500* (rechts)

Da sich der SSI-Score auf die grundlegenden strukturellen Ähnlichkeiten konzentriert, wie die Anordnung und Ausrichtung von Objekten, Kanten und Texturen und diese Aspekte ähnlich gut von beiden Modellen erfasst werden, sind die SSI-Scores trotz unterschiedlicher Bildqualität hoch.

Die Tabelle 7.8 zeigt, dass die Modelle ab Zielgröße 1.536 einen durchweg größeren MSE haben, wobei die beiden DR-Datensätze und „adidas\_syn\_se\_random\_texture“ die größten Fehler unabhängig der Zielgröße zeigen.

	adidas_real_mse	adidas_syn_se_mse	adidas_syn_simple_mse	adidas_syn_dr_mse	adidas_syn_dr_real_texture_mse	adidas_syn_se_random_texture_mse	adidas_syn_simple_real_texture_mse	adidas_syn_simple_random_texture_mse	mean_score
r_cae_on_base_comb_fm8x12_dim6144_real3500	29,76	39,97	17,06	99,65	90,98	181,12	27,59	56,47	67,83
r_cae_on_base_comb_fm8x12_dim3072_real3500	37,60	50,64	19,96	140,82	120,55	253,18	34,85	80,07	92,21
r_cae_on_base_comb_fm4x6_dim1536_real3500	53,79	81,21	26,19	213,33	184,95	438,97	55,09	169,44	152,87
r_cae_on_base_comb_fm4x6_dim768_real3500	83,88	144,80	46,56	278,65	242,40	552,29	86,08	254,71	211,17
r_cae_on_base_comb_fm2x3_dim384_real3500	150,42	262,56	105,03	594,81	522,62	968,33	182,68	489,34	409,47
r_cae_on_base_comb_fm2x3_dim192_real3500	172,51	306,36	126,87	616,87	548,52	984,94	202,21	429,76	423,50

Tabelle 7.8: MSE-Scores der kombiniert trainierten Modelle mit absteigender Zielgröße von 6.144 bis 192

#### 7.1.2.4 Kombiniert: Abnahme der Real-Trainingsdaten

Da Real-Daten wertvoll sind, wurde auch untersucht, ob der Real-Anteil im Training reduziert werden könnte bei gleich gutem SSI-Score für die Real-Testdaten. Die Tabelle 7.9 zeigt ausschließlich die MSE und SSI-Scores des Real-Datensatzes für die acht Modelle: Jeweils drei für die beiden Zielgrößen 3.072

	r_cae_on_base_comb_fm8x12_dim3072_real500	r_cae_on_base_comb_fm8x12_dim3072_real2500	r_cae_on_base_comb_fm8x12_dim3072_real1500	r_cae_on_base_comb_fm8x12_dim3072_real500	r_cae_on_base_comb_fm8x12_dim6144_real3500	r_cae_on_base_comb_fm8x12_dim6144_real2500	r_cae_on_base_comb_fm8x12_dim6144_real1500	r_cae_on_base_comb_fm8x12_dim6144_real500
SSI	0,9370	0,9350	0,9254	0,9122	0,9477	0,9509	0,8964	0,9434
MSE	37,60	37,64	54,35	67,66	29,76	25,04	82,87	42,76

Tabelle 7.9: SSI- und MSE-Scores der kombiniert trainierten Modelle mit abnehmendem Real-Anteil

und 6.144 mit abnehmenden Real-Anteil von – 2.500 – 1.500 – 500 – und den zwei Modellen des vorigen Kapitels mit Real-Anteil von 3.500.

Wie zu sehen, ist in den 3.072 Modellen eine kleine Abstufung der SSI-Scores mit abnehmenden Real-Anteil zu erkennen. Diese sind wieder sehr ähnlich, da ab Zielgröße 3.072 auf kombinierten Trainingsdaten, die strukturellen Informationen weitestgehend erhalten bleiben. Der Unterschied wird

---

an den MSE-Scores deutlicher. Zwischen dem Real-Anteil von 3.500 mit MSE von 37,60 und 2.500 Real-Daten mit MSE 37,64 besteht nur eine kleine Differenz von 0,04. Der Sprung von 2.500 auf 1.500 Real-Daten erzeugt schon einen größeren Durchschnitts-MSE von 54,35 für die Real-Testdaten und zu 500 eine weitere Verschlechterung zu 67,66. Demnach könnte der Real-Anteil zumindest bei Zielgröße 3.072 auf 2.500 reduziert werden ohne große Einbußen nach SSI oder MSE zu erzeugen. Die anderen Modelle mit Zielgröße 6.144 zeigen beim Vergleich zwischen einem Real-Anteil von 3.500 und 2.500 überraschenderweise, dass das Modell mit 2.500 Real-Daten einen leicht kleineren MSE aufweist. Demnach könnte die Wahl von 2.500 Real-Daten sogar zu einer Verbesserung bei dieser Zielgröße führen. Das Modell-Training von „r\_cae\_on\_base\_comb\_fm8x12\_dim6144\_real1500“ wurde durch einen Fehler frühzeitig nach sechs Epochen abgebrochen. Aus diesem Grund sind die Scores für dieses Modell nicht aussagekräftig und wurden nicht berücksichtigt. Der Sprung von 2.500 Real-Daten zu 500 ist, wie bei der Zielgröße von 3.072, stark am MSE zu sehen, der von 25,04 auf 42,76 steigt. Demnach zeigen die Ergebnisse entgegen der Annahme, dass die Real-Domäne gleichermaßen vertreten werden sollte, dass zumindest für die beiden Zielgrößen der Real-Anteil auf 2.500 reduziert werden könnte.

### 7.1.2.5 Kombiniert: Filteranzahl-Strategie

Die nächsten zwei Modelle wurden mit unterschiedlichen Filteranzahlen erstellt, für die Zielgröße von 1.536, um zu testen welchen Einfluss die Strategie der Filterwahl in den Encoding- und Decoding-Schritten hat. Dabei hat das Modell „r\_cae\_on\_base\_comb\_fm4x6\_dim1536\_real3500\_up“ eine Latente Dimension von (4,6,64), durch Filterkonfiguration [16,16,32,32,64,64] und das Modell „r\_cae\_on\_base\_comb\_fm8x12\_dim1536\_real3500\_down“ eine Latente Dimension von (8,12,16), durch Filterkonfiguration [64,64,32,32,16].

In Tabelle 7.10 und Tabelle 7.11 wird deutlich, dass das Modell mit einer von Anfang an hohen, aber dann abnehmenden Anzahl an Filtern besser abschneidet als das Modell, bei dem die Anzahl der Filter schrittweise zunimmt. Dieses Modell erreicht höhere SSI- und niedrigere MSE-Werte über alle Testdaten hinweg. Das könnte daran liegen, dass die Strategie der absteigenden Filteranzahl die Eingabedaten früh und gezielt komprimiert. Das bedeutet, es werden unnötige oder überflüssige Informationen schnell aussortiert. Im Gegensatz dazu beginnt das Modell mit aufsteigender Filteranzahl mit einer geringeren Anzahl von Filtern, was bedeutet, dass es anfangs weniger Merkmale gleichzeitig erfassen kann. Mit zunehmender Tiefe und damit steigender Filteranzahl kann das Modell zwar mehr und spezifischere Informationen erfassen, aber dieser schrittweise Aufbau führt möglicherweise zu einer weniger effizienten Kompression, weil kritische Details erst erfasst werden, wenn bereits eine erhebliche Menge an Datenverarbeitung stattgefunden hat. Demnach könnte es entgegen dem CNN-Prinzip (bei dem die Filteranzahl stetig erhöht wird) vom Vorteil sein, diese bei CAEs zu verringern, sofern der Anwendungsfall eine Dimensionsreduktion wie in diesem Projekt ist.

	adidas_real_ssi	adidas_syn_se_ssi	adidas_syn_simple_ssi	adidas_syn_dr_ssi	adidas_syn_dr_real_texture_ssi	adidas_syn_se_random_texture_ssi	adidas_syn_simple_real_texture_ssi	adidas_syn_simple_random_texture_ssi	mean_ssi
r_cae_on_base_comb_fm4x6_dim1536_real3500_up	0,9111	0,8868	0,9372	0,7437	0,7621	0,7589	0,9233	0,8324	0,8444
r_cae_on_base_comb_fm8x12_dim1536_real3500_down	0,9340	0,9298	0,9678	0,7821	0,7989	0,8070	0,9560	0,8712	0,8809

Tabelle 7.10: SSI-Scores der Modelle mit abnehmender und wachsender Filteranzahl

	adidas_real_mse	adidas_syn_se_mse	adidas_syn_simple_mse	adidas_syn_dr_mse	adidas_syn_dr_real_texture_mse	adidas_syn_se_random_texture_mse	adidas_syn_simple_real_texture_mse	adidas_syn_simple_random_texture_mse	mean_mse
r_cae_on_base_comb_fm4x6_dim1536_real3500_up	74,34	115,14	42,82	245,29	215,07	513,10	78,07	209,73	186,70
r_cae_on_base_comb_fm8x12_dim1536_real3500_down	41,00	56,70	17,57	165,02	140,44	331,13	38,26	105,89	112,00

Tabelle 7.11: MSE-Scores der Modelle mit abnehmender und wachsender Filteranzahl

### 7.1.2.6 Bestimmung des optimalen Modells

Das optimale Modell sollte eine starke Dimensionsreduktion bieten und trotzdem die strukturellen Informationen beibehalten, da angenommen wird, dass dadurch ausreichend Domänen-Eigenschaften für den Vergleich im Auswahlverfahren erhalten bleiben. Nach der Vorstudie geht hervor, dass die sequenziell trainierten Modelle erst ab Zielgröße 3.072 genügend Rekonstruktionsqualität bieten. Der weitere Vergleich zwischen den verschiedenen Datensatz-Reihenfolgen im Training zeigt zwar, dass die Reihenfolge – RE – SE – Simple – DR – im Vergleich zu den anderen getesteten, bessere Ergebnisse erzielt, jedoch noch deutlich schlechtere als das kombinierte Training mit derselben Zielgröße. Nach der Untersuchung der Reduktion der Real-Trainingsdaten geht hervor, dass bei einer Zielgröße von 6.144 ein Real-Anteil von 2.500 sogar zu leicht besseren Ergebnissen führt, und bei Zielgröße 3.072 kein großer Unterschied besteht. Nach den Ergebnissen der Filterstrategie zeigt das Modell mit der Strategie „abnehmend“ eine bessere Leistung. Ausgehend davon, wurden im Auswahlverfahren alle 17 verfügbaren Datensätze mit dem Encoder von „r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500“, als kombiniert trainiertes Modell, dessen Trainingsdaten, jeder Domäne gleich vertreten waren, kodiert. Diese wurde dem Modell mit geringeren Real-Anteil vorgezogen, da das Auswahlverfahren auch auf die Datensatz-Kodierungen von Modell „r\_cae\_on\_base\_comb\_fm8x12\_dim6144\_real3500“ und von Modell „r\_cae\_on\_base\_comb\_fm8x12\_dim1536\_real3500“ durchgeführt wurde. Es soll verglichen werden, ob die Kodierungen größerer oder kleinerer Zielgröße ähnliche Ergebnisse erzielen. Die Ergebnisse dieser Rankings sind entsprechend im Anhang C und D hinterlegt.

---

## 7.2 Latenter Raum

Die folgenden Ergebnisse beziehen sich auf die Datensatz-Kodierungen von Modell: „r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500“. Nachdem alle Datensatz-Kodierungen der Form  $(n, 8, 12, 32)$  vorliegen, wurden sie aggregiert, indem der Durchschnitts-Wert jeder Pixelposition in jeder Feature-Map bestimmt wurde, um einen Durchschnitts-Feature-Vektor pro Datensatz als Datensatz-Repräsentant zu erzeugen. Im Anhang Durchschnitts-Feature-Vektoren (A.1) sind die Datensatz-Repräsentanten in jeweils 32 Feature-Maps der Größe  $8 \times 12$  dargestellt, als Ergebnis der Aggregation.

Bei Betrachtung der Durchschnitts-Feature-Vektoren der Datensätze jeweils gleicher Domäne ist visuell deutlich zu erkennen, dass die Simple-, SE- und DR-Datensätze sich jeweils einander ähneln. Die Simple-Datensätze zeigen im Vergleich zu den anderen Domänen, sehr glatte Kanten und klare Konturen, die die minimalistischen Strukturen der ursprünglichen Bilder erkennen lassen. Dabei zeigt die Variante „adidas\_syn\_simple\_real\_camera“ insbesondere in Feature-Map Ch21 und Ch22, eine größere Abweichung zu den Feature-Maps des Simple-Basis-Datensatzes, was darauf hinweist, dass diese Feature-Maps die Merkmale der „real\_camera“ erfassen. Die restlichen Simple-Datensätze scheinen sehr ähnlich zu sein. Auch die drei Datensätze der DR-Domäne scheinen fast identisch zu sein, wobei jedoch minimale Intensitäts-Unterschiede zu erkennen sind. Innerhalb der SE-Domäne sind die Datensätze variabler. Fast alle Feature-Maps von „adidas\_syn\_se\_random\_camera“ und die meisten von „adidas\_syn\_se\_no\_lc\_table“ scheinen eher Ähnlichkeiten mit der DR-Domäne aufzuweisen, da die Aktivierungen sehr variabel verteilt sind. Es ist daher zu erwarten, dass bei den Distanzmessungen dieser Datensatz-Repräsentanten eine deutlich erkennbare Nähe zur DR-Domäne auftritt. Insgesamt zeigen die Feature-Maps Ch11, Ch13, Ch17 und Ch28 über alle Datensatz-Kodierungen hinweg eher Aktivierungen an den Randpixeln auf, was darauf hindeuten könnte, dass diese Merkmale Hintergrund-Informationen wahrnehmen könnten.

Die Durchschnitts-Feature-Vektoren scheinen so zumindest visuell bereits Ähnlichkeiten aufzuweisen, wenn sie aus gleichen Domänen stammen, wobei sich die DR-Domäne im Vergleich zu den anderen drei noch deutlich abhebt. Betrachtet man die Durchschnitts-Feature-Vektoren verschiedener Datensätze gleicher Domäne, so sind sie in den meisten Feature-Maps zwar ähnlich, zeigen jedoch minimale Intensitäts-Unterschiede oder deutliche Abweichungen in bestimmten Feature-Maps bzw. in spezifischen Positionen, die eben auf die Merkmale oder Gruppeninformationen der Varianten hinweisen. Aus diesem Grund eignen sich die Durchschnitts-Feature-Vektoren gut als Repräsentanten für das Auswahlverfahren, da die Domänen-Zugehörigkeit und Ähnlichkeiten durch die Distanzmessungen deutlich erkennbar sein sollten.

Neben der Darstellung der Durchschnitts-Feature-Vektoren ist auch die Standardabweichung von Interesse, da in den Heat-Maps die Abweichungen in den Feature-Maps leicht erkennbar werden. Die

---

Std-Dev-Feature-Vektoren der vier Basis-Datensätze sind im Anhang Std-Dev-Feature-Vektoren (A.2) zu finden. Dabei ist deutlich erkennbar, dass der DR-Datensatz, wie auch schon in seinem Durchschnitts-Feature-Vektor zu sehen war, aufgrund der randomisierten Positionen, Hintergründe und Texturen im gesamten Bild, eine hohe Abweichung in allen Feature-Maps zeigt. Die anderen drei Basis-Datensätze sehen von weitem sehr ähnlich aus, da sie die Gemeinsamkeit haben, dass alle Bilder eine zentrierte Box zeigen und innerhalb dieser die meisten Abweichungen über alle Bilder auftreten.

## 7.3 Evaluierung des Auswahlverfahrens

In dem Kapitel werden die Ergebnisse des Auswahlverfahrens gezeigt, das auf den Datensätzen angewendet wurde, die durch das Modell "r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500" kodiert wurden und für die jeweils Durchschnitts-Feature-Vektoren als Datensatz-Repräsentanten bestimmt wurden. Die Datensatz-Repräsentanten wurden nach der Aggregation mit dem Robust-Scaler normalisiert und nach der Distanzmessung mit der Manhattan Distanz durch die Min-Max-Skalierung in den Wertebereich [0,1] gebracht.

### 7.3.1 Referenzwerte der Objekterkennungsaufgabe

Um das resultierende Ranking bewerten und beurteilen zu können, werden die Performance-Scores der Objekterkennungsaufgabe herangezogen. Die mAPs können im Anhang Referenzwerte der Objekterkennungsaufgabe (B.1) nachvollzogen werden. Wichtig für die Überprüfung des Rankings ist die ordinale Ordnung, die sich aus der absteigenden Objekterkennungs-Genauigkeit ergibt, da das zu erstellende Ranking auch ordinal sein wird:

1. adidas\_real
2. adidas\_syn\_se\_random\_light
3. adidas\_syn\_se
4. adidas\_syn\_se\_random\_camera
5. adidas\_syn\_se\_random\_texture
6. adidas\_syn\_simple\_real\_camera
7. adidas\_syn\_simple
8. adidas\_syn\_se\_no\_lc\_table
9. adidas\_syn\_dr\_real\_texture
10. adidas\_syn\_dr

### 7.3.2 Distanzmatrix auf Feature-Vektor-Level

Die Distanzmatrix in Abbildung 7.12, die alle Distanzen aller Datensatz-Repräsentanten zueinander zeigt, wurde mit MDS in Abbildung 7.11 visualisiert:

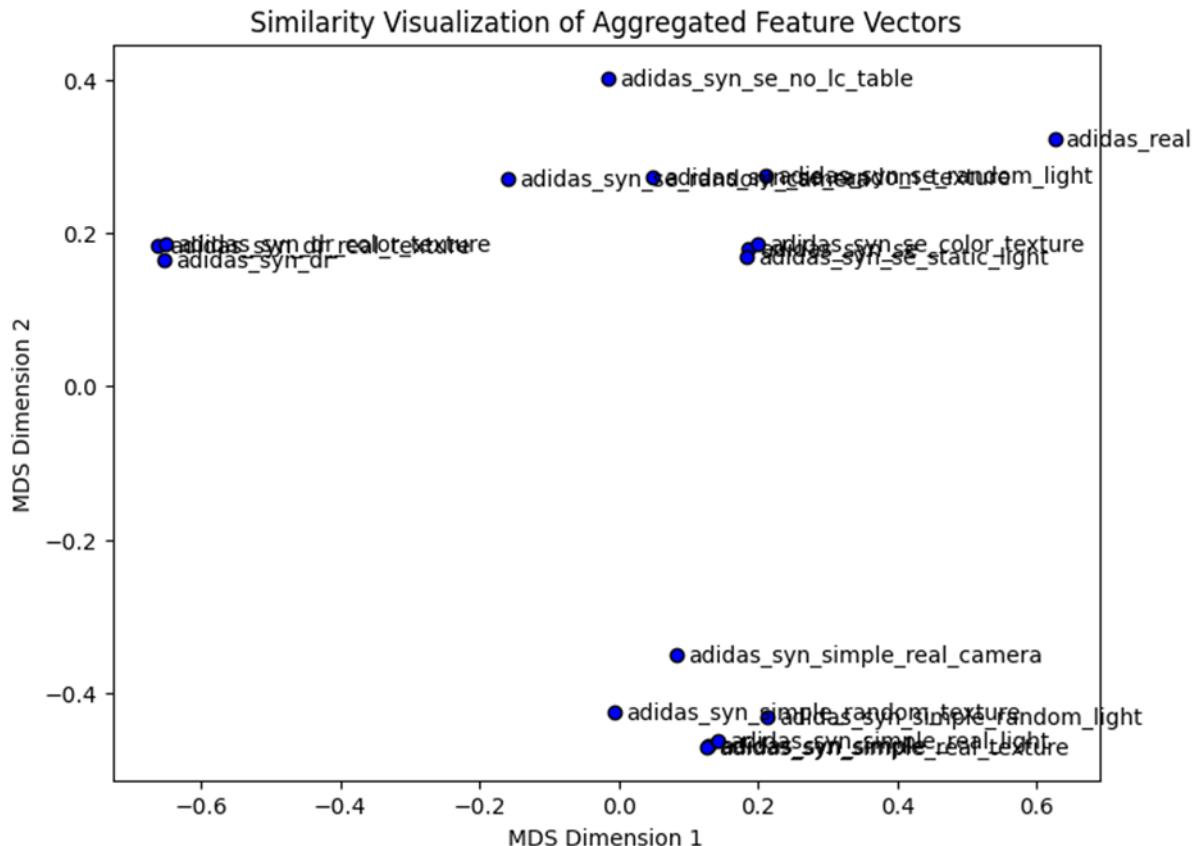


Abbildung 7.11: MDS-Visualisierung der Datensatz-Repräsentanten

Daran ist schon deutlich zu erkennen, dass Datensätze gleicher Domäne eine geringe Distanz zueinander aufweisen und Gruppen bilden, die ihren Domänen-Zugehörigkeiten entsprechen. Insbesondere die Datensätze der Domänen Simple und DR scheinen sich stark zu clustern. Hierbei ist jedoch nicht zu erkennen, dass sich innerhalb der Domänen deutliche Subgruppen bilden. Innerhalb der Simple-Domäne sind die Datensätze „adidas\_syn\_simple\_real\_light“ und „adidas\_syn\_simple\_real\_texture“ sehr nah an ihrem Simple-Basis-Datensatz und innerhalb der DR-Domäne überlappen sich die Datenpunkte von „adidas\_syn\_dr\_real\_texture“ und „adidas\_syn\_dr\_color\_texture“, wobei sie auch sehr nah zu ihrem DR-Basis-Datensatz sind. Die Datensätze der Domäne SE sind breiter gestreut. Auffällig ist, dass die Datensätze "adidas\_syn\_se\_random\_camera" und "adidas\_syn\_se\_no\_lc\_table" weiter vom SE-Basis-Datensatz entfernt sind und den DR-Datensätzen nahekommen. Das könnte darauf zurückzuführen sein, dass die erstgenannte SE-Variante durch die zufällige Kamerastellung, eher der DR-Domäne ähnelt, da die Bilder in verschiedenen Positionen und Winkeln aufgenommen wurden. Somit ist es charakteristisch für diese SE-Varianten, dass sie im Vergleich zu anderen Datensätzen der gleichen Domäne eine größere

Varianz aufweist, die nicht zentral liegt. Diese Beobachtung konnte bereits an den visuellen Darstellungen der Durchschnitts-Feature-Vektoren aus Latenter Raum (7.2) festgestellt werden.

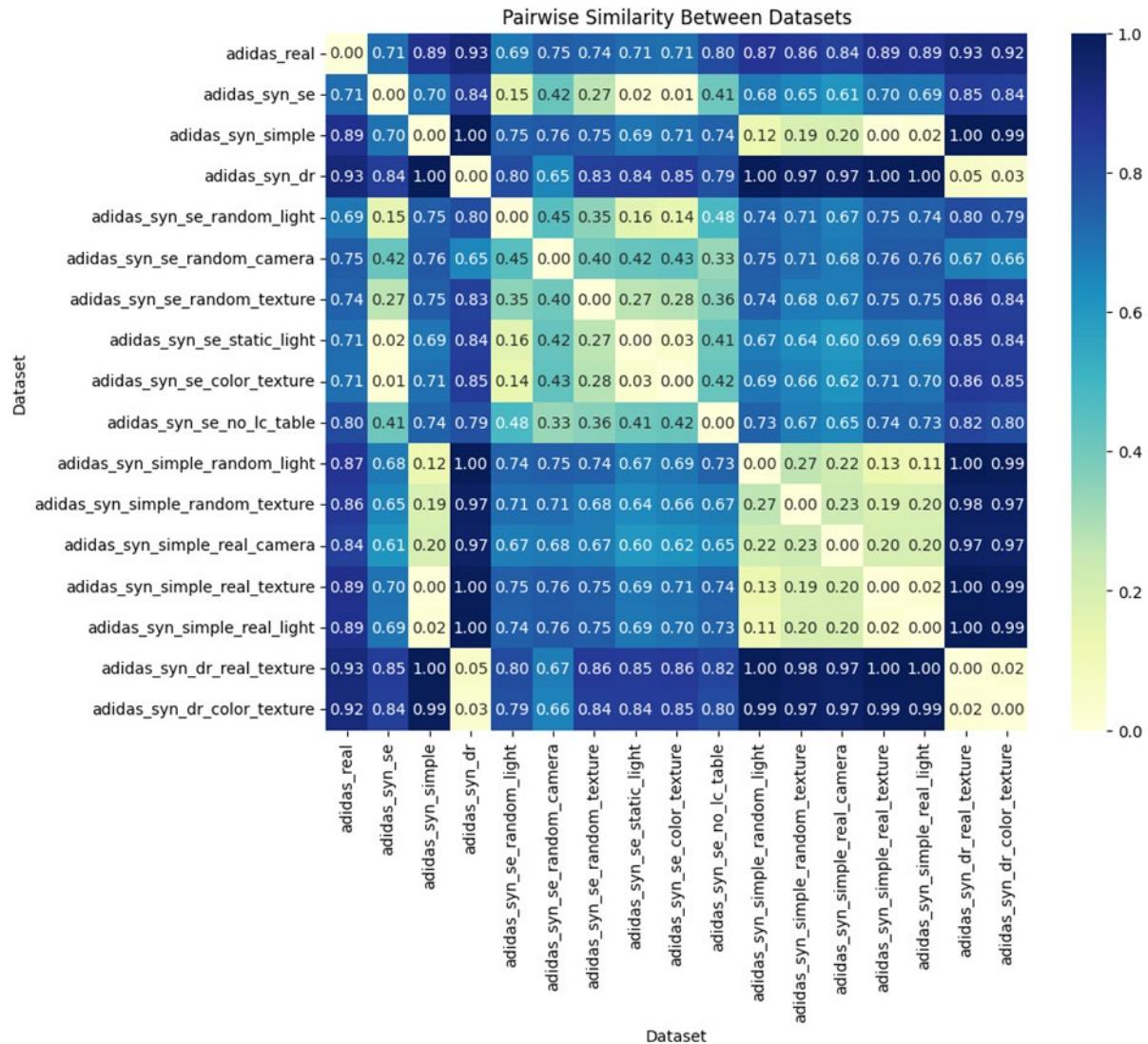


Abbildung 7.12: Feature-Vektor-Level-Distanzmatrix der Durchschnitts-Feature-Vektoren als Datensatz-Repräsentanten; Niedrige Distanzen sind hellgelb und hohe Distanzen dunkelblau hervorgehoben.

### 7.3.3 Erfasste Domänen-Reihenfolge

Im ersten Schritt der Analyse wurde die Nähe der drei synthetischen Basis-Datensätze zum Real-Datensatz untersucht, indem die Distanzen zwischen den Basis-Datensatz-Repräsentanten verglichen wurden. Ziel war es festzustellen, ob die Abstufungen in der Nähe zum Real-Datensatz mit den Performanz-Abstufungen der Objekterkennungsaufgabe übereinstimmen.

	<b>adidas_real</b>	<b>adidas_syn_se</b>	<b>adidas_syn_simple</b>	<b>adidas_syn_dr</b>
<b>adidas_real</b>	0	0.7107	0.8896	0.9331
<b>adidas_syn_se</b>	0.7107	0	0.6974	0.8411
<b>adidas_syn_simple</b>	0.8896	0.6974	0	0.9972
<b>adidas_syn_dr</b>	0.9331	0.8411	0.9972	0

*Tabelle 7.12: Ausschnitt der normalisierten Distanzmatrix der Basis-Datensätze*

An den in der Tabelle 7.12 dargestellten Distanzen zwischen den Basis-Datensatz-Repräsentanten ist eine Hierarchie der Real-Nähe erkennbar. Wie erwartet ist der SE-Basis-Datensatz mit den realimitierenden Eigenschaften näher zum Real-Datensatz als die anderen zwei, wobei der DR-Basis-Datensatz am weitesten entfernt ist. Es ist auch zu sehen, dass der DR-Basis-Datensatz zu den anderen eine große Distanz hat, wobei die geringste die zum SE-Datensatz ist. Der Simple- und SE-Basis-Datensatz haben die geringste Distanz von allen zueinander. Die Distanz-Abstufungen stimmen somit mit den erwarteten Performanz-Abstufungen der Objekterkennungsaufgabe überein und suggeriert, dass die Performanz sinkt, je weiter der Datensatz von der Real-Domäne entfernt ist.

### 7.3.4 Evaluierung der Domänen-Zuordnung

Nachdem die Domänen-Reihenfolge nach der Real-Nähe bestimmt wurde – Real – SE – Simple – DR, werden im nächsten Schritt alle Distanzen zwischen allen Varianten-Datensatz-Repräsentanten zu den Basis-Datensatz-Repräsentanten verglichen, um ihre Domänenzugehörigkeit zu bestimmen. Die Distanzen können der Distanzmatrix in Abbildung 7.12 entnommen werden. Der Schwellenwert, ab dem eine Zuordnung stattfinden soll, kann variabel festgelegt werden. Die ersten vier Datensätze der x-Achse zeigen die Basis-Datensätze. Betrachtet man die Distanzen der einzelnen Varianten zu den Domänen-Repräsentanten, ist zu erkennen, dass für die eindeutige Zuordnung der Simple- und DR-Datensätze bereits ein Schwellenwert von  $\alpha = 0.25$  ausreicht. Dies gilt jedoch nicht für die SE-Datensätze. Wie schon an der MDS-Visualisierung zu erkennen, streuen diese Datenpunkte weiter, wobei erst durch einen Schwellenwert von  $\alpha = 0.5$ , alle SE-Datensätze zugeordnet werden, inklusive dem „adidas\_syn\_se\_no\_lc\_table“ mit einer Distanz von 0,4096 und „adidas\_syn\_se\_random\_camera“

---

mit 0,4192 zum Simple-Basis-Datensatz. Ein Schwellenwert von  $\alpha = 0.5$  würde jedoch noch klein genug sein, damit Datensätze anderer Domänen nicht miteinander gruppiert werden. Die Distanzen zu den Basis-Datensätzen innerhalb der Domänen können im Anhang Domänenbildung nach Basis-Referenzwerte (B.2) nachvollzogen werden.

Alternativ wurde auch die Möglichkeit implementiert die Datensätze ohne Referenz auf Basis-Datensätze nach ihren Distanzen zu gruppieren bis zu einem festgelegten Schwellenwert. Dadurch sollen Gruppen flexibel gebildet werden, durch alleinige Betrachtung aller Distanzen und nicht nur die zu den Basis-Datensätzen. Bei einem Schwellenwert von  $\alpha = 0.4$  werden die Datensätze weitestgehend nach ihrer Domänen-Zugehörigkeit gruppiert. Nur die SE-Varianten „adidas\_syn\_se\_random\_camera“ und „adidas\_syn\_se\_no\_lc\_table“ bilden mit einer Distanz von 0,3307 zueinander eine fünfte Gruppierung. Mit sinkendem Schwellenwert wird die geforderte Ähnlichkeit erhöht. Bei einem sehr kleinen Schwellenwert von 0,1 bilden sich die Gruppierungen aus Tabelle 7.13.

	adidas_syn_se	adidas_syn_se_static_light	adidas_syn_se_color_texture
adidas_syn_se	0	0.0164	0.0149
adidas_syn_se_static_light	0.0164	0	0.0315
adidas_syn_se_color_texture	0.0149	0.0315	0

	adidas_syn_simple	adidas_syn_simple_real_texture	adidas_syn_simple_real_light
adidas_syn_simple	0	0	0.0191
adidas_syn_simple_real_texture	0	0	0.0238
adidas_syn_simple_real_light	0.0191	0.0238	0

	adidas_syn_dr	adidas_syn_dr_real_texture	adidas_syn_dr_color_texture
adidas_syn_dr	0	0.0525	0.0342
adidas_syn_dr_real_texture	0.0525	0	0.0193
adidas_syn_dr_color_texture	0.0342	0.0193	0

Tabelle 7.13: Natürliche Domänenbildung ohne Referenzwerte mit Schwellenwert von  $\alpha = 0.1$

Das zeigt, dass diese Datensätze innerhalb ihrer entsprechenden Domäne am ähnlichsten sind und Subgruppen bilden. Im Falle von den DR-Datensätzen, die alle eine sehr geringe Distanz zueinander aufweisen, entspricht diese der tatsächlichen DR-Domäne selbst.

---

### 7.3.5 Domain-Based Ranking mit Real-Nähe

Im Kapitel Distanzbasiertes Auswahlverfahren (6.6), wurden verschiedene Möglichkeiten zur Erstellung des Rankings dargestellt, die für das System implementiert wurden. Das Finale Ranking kann somit dynamisch nach bestimmten oder allen Bewertungskriterien erstellt werden: Unter Berücksichtigung der Domänen-Reihenfolge der Basis-Datensätze, der Domänen-Zugehörigkeit der Varianten, ihrer Real-Nähe und oder mit Einbeziehung der Feature-Importance des Real-Datensatzes.

Die Erfasste Domänen-Reihenfolge (7.3.3) ergab die Reihenfolge von [‘adidas\_real’, ‘adidas\_syn\_se’, ‘adidas\_syn\_simple’, ‘adidas\_syn\_dr’] und für die Domänen damit entsprechend die ordinalen Ranks [1, 2, 3, 4]. Wird zunächst nur die Real-Nähe berücksichtigt, ohne Domänen-Reihenfolge, werden die Datensatz-Repräsentanten alleinig auf ihrer Nähe zum Real-Datensatz geordnet. Datensätze mit kleinerer Distanz zum Real-Datensatz bekommen einen höheren Rank als weiter entfernte, wobei nicht eingegrenzt wird, dass Datensätze, die zu einer Domäne gehören, die nach der ermittelten Domänen-Reihenfolge besser bewertet wurde, immer besser eingestuft werden als die der schlechter bewerteten Domänen. Dabei zeigt die erste Tabelle im Anhang Ranking mit Real-Nähe (B.3), dass es keinen Datensatz einer Domäne gibt, der mit seinem Rank, aus der Reihenfolge – Real – SE – Simple – DR – fällt. Dadurch ergibt sich die gleiche Rangfolge, wie unter Berücksichtigung der Domänen im Anhang Domain-Based-Ranking mit Real-Nähe (B.4). Das bedeutet, dass sich mit oder ohne Domänenbildung nach Basis-Referenzwerte (B.2) eine Reihenfolge ergibt, bei der die Basis- und Varianten-Datensätze, sich natürlich ihren Domänen ordnen.

Im Vergleich zu Referenzwerte der Objekterkennungsaufgabe (7.3.1) zeigt sich, dass der Datensatz „adidas\_syn\_se\_no\_lc\_table“ nicht passend eingeordnet wurde. Demnach scheint dieser, trotz schlechter Performanz in der Objekterkennungsaufgabe, dem Real-Datensatz näher zu sein als die nachfolgenden Simple-Datensätze, die eine bessere Leistung erzielen. Er ist jedoch mit einer Real-Nähe von 0,8013 um 0,0344 näher als der erstgerankte Datensatz „adidas\_syn\_simple\_real\_camera“ innerhalb der Simple-Domäne mit 0,8357 und um nur 0,0883 näher als der Simple-Basis-Datensatz, der eine Real-Nähe von 0,8896 zeigt. Auch sind die beiden SE-Varianten nach den Referenzwerten mit einer Real-Nähe von 0,7402 für „adidas\_syn\_se\_random\_texture“ und 0,7533 für „adidas\_syn\_se\_random\_camera“ vertauscht.

### 7.3.6 Variance-Penalty-Reward Ranking

Um die Bedeutung wichtiger Features stärker zu berücksichtigen, wurde das Variance-Penalty-Reward Ranking eingeführt. Wie im Kapitel Variance-Penalty-Reward (6.6.6) beschrieben, wird dafür zuerst ein Feature-Importance-Ranking für die 32 Feature-Maps des Real-Datensatzes durchgeführt, um die Wichtigkeit der Features (basierend auf ihrer Varianz) im Ranking der Datensätze zu berücksichtigen.

[7, 19, 11, 26, 23, 13, 14, 20, 25, 16, 12, 9, 3, 5, 27, 24, 0, 31, 15, 1, 30, 22, 10, 21, 17, 6, 2, 8, 28, 29, 18, 4]

Die resultierende Liste, zeigt die Feature-Map-Indexe, sortiert nach ihrer Wichtigkeit. Für das Ranking der Datensätze bedeutet das, dass Feature-Map 7 am stärksten zur Importance Score des Datensatzes beiträgt. Nachdem für jeden Datensatz-Repräsentant und für jeden seiner 32 Feature-Maps die Distanz zu den entsprechenden Real- und DR-Feature-Maps berechnet wurde, wird anhand der Ergebnisse bestimmt, ob der Datensatz für die Distanzen in dieser Feature-Map belohnt oder bestraft werden soll, wobei die Stärke dieser anhand der ermittelten Feature-Importance gewichtet wird. Nachdem die Importance Score für jeden Datensatz ermittelt wurde, ergibt sich das Ranking in Anhang Domain-Based-Ranking mit Importance Score (B.6) unter Berücksichtigung der Domänen-Reihenfolge. Im Vergleich zum Ranking mit Real-Nähe als Kriterium fallen hier vier Datensätze aus der erwarteten Reihenfolge der Referenzwerte. Ohne Berücksichtigung der ermittelten Domänen-Zugehörigkeit fallen alle Datensätze raus, wobei nur der DR-Basis-Datensatz und „adidas\_syn\_dr\_real\_texture“ korrekt zueinanderstehen. Die resultierende Rangfolge ist in Ranking mit Importance Score (B.5) zu sehen.

### 7.3.7 Vergleich der Ergebnisse

	Referenzwerte der Objekterkennungsaufgabe	(Domain-Based)-Ranking mit Real-Nähe	Domain-Based Ranking mit Importance Score	Ranking mit Importance Score
1	adidas_real	adidas_real	adidas_real	adidas_real
2	adidas_syn_se_random_light	adidas_syn_se_random_light	adidas_syn_se_random_light	adidas_syn_simple_real_camera
3	adidas_syn_se	adidas_syn_se	adidas_syn_se_no_lc_table	adidas_syn_se_random_light
4	adidas_syn_se_random_camera	adidas_syn_se_random_texture	adidas_syn_se_random_camera	adidas_syn_se_no_lc_table
5	adidas_syn_se_random_texture	adidas_syn_se_random_camera	adidas_syn_se_random_texture	adidas_syn_simple
6	adidas_syn_simple_real_camera	adidas_syn_se_no_lc_table	adidas_syn_se	adidas_syn_se_random_camera
7	adidas_syn_simple	adidas_syn_simple_real_camera	adidas_syn_simple_real_camera	adidas_syn_se_random_texture
8	adidas_syn_se_no_lc_table	adidas_syn_simple	adidas_syn_simple	adidas_syn_dr_real_texture
9	adidas_syn_dr_real_texture	adidas_syn_dr_real_texture	adidas_syn_dr_real_texture	adidas_syn_dr
10	adidas_syn_dr	adidas_syn_dr	adidas_syn_dr	adidas_syn_se

Tabelle 7.14: Vergleich der Referenzwerte und der Rankings mit verschiedenen Bewertungskriterien

---

Beim direkten Vergleich in Tabelle 7.14 der Rankings verschiedener Bewertungskriterien fällt auf, dass das Ranking, alleinig unter Berücksichtigung der Real-Nähe, die wenigsten Fehleinschätzungen trifft, gefolgt vom Domain-Based Ranking mit Importance Score. Der Importance-Score als einziges Bewertungskriterium, trifft die meisten Fehleinschätzungen.

Jedoch sind diese Ergebnisse nicht aussagekräftig. Beim direkten Vergleich der Rankings, die ohne Normalisierung durch den Robust-Scaler erstellt wurde, werden die Datensätze der Simple-Domäne immer schlechter bewertet, da sie etwas weiter als die DR-Datensätze zum Real-Datensatz entfernt sind (nach den nicht-normalisierten Distanzmessungen). Diese sind in dem Anhang Auswahlverfahren: r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500 (B), der jeweils zweiten Tabelle zu entnehmen. Das Auswahlverfahren wurde auch auf die Datensatz-Kodierungen der Modelle der Zielgröße 1.536 und 6.144 angewandt.

Im Ranking von Anhang Auswahlverfahren: r\_cae\_on\_base\_comb\_fm8x12\_dim6144\_real3500 (C) und Auswahlverfahren: r\_cae\_on\_base\_comb\_fm4x6\_dim1536\_real3500 (D) ist ebenfalls zu sehen, dass sich die Datensatz-Repräsentanten inkonsistent im Auswahlverfahren verhalten. Die Distanzen eines Datensatzes einer Domäne zu dem Real-Datensatz oder auch zu den anderen Datensätzen anderer Domänen sind sich sehr ähnlich, was womöglich an dem anfangs erwähnten Fluch der Dimensionalität (2.1.2) liegt, der sich negativ auf Distanzfunktionen auswirkt, die bei hochdimensionalen Daten zunehmend an Aussagekraft verlieren. Dies impliziert, dass die Datensatz-Repräsentanten bzw. auch die Datensatz-Kodierungen selbst (vor der Aggregation), trotz der starken Dimensionsreduktion von 99,22% bei der Zielgröße von 3.072 zu groß sind, um hilfreiche Aussagen im Auswahlverfahren treffen zu können. Die Domänen-Zugehörigkeiten scheinen durch alle drei Modelle (6.144, 3.072 und 1.536) korrekt erkannt zu werden, da die Durchschnitts-Feature-Vektoren der Datensätze gleicher Domäne wohl genügend Domänen-Eigenschaften aufweisen (neben ihren spezielleren Eigenschaften), die sie im Vergleichsraum näher aneinanderbinden. Dies konnte auch schon bei der Analyse im Kapitel Latenter Raum (7.2) anhand der Visualisierungen gesehen werden.

### 7.3.8 Auswertung der SVM-Ergebnisse

Zur Überprüfung der Distanz-Ergebnisse für die Domänen-Zuordnung wurde eine SVM trainiert auf den Feature-Vektoren der Basis-Datensätze, um für die Feature-Vektoren der Varianten-Datensätze Vorhersagen über die Domänen-Zugehörigkeit zu treffen. Diese stimmen größtenteils mit den tatsächlichen Domänen-Zugehörigkeiten überein, wie in Tabelle 7.15 zu sehen.

	Variant Dataset	DR-Domain	Simple-Domain	SE-Domain	Real-Domain
1	adidas_syn_se_random_texture	825	0	13175	0
2	adidas_syn_dr_real_texture	13998	0	2	0
3	adidas_syn_simple_random_texture	2	13998	0	0
4	adidas_syn_simple_real_texture	0	14000	0	0
5	adidas_syn_dr_color_texture	13999	0	1	0
6	adidas_syn_se_random_light	152	0	13848	0
7	adidas_syn_se_static_light	0	0	14000	0
8	adidas_syn_simple_random_light	1022	12062	916	0
9	adidas_syn_simple_real_light	0	14000	0	0
10	adidas_syn_se_color_texture	0	1	13999	0
11	adidas_syn_se_random_camera	8534	1	5441	24
12	adidas_syn_simple_real_camera	0	14000	0	0
13	adidas_syn_se_no_lc_table	1072	0	12928	0

Tabelle 7.15: SVM-Ergebnisse, Vorhersagen für die Feature-Vektoren der Varianten-Datensätze

Interessant ist, dass die Datensätze mit random-Faktor, die meisten Fehl-Vorhersagen für DR treffen, wie beispielsweise Datensätze: „adidas\_syn\_simple\_random\_light“, „adidas\_syn\_se\_random\_texture“, „adidas\_syn\_se\_random\_camera“, „adidas\_syn\_se\_random\_light“. Die SVM scheint daher die random-Eigenschaften korrekterweise der DR-Domäne zuzuordnen. Auch für den Datensatz „adidas\_syn\_se\_no\_lc\_table“ gab es Fehl-Vorhersagen, was damit zusammenhängen kann, dass das fehlende Tisch-Objekt in den Feature-Vektoren als DR-Eigenschaft erfasst wird. Für eine verlässliche Vorhersage würde sich die trainierte SVM jedoch nicht eignen, da zum Beispiel für Datensatz „adidas\_syn\_se\_random\_camera“ für 8.534 von 14.000 Feature-Vektoren eine falsche Vorhersage der Domänen-Zugehörigkeit getroffen wurde. Jedoch reflektiert dies auch teilweise die Beobachtungen, die durch MDS und der Distanzmatrix visualisiert wurden. Diese SE-Variante ist im Vergleich zu den anderen SE-Datensätzen am nächsten zu den Datensätzen der DR-Domäne.

### 7.3.9 Auswertung der Clusteranalyse

Die Abbildung 7.13 zeigt die Ergebnisse der Clusteranalyse die auf den Datensatz-Repräsentanten, als Durchschnitts-Feature-Vektoren, mit UMAP durchgeführt wurde.

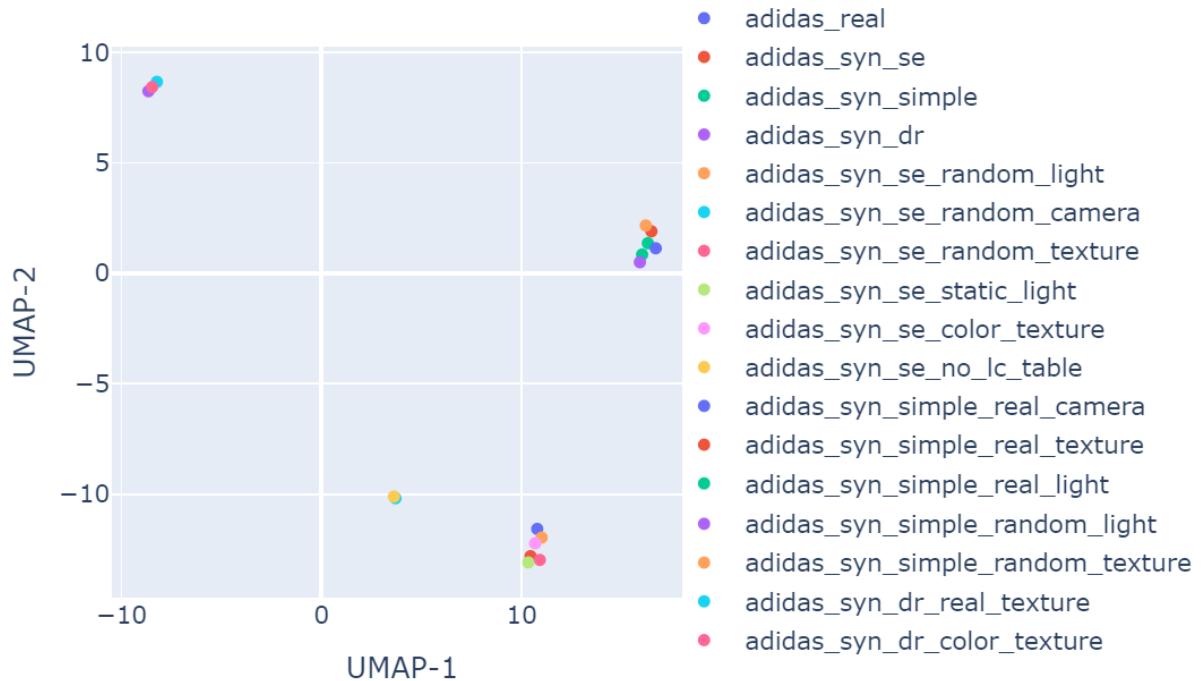


Abbildung 7.13: Ergebnisse der Cluster-Analyse durch UMAP auf den Datensatz-Repräsentanten

Daran sind auch deutlich die Domänen zu erkennen. Oben links die Datensätze der DR-Domäne, oben rechts die der Simple-Domäne und rechts unten die SE-Domäne, wobei der blaue Punkt oberhalb des SE-Clusters den Real-Repräsentanten darstellt. Die zwei unteren Ausreißer sind die beiden SE-Varianten „adidas\_syn\_se\_no\_lc\_table“ (gelb) und „adidas\_syn\_se\_random\_camera“ (hellblau).

Damit stimmt das Ergebnis der UMAP-Clusteranalyse mit den Distanzergebnissen überein und zeigt, dass die beiden zwar eine erkennbare Nähe zur SE-Domäne haben und damit Eigenschaften dieser teilen, sich jedoch von den anderen SE-Datensätzen abheben und eine eigene kleine Subgruppe bilden, die den DR-Datensätzen nahe kommen.

---

## 8 Diskussion

### 8.1 Modell-Grenzen

Die Ergebnisse des Auswahlverfahrens – der Distanzmessungen, der Clusteranalyse und des SVM-Trainings – hängen stark von den Feature-Vektoren ab, auf denen sie angewandt wurden. Die Qualität der Feature-Vektoren wiederum ist direkt auf die Modell-Leistung der trainierten R-CAE-Modelle zurückzuführen und der anschließenden Aggregation zu den Durchschnitts-Feature-Vektoren.

Da ressourcenbedingt die Modell-Komplexität klein gehalten wurde, spiegelt sich das auch in den Ergebnissen der Rekonstruktionen entsprechend der Reduktionsstärke wider. Bei höherer Speicherkapazität und Rechenleistung, wäre es vermutlich möglich, die Größe der Eingabedaten noch weiter zu reduzieren bei gleichbleibender Qualität. Demnach wäre die Erstellung komplexerer Modelle ein Punkt, den es weiter zu untersuchen gilt. Denn je besser das Modell, desto besser die resultierenden Feature-Vektoren und damit Kodierungen ganzer Datensätze, die als Grundlage für das Auswahlverfahren dienen.

Insbesondere die Strategien der Filteranzahl-Reihenfolge sollte weiter untersucht werden. In vielen Beispielen wird für die Anwendung in CAEs eine Reihenfolge nach CNN-Prinzip gewählt, da dies die Verwendung eines Undercomplete AEs nicht ausschließt und die Reduktion der räumlichen Dimension durch Erhöhung der Filter kompensiert wird. Jedoch gibt es wenige Beispiele, die sich mit dem Einfluss der Reihenfolge beschäftigen, wenn es darum geht die Ergebnisse des Encoders (die Feature-Vektoren) weiterzuverarbeiten und nicht die guten Rekonstruktionen selbst das Ziel sind. Da es beide Filter-Ansätze gibt – abnehmend und wachsend –, wäre es interessant zu untersuchen, inwiefern dies die Distanzmessungen beeinflussen würde, da die Filteranzahl der letzten Encoder-Schicht die Anzahl der Feature-Maps der Feature-Vektoren im Latenten Raum bestimmt. Ob es für die Distanzmessungen der Datensatz-Repräsentanten, aber insbesondere auch für die auf Feature-Map-Level ein Vorteil wäre, mehr Feature-Maps zu haben, jedoch mit kleinerer räumlicher Dimension oder weniger Feature-Maps mit größerer Dimension, müsste weiter untersucht werden. Da die zwei getesteten Modelle ressourcenbedingt, nur mit einer generell kleinen Anzahl an Filtern getestet werden konnte, sollten diese Ergebnisse nicht als definitiv angesehen werden. Daraus sollte nicht abgeleitet werden, dass ein Modell mit wenigen aber größeren Feature-Maps im Latenten Raum zu besseren Ergebnissen führt als ein Modell, welches viele aber sehr kleine Feature-Maps erzeugt.

Ein weiterer Kritikpunkt ist die zweite Dimensionsreduktion durch die Aggregation über die  $n$  Feature-Vektoren, die nach der Kodierung entstehen. Denn das verursacht ein Informationsverlust der individuellen Kodierungen, da nur die durchschnittlichen Zustände verglichen werden. Mit der ersten Dimensionsreduktion durch den R-CAE-Encoder könnten durch komplexere Modelle potenziell

---

kleinere Zielgrößen erreicht werden, was wiederum die Komplexität der weiteren Verarbeitungsschritte reduziert und eine stärkere Komprimierung und damit Fokussierung der relevanten, unterscheidungskräftigsten Merkmale bietet, die einen Datensatz charakterisieren. Jedoch würde sich die zweite Dimensionsreduktion nicht vermeiden lassen. Denn auch wenn ein Modell trainiert werden könnte, welches mit sehr wenigen Feature-Vektoren - beispielsweise (2,3,192) – 768 Werte pro Feature-Vektor - effektive Kodierungen erzeugt, würde dies bei  $n$  Feature-Vektoren, dennoch zu ( $n$ , 768) zu vergleichenden Vektoren führen und dies für  $m$  zu vergleichende Datensätze. Einen Repräsentanten für jeden Datensatz zu bestimmen, würde deshalb trotzdem notwendig sein, wenn es das Ziel ist, ein nicht aufwändiges Vergleichsverfahren anzuwenden. Ausgehend davon sollten weitere Methoden untersucht werden, wie eine zweite Dimensionsreduktion auf den Ergebnissen der Kodierungen weiter für den Vergleich im Auswahlverfahren stattfinden kann.

## 8.2 Grenzen des Auswahlverfahrens

Das Auswahlverfahren selbst wurde explizit auf Methoden beschränkt, die kein zusätzliches Training pro zu bewertenden Datensatz benötigen, die zu einer individuellen Performance-Score führen würden. Das ursprüngliche Ziel war es, so weit wie möglich die Bewertung unabhängig von den Datensätzen und dem Anwendungsfällen zu erstellen, weswegen das Auswahlverfahren nach der allgemein gängigen Annahme – Realdaten eignen sich am besten – aufgebaut wurde. Zur Überprüfung der Annahme wurden jedoch die Performance-Scores der vorangegangenen Objekterkennungsaufgabe herangezogen, wodurch die Auswertung von diesem Anwendungsfällen beeinflusst wurde. Dabei könnte jedoch argumentiert werden, dass die aus der Objekterkennungsaufgabe resultierende Domänen-Reihenfolge – Real – SE – Simple – DR – bereits auf der vorherherrschenden Annahme beruht, dass SE-Daten als Realimitationen meist bessere Leistungen erzielen. Da das Ranking im System jedoch dynamisch nach den implementierten Bewertungskriterien gestaltet werden kann, ist es möglich es je nach Bedarf einzustellen, wie beispielsweise nach Ähnlichkeit eines Datensatzes bestimmter Domäne und nicht nur zum Real-Datensatz.

Unabhängig von Datensätzen ist der Domänenbasierte Ansatz jedoch nicht. Denn übertragen auf andere Datensätze könnten diese womöglich keine so eindeutige Domänen-Abgrenzung aufweisen. Aus diesem Grund wurde die Alternative implementiert, die Domänen bzw. Gruppen selbst erkennen zu lassen ohne Referenz auf die Basis-Datensätze. Jedoch müsste für diese Variante am Ende noch ein Bewertungskriterium festgelegt werden, nach welchem die Datensätze innerhalb der Domäne geordnet werden sollten. Dieser Ansatz würde sich eher dafür eignen, ähnliche Datensätze zu identifizieren. Ein Anwendungsfällen dafür wäre, bei Notwendigkeit weiterer Trainingsdaten, die aktuellen Trainingsdaten mit den weiteren verfügbaren zu vergleichen und auf Basis dessen eine Entscheidung zu treffen. Oder auch um erst testweise mit variablen Schwellenwert die Domänenrepräsentierenden Datensätze zu identifizieren und diese dann als Basis-Datensätze zu verwenden. Wobei dies voraussetzt, dass zum

---

Training der CAE-Modelle selbst Datensätze gewählt wurden, die genügend Variation bieten (für effektive Kodierungen der Varianten). Die Wahl geeigneter Datensätze (für das CAE-Training) bzw. für das Performanz-Schätzungs-System selbst ist daher ironischerweise ein großer Kritikpunkt.

Zu der Funktion des Variance-Penalty-Reward Rankings ist noch zu erwähnen, dass dieser Ansatz der Belohnung von Real-Eigenschaften und gleichzeitiger Bestrafung von DR-Eigenschaften die Vorteile der Randomisierung vernachlässigt. Datensätze mit variablen Umgebungsverhältnissen können je nach Anwendungsfall sehr vorteilhaft sein, um das Modell auf verschiedene Situationen vorzubereiten, wie es auch an den Referenzwerten der Objekterkennungsaufgabe zu sehen war. Deshalb ist es auch nicht verwunderlich, dass dieses Ranking unter alleiniger Berücksichtigung der errechnete Importance-Score als Bewertungskriterium, die meisten Fehleinschätzungen trifft.

Das Ranking mit Real-Nähe für die Datensatz-Repräsentanten, die durch das Encoder-Modell von „r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500“ kodiert wurden, ergab drei Fehleinschätzungen bei zehn Referenzwerten. Jedoch sind diese nicht sehr aussagekräftig, da im Vergleich zu den Ergebnissen der Rankings ohne Normalisierung und im direkten Vergleich mit den Modellen der Zielgröße 6.144 und 3.072 sich sehr unterschiedliche ordinale Folgen ergeben. Sie erkennen zwar alle, dass die SE-Datensätze am nächsten dem Real-Datensatz kommen, aber die Reihenfolge der Simple- und DR-Domäne ist nicht konsistent. Das könnte natürlich daran liegen, dass die Modelle unterschiedliche Aspekte aufgrund ihrer größeren (6.144) und kleineren (3.072) Kapazität im Latenten Raum wahrnehmen. Jedoch sind sich die Distanzen innerhalb der Domänen zum Real-Datensatz sehr ähnlich, was der anfangs erwähnten Befürchtung entspricht, dass auch eine starke Dimensionsreduktion, wie auf Zielgröße 1.536 - (4,6,64) noch zu hochdimensional ist, als dass die Distanzmessungen brauchbare Ergebnisse liefern. Das würde auch erklären, weshalb zumindest die großen Domänen-Unterschiede als solche erkannt werden. Daher wäre es notwendig weiter zu untersuchen, ob eine noch stärkere Reduktion eine bessere Grundlage für das Auswahlverfahren liefert, vorausgesetzt die Rekonstruktionsqualität bleibt erhalten.



---

## 9 Zusammenfassung und Ausblick

Das Projektziel war es, die verfügbaren Datensätze in eine vergleichbare Form zu bringen, um sie hinsichtlich ihrer generellen Eignung für ML-Aufgaben zu bewerten, ohne für jeden Datensatz ein individuelles Training zur Bewertung durchzuführen. Zur Dimensionsreduktion wurden R-CAE Modelle mit unterschiedlichen Konfigurationen trainiert, um die optimale Dimensionsgröße zu ermitteln, die eine starke Kompression erzeugt und dennoch die strukturellen Informationen und Eigenschaften der Eingabebilder beibehält. Dafür wurden die Bilder aller Datensätze unter Einbezug der Maskeninformationen vorverarbeitet.

Damit die Modelle dazu fähig sind vielseitige Merkmale unterschiedlicher Datensatz-Arten und Domänen erfassen zu können, wurden vier Basis-Datensätze aus der Menge an verfügbaren Datensätzen bestimmt, die als Domänen-Repräsentanten dienen. Die Modelle wurden auf jeweils 3.500 Daten der Basis-Datensätze trainiert und mit den Evaluierungsmethoden ausgewertet. Sowohl die Analyse des Latenten Raums als auch die Evaluierungsmethoden bestätigen die anfangs getroffene Annahme, dass eine starke Dimensionsreduktion trotz des Verlustes der Bildqualität noch die wesentlichen Merkmale des Datensatzes erfassen kann. Die Ergebnisse der Evaluierung zeigen auch, dass die Basis-Datensätze als gewählte Domänen-Repräsentanten genügend Vielfalt in ihren Trainingsdaten bieten, um die spezifischen Eigenschaften der Varianten-Datensätze zu erfassen. Die Annahme, dass ein ausgewogener Anteil an Real-Daten wie synthetische Trainingsdaten erforderlich ist, bzw. dass die Domänen gleichermaßen vertreten sein sollten, konnte nur teilweise bestätigt werden, da sie zusätzlich von der Zielgröße des Modells abhängt. Nach der Modell-Evaluierung könnte für das R-CAE Modell mit Zielgröße von 3.072 der Real-Anteil von 3.500 auf 2.500 reduziert werden, bei gleicher Rekonstruktionsqualität. Ausgehend von der Modell-Evaluierung wurde von dem Modell „r\_cae\_on\_comb\_fm8x12\_dim3072\_real3500“ der Encoder-Part extrahiert, um alle Datensätze, einschließlich der Basis-Datensätze zu kodieren.

Dadurch wurden die 17 Datensätze der Form ( $n, 256, 384, 4$ ) auf ( $n, 8, 12, 32$ ) reduziert, was eine Reduktion von 99,22% darstellt. Es entstehen  $n$  Feature-Vektoren, wobei ein Feature-Vektor die Kodierung eines ursprünglichen Bildes in 32 Feature-Maps der Größe  $8 \times 12$  zeigt – im Latenten Raum. Da  $n$  Feature-Vektoren noch eine zu große zu vergleichende Menge darstellen, wurden die Kodierungen pro Datensatz aggregiert, indem der Durchschnitts-Wert jeder Pixelposition in jeder Feature-Map und für alle Feature-Maps berechnet wurde. Damit wurde ein Durchschnitts-Feature-Vektor für jeden Datensatz erstellt. Diese wurden mit dem Robust-Scaler normalisiert und dann als Datensatz-Repräsentanten im Auswahlverfahren miteinander verglichen. Zum Vergleich wurde eine Distanzmatrix mit der Manhattan-Distanz auf Feature-Vektor-Level für alle Datensätze und auf Feature-Map-Level für jede Feature-Map aller Datensätze, aufgestellt. Diese Distanzen wurden verwendet, um die Domänen-Reihenfolge zu bestimmen, die angibt, wie nah die synthetischen Basis-Datensätze – SE –

---

Simple – DR – dem Real-Datensatz kommen. Danach wurden auf Feature-Vektor-Level die Distanzen der Varianten- zu den Basis-Datensätzen betrachtet, um ihre Domänen-Zugehörigkeit zu bestimmen, wofür der Schwellenwert auf 0,5 festgelegt wurde. Die Annahme, dass Datensätze, dessen Domänen-Zugehörigkeit nicht eindeutig im Auswahlverfahren bestimmt wird, außerhalb der Bewertung fallen, wurde nur teilweise bestätigt, da dies von der Wahl des Schwellenwerts abhängt. Bei einem Schwellenwert von 0,5, werden noch alle Domänenzugehörigkeiten korrekt erfasst, ohne dass Domänen-Überlappungen auftreten. Ein kleinerer Wert führt dazu, dass nicht zuordbare aus dem Auswahlverfahren fallen und nur die ähnlichsten Datensätze Gruppen bilden.

Zuletzt wurde die ermittelte Domänen-Reihenfolge, die Domänen-Zugehörigkeit der Varianten-Datensätze und die Real-Nähe auf Feature-Vektor-Level verwendet, um ein Domänenbasiertes Ranking zu erstellen, welches die Datensätze erst nach ihrer Domänen-Zugehörigkeit bewertet und innerhalb der Domänen nach ihrer Real-Nähe. Zuletzt wurden die Distanzen auf Feature-Map-Level verwendet, um ein Variance-Penalty-Reward Ranking zu erstellen, welches die Feature-Importance (gemessen an der Varianz) berücksichtigt.

Da die Ergebnisse der Rankings nicht sehr aussagekräftig sind, wäre es neben dem Versuch, die Dimensionalität noch stärker zu reduzieren, auch für zukünftige Forschungen und zur Weiterentwicklung des Performanz-Schätzungs-Systems interessant, weitere Modelle basierend auf CAEs anzuwenden. Beispielsweise der Contractive und Sparse AEs, um zu vergleichen, wie sich die Datensatz-Kodierungen, gewonnen durch diese Modellarchitekturen, im Auswahlverfahren verhalten. Oder auch eine Art Stacked Deep CAE, der erst eine starke Dimensionsreduktion erzeugt – wie die Modelle des Projekts -, aber dann ein weiterer CAE nachgeschaltet wird, der versucht die Kodierungen des ersten CAEs zu rekonstruieren und dadurch selbst wieder eine reduzierte Darstellung dieser im Latenten Raum erzeugt – als eine zweite Dimensionsreduktion.

Des Weiteren könnte eine Automatisierung des Auswahlverfahrens - durch weitere maschinelle Lernverfahren - eine zusätzliche Optimierung für die Datensatzauswahl bieten, vorausgesetzt sie erfordern keine individuellen Trainings. Ein weiterer Schritt könnte die Integration zusätzlicher statistischer Metriken sein, die neben der Varianz zur Bestimmung der Feature-Importance oder zur Bewertung einzelner Feature-Maps angewendet werden könnten. Die Anwendung dieser Metriken in statistischen Verfahren auf Basis der Feature-Vektoren der Datensatz-Kodierungen könnte dazu beitragen, eine genauere Bewertung der Datensatzqualität zu erzielen – als eine Kombination von lernenden und statistischen Verfahren.

---

## Literaturverzeichnis

Abouzid, H., Chakkor, O., Reyes, O., & Ventura, S. (2019). Signal speech reconstruction and noise removal using convolutional denoising autoencoders with neural deep learning. *Analog Integrated Circuits and Signal Processing*. doi:<https://doi.org/10.1007/s10470-019-01446-6>

Aggarwal, C., Hinneburg, A., & Keim, D. (2002). On the Surprising Behavior of Distance Metric in High-Dimensional Space. *Database theory, ICDT 200, 8th International Conference*. Retrieved from [https://www.researchgate.net/publication/30013021\\_On\\_the\\_Surprising\\_Behavior\\_of\\_Distance\\_Metric\\_in\\_High-Dimensional\\_Space](https://www.researchgate.net/publication/30013021_On_the_Surprising_Behavior_of_Distance_Metric_in_High-Dimensional_Space)

Altman, N., & Krzywinski, M. (2018). The curse(s) of dimensionality. *Nature Methods*, 1. doi:<https://doi.org/10.1038/s41592-018-0019-x>

Bellman, R. E. (1957). *Dynamic Programming*. NJ, USA: Princeton University Press. doi:<https://doi.org/10.1126/science.153.3731.34>

Berahmand, K., Daneshfar, F., Salehi, E., Li, Y., & Xu, Y. (2024). Autoencoders and their applications in machine learning: a survey. *Artificial Intelligence Review*. doi:<https://doi.org/10.1007/s10462-023-10662-6>

Berisha, V., Krantsevich, C., Hahn, P. R., Dasarathy, G., Turaga, P., & Liss, J. (2021). Digital medicine and the curse of dimensionality. *npj Digital Medicine*. doi:<https://doi.org/10.1038/s41746-021-00521-5>

Bernard, E. (2021). Dimensionality Reduction. In E. Bernard, *Introduction to Machine Learning*. Wolfram Media, Incorporated.

Bing, X., Naiyan, W., Tianqi, C., & Mu, L. (2015). Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv*. doi:<https://doi.org/10.48550/arXiv.1505.00853>

Choi, J. S., Kim, J., & Ko, J. K. (2021). A Weight-Sharing Autoencoder with Dynamic Quantization for Efficient Feature Compression. *2021 International Conference on Information and Communication Technology Convergence (ICTC)*. doi:<https://doi.org/10.1109/ICTC52510.2021.9620912>

Chollet, F. (2016, 05 14). *Building Autoencoders in Keras*. Retrieved from The Keras Blog: <https://blog.keras.io/building-autoencoders-in-keras.html>

---

Cunningham, H., Ewart, A., Riggs, L., Huben, R., & Sharkey, L. (2023). Sparse Autoencoders Find Highly Interpretable Features in Language Models. *arXiv*.  
doi:<https://doi.org/10.48550/arXiv.2309.08600>

Dertat, A. (2017, 10 03). *Applied Deep Learning - Part 3: Autoencoders*. Retrieved from Towardsdatascience:  
[https://miro.medium.com/v2/resize:fit:1100/format:webp/1\\*44eDEuZBEsmG\\_TCAKRI3Kw@2x.png](https://miro.medium.com/v2/resize:fit:1100/format:webp/1*44eDEuZBEsmG_TCAKRI3Kw@2x.png)

Dhinakaran, A. (2023, 02 02). *Understanding KL Divergence*. Retrieved from Towardsdatascience:  
<https://towardsdatascience.com/understanding-kl-divergence-f3ddc8dff254>

Diallo, B., Hu, J., Li, T., Khan, G. A., Liang, X., & Zhao, Y. (2021). Deep embedding clustering based on contractive autoencoder. *Deep embedding clustering based on contractive autoencoder*.  
doi:<https://doi.org/10.1016/j.neucom.2020.12.094>

Ding, Y., Kui, Z., & Weihong, B. (2020, 08 01). *Feature selection based on hybridization of genetic algorithm and competitive swarm optimizer*. Retrieved from ResearchGate:  
[https://www.researchgate.net/figure/Difference-between-feature-extraction-and-feature-selection\\_fig1\\_339209170](https://www.researchgate.net/figure/Difference-between-feature-extraction-and-feature-selection_fig1_339209170)

Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep. *arXiv*.

Ezukwoke, K., & Zareian, S. (2019). A comparative study of classical and kernel PCA. In K. Ezukwoke, & S. Zareian, *KERNEL METHODS FOR PRINCIPAL COMPONENT ANALYSIS (PCA)*. doi:<https://doi.org/10.13140/RG.2.2.17763.09760>

Fakultät W. (2016). Richtlinien und Hinweise zur Anfertigung wissenschaftlicher Arbeiten an der Fakultät für Wirtschaftswissenschaften. pp. 1-46.

Frenzel, M. (2020, 03 02). *CompressionVAE — A Powerful and Versatile Alternative to t-SNE and UMAP*. Retrieved from Towardsdatascience: <https://towardsdatascience.com/compressionvae-a-powerful-and-versatile-alternative-to-t-sne-and-umap-5c50898b8696>

Garima, N. (2020, 08 03). *Reconstruct corrupted data using Denoising Autoencoder(Python code)*. Retrieved from Medium:  
[https://miro.medium.com/v2/resize:fit:1100/format:webp/1\\*qKiQ1noZdw8k05-YRIl6hw.jpeg](https://miro.medium.com/v2/resize:fit:1100/format:webp/1*qKiQ1noZdw8k05-YRIl6hw.jpeg)

Genender-Feltheimer, A. (2018). Visualizing High Dimensional and Big Data. *Procedia Computer Science*. doi:<https://doi.org/10.1016/j.procs.2018.10.308>

---

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media. Retrieved from <https://raw.githubusercontent.com/data-science-projects-and-resources/Data-Science-EBooks/main/Machine%20Learning/Hands-on-Machine-Learning.pdf>

Guo, P., Valanarasu, J. M., Wang, P., Zhou, J., Jiang, S., & Patel, V. M. (2021). Over-and-Under Complete Convolutional RNN for MRI Reconstruction. *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer International Publishing. doi:[https://doi.org/10.1007/978-3-030-87231-1\\_2](https://doi.org/10.1007/978-3-030-87231-1_2)

Harmouch, M. (2021). *17 Types of similarity and dissimilarity measures used in data science*. Retrieved from Towardsdatascience: <https://towardsdatascience.com/17-types-of-similarity-and-dissimilarity-measures-used-in-data-science-3eb914d2681>

Hira, Z., & Gillies, D. (2015). A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data. *PubMed-not-MEDLINE*. doi:<https://doi.org/10.1155/2015/198363>

Houle, M. K. (2010). Can Shared-Neighbor Distances Defeat the Curse of Dimensionality? In M. K. Houle, *Scientific and Statistical Database Management* (p. 483). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:[https://doi.org/10.1007/978-3-642-13818-8\\_34](https://doi.org/10.1007/978-3-642-13818-8_34)

Ippolito, P. (2019). *Feature Extraction Techniques*. Retrieved from Towardsdatascience: <https://towardsdatascience.com/feature-extraction-techniques-d619b56e31be>

Jäckle, S. (2017). Multidimensionale Skalierung. In P. König, *Neue Trends in den Sozialwissenschaften*. Springer.

Jia, W., Sun, M., Lian, J., & Hou, S. (2022). Feature dimensionality reduction: a review. *Complex & Intelligent Systems*.

Joos, K. (2021). *Multidimensional Scaling*. Retrieved from Towardsdatascience: <https://towardsdatascience.com/multidimensional-scaling-d84c2a998f72>

Kaiming, H., Xiangyu, Z., Shaoqing, R., & Jian, S. (2015). Deep Residual Learning for Image Recognition. *arXiv*. doi:<https://doi.org/10.48550/arXiv.1512.03385>

Keiron, O., & Ryan, N. (2015). *An Introduction to Convolutional Neural Networks*. Retrieved from ResearchGate: [https://www.researchgate.net/figure/Activations-taken-from-the-first-convolutional-layer-of-a-simplistic-deep-CNN-after\\_fig5\\_285164623](https://www.researchgate.net/figure/Activations-taken-from-the-first-convolutional-layer-of-a-simplistic-deep-CNN-after_fig5_285164623)

---

Kobak, D., & Linderman, G. C. (2019). UMAP does not preserve global structure any better than t-SNE when using the same initialization. *bioRxiv*.  
doi:<https://doi.org/10.1101/2019.12.19.877522>

Kolarik, M., Burget, R., & Riha, K. (2019). Upsampling Algorithms for Autoencoder Segmentation Neural Networks: A Comparison Study. *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*.  
doi:<https://doi.org/10.1109/ICUMT48472.2019.8970918>

Kuhn, M., & Johnson, K. (2019). 1.4. Feature Selection. In J. K. Kuhn Max, *Feature Engineering and Selection*. CRC Press.

Lanham, M. (2023). *Evolutionary Deep Learning: Evolving autoencoders*. Retrieved from Manning: <https://drek4537l1klr.cloudfront.net/lanham/v-12/Figures/08image001.png>

Leyer, I., & Wesche, K. (2007). Hauptkomponentenanalyse (PCA). In *Multivariate Statistik in der Ökologie*. Heidelberg, Berlin: Springer-Verlag. doi:<https://doi.org/10.1007/978-3-540-37706-1>

Lokman, S. F., Othman, A. T., Musa, S., & Abu Bakar, M. H. (2019). Deep Contractive Autoencoder-Based Anomaly Detection for In-Vehicle Controller Area Network (CAN). In S. F. Lokman, A. T. Othman, S. Musa, & M. H. Abu Bakar, *Progress in Engineering Technology*. Springer International Publishing. doi:[https://doi.org/10.1007/978-3-030-28505-0\\_16](https://doi.org/10.1007/978-3-030-28505-0_16)

McInnes, L., & Healy, J. (2018). Theoretical Foundations for UMAP. In L. McInnes, & J. Healy, *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. ArXiv e-prints.

Michelucci, U. (2022). An Introduction to Autoencoders. *arXiv*.  
doi:<https://doi.org/10.48550/arXiv.2201.03898>

Nadipally, M. (2019). Peak Signal-to-Noise Ratio. In M. Nadipally, *Intelligent Data Analysis for Biomedical Applications*. Academic Press. doi:<https://doi.org/10.1016/B978-0-12-815553-0.00002-1>

Nguyen LH, H. S. (2019, June 20). *Ten quick tips for effective dimensionality reduction*. Retrieved from PLoS Computational Biology: <https://doi.org/10.1371/journal.pcbi.1006907>

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. *ArXiv*.  
doi:<https://doi.org/10.48550/arXiv.1511.08458>

- 
- Oskolkov, N. (2019, 10 04). *How Exactly UMAP Works*. Retrieved from Towardsdatascience: <https://towardsdatascience.com/why-umap-is-superior-over-tsne-faa039c28e99>
- Pani, A. K. (2022). Non-linear process monitoring using kernel principal component analysis: A review of the basic and modified techniques with industrial applications. *Brazilian Journal of Chemical Engineering*. doi:<https://doi.org/10.1007/s43153-021-00125-2>
- Rajas, B. (2021, 06 28). *Stacked Autoencoders*. Retrieved from Medium: [https://miro.medium.com/v2/resize:fit:1100/format:webp/1\\*9etlfhalsPFJ8Eh\\_Cmlobg.png](https://miro.medium.com/v2/resize:fit:1100/format:webp/1*9etlfhalsPFJ8Eh_Cmlobg.png)
- Rohwedder, L. H. (2007, 9 9). *Hyperbolic Convolutional Neural Networks - Scientific Figure on ResearchGate*. Retrieved from ResearchGate: [https://www.researchgate.net/figure/Earth-as-the-simplest-example-of-a-manifold-reproduced-from-Wikipedia\\_fig4\\_373518451](https://www.researchgate.net/figure/Earth-as-the-simplest-example-of-a-manifold-reproduced-from-Wikipedia_fig4_373518451)
- Schintler, L. A. (2020). High DIimensional Data. In L. A. Schintler, *Encyclopedia of Big Data* (pp. 1-3). Fairfax, VA, USA: Springer International Publishing. doi:[https://doi.org/10.1007/978-3-319-32001-4\\_552-1](https://doi.org/10.1007/978-3-319-32001-4_552-1)
- Sharma, A. (2021, 04 26). *Variational Autoencoder in TensorFlow*. Retrieved from LearnOpenCV: <https://learnopencv.com/wp-content/uploads/2020/11/vae-diagram-1-1024x563.jpg>
- Shinde, H. (2018, 09 24). *Deep Autoencoders for movie recommendations - A practical approach*. Retrieved from Medium: [https://miro.medium.com/v2/resize:fit:1100/format:webp/1\\*5j8KAPbFLbgzT3BWvrs75g.png](https://miro.medium.com/v2/resize:fit:1100/format:webp/1*5j8KAPbFLbgzT3BWvrs75g.png)
- Shuwei, Z., Yefeng, L., Xingyu, L., Shibo, L., Xiaofeng, X., & Lihai, Z. (2023). AESR3D: 3D overcomplete autoencoder for trabecular computed tomography super resolution. *CAAI Transactions on Intelligence Technology*.
- Siva, S. (2020, 05 31). *Dimensionality Reduction for Data Visualization: PCA vs TSNE vs UMAP vs LDA*. Retrieved from Towardsdatascience: <https://towardsdatascience.com/dimensionality-reduction-for-data-visualization-pca-vs-tsne-vs-umap-be4aa7b1cb29>
- Skliar, A., & Weiler, M. (2023). Topology and Smooth Manifolds. In A. Skliar, & M. Weiler, *Hyperbolic Convolutional Neural Networks*. doi:<https://doi.org/10.48550/arXiv.1910.12933>
- Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2015). Striving for Simplicity: The All Convolutional Net. *arXiv*. doi:<https://doi.org/10.48550/arXiv.1412.6806>

---

Van der Maaten, L., & Geoffrey, H. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*. Retrieved from  
[https://www.cns.nyu.edu/events/spf/SPF\\_papers/JMLR\\_Final.pdf](https://www.cns.nyu.edu/events/spf/SPF_papers/JMLR_Final.pdf)

Vlachos, M. (2011). Dimensionality Reduction. *Encyclopedia of Machine Learning*.  
doi:[https://doi.org/10.1007/978-0-387-30164-8\\_216](https://doi.org/10.1007/978-0-387-30164-8_216)

Wang, Z., Bovik, A. C., & Sheikh, H. R. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*.  
doi:<https://doi.org/10.1109/TIP.2003.819861>

Wilimitis, D. (2018, 12 12). *The Kernel Trick in Support Vector Classification*. Retrieved from  
Towardsdatascience: <https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f>

---

## Abbildungsverzeichnis

Abbildung 2.1: Veranschaulichung der Data Sparsity in hochdimensionalen Daten; Quelle: (Altman & Krzywinski, 2018).....	6
Abbildung 2.2: Veranschaulichung der Dimensionsreduktion: Feature-Extraction (links) und Feature-Selection (rechts); Quelle: (Ding, Kui, & Weihong, 2020).....	7
Abbildung 2.3: Veranschaulichung von PCA; Bestimmung der Achsen, dessen Projektion zur meisten Varianz beiträgt; Quelle: ( Géron, 2019, S. 222), Figure 8-7) .....	8
Abbildung 2.4: Prinzip des Kernel-Tricks: Transformation vom eindimensionalem zum linear separierbaren zweidimensionalen Raum; Quelle: (Wilimitis, 2018).....	9
Abbildung 2.5: Erdoberfläche zur Veranschaulichung eines Manifold; Quelle: (Rohwedder, 2007), (Skliar & Weiler, 2023, S. 7).....	10
Abbildung 2.6: Übersicht der Feature-Extraction Methoden; Angelehnt, an Quelle: ( Berahmand, Daneshfar, Salehi, Li, & Xu, 2024, S. 4), Fig.1).....	13
Abbildung 2.7: Klassisches Autoencoder-Modell mit Encoder und Decoder als Sub-Modelle; Quelle: (Dertat, 2017) .....	14
Abbildung 2.8: Modellarchitektur eines Convolutional Autoencoder (CAE); Quelle: (Lanham, 2023) .....	16
Abbildung 2.9: Residual-Learning-Block; Quelle: ( Kaiming, Xiangyu, Shaoqing, & Jian, 2015, S. 2), Fig.2).....	19
Abbildung 3.1: Modellarchitektur eines Variational Autoencoders (VAE); Quelle: (Sharma, 2021) ..	23
Abbildung 3.2: Darstellung der tiefen Architektur eines Deep AEs mit mehreren Hidden-Layers; Quelle: (Shinde, 2018) .....	24
Abbildung 3.3: Architektur eines Stacked AEs bestehend aus mehreren einzelnen AEs; Quelle: (Rajas, 2021).....	25
Abbildung 3.4: Übersicht verschiedener AE-Typen kategorisiert nach ihrer Architektur; Quelle: (Berahmand, Daneshfar, Salehi, Li, & Xu, 2024, S. 16), Fig.4). .....	26
Abbildung 6.1: Conv-Block (Standard Stride = (1,1), bestehend aus einer Conv2D, Batch Normalization und LeakyReLU-Aktivierung.....	46

---

Abbildung 6.2: Conv-Block (Downsampling Stride = (2,2) .....	46
Abbildung 6.3: DeConv-Block zum Upsampling mit Stride = (2,2), bestehend aus einem Conv2DTranspose-Layer, Batch Normalization und LeakyReLU-Aktivierung.....	47
Abbildung 6.4: Deep-Conv-Block mit 3 Conv-Blöcken, zum reinen Feature Learning.....	47
Abbildung 6.5: Residual-Block als Deep-Conv-Block mit Residual Connection.....	48
Abbildung 6.6: Encoding-Schritt mit Residual-Block für R-CAE (links) oder Deep-Conv-Block für CAE (rechts).....	48
Abbildung 6.7: Decoding-Schritt mit Residual-Block für R-CAE (links) oder Deep-Conv-Block für CAE (rechts)) .....	48
Abbildung 6.8: Aufbau des CAE/ R-CAE mit Submodellen Encoder (links) und Decoder (rechts)....	49
Abbildung 6.9: Mischform eines Autoencoders mit Conv- und Dense- Encoder und Decoder .....	50
Abbildung 6.10: R-CAE Modell „r_cae_on_base_comb_fm8x12_dim3072_real3500“ mit 5 Downsamplings im Encoder zum Latenten Raum gefolgt von 5 Upsamplings im Decoder zu der ursprünglichen Dimension der Eingabedaten: 256x384 – 128x192 – 64x96 – 32x48 – 16x24 – 8x12 – 16x24 – 32x48 – 64x96 – 128x192 – 256x384 .....	58
<i>Abbildung 6.11: Sample vom Datensatz "adidas_real" (preprocessed_adidas_real_109)</i> .....	59
Abbildung 6.12: Aktivierung nach dem ersten Encoding-Schritt, vor dem Downsampling.....	59
Abbildung 6.13: Aktivierungen des Latenten Raums .....	59
Abbildung 7.1: Original-Bild (links) und -Maske (rechts) als Sample von Datensatz "adidas_syn_dr" mit Original-Dimension von (256,384,4) = 393.216.....	69
<i>Abbildung 7.2: Rekonstruktion durch Modell "r_cae_fm16x24_dim12288" mit Latenter Dimension (16,24,32) = 12.288 und Reduktion von 96,88%</i> .....	70
Abbildung 7.3: Rekonstruktion durch Modell "r_cae_fm8x12_dim6144" mit Latenter Dimension (8,12,64) = 6.144 und Reduktion von 98,44% .....	70
Abbildung 7.4: Rekonstruktion durch Modell "r_cae_fm8x12_dim3072" mit Latenter Dimension (8,12,32) = 3.072 und Reduktion von 99,22% .....	70

---

Abbildung 7.5: Rekonstruktion durch Modell "r_cae_fm4x6_dim1536" mit Latenter Dimension (4,6,64) = 1.536 und Reduktion von 99,61% .....	70
<i>Abbildung 7.6: Rekonstruktion durch Modell "r_cae_fm4x6_dim768" mit Latenter Dimension (4,6,32) = 768 und Reduktion von 99,80% .....</i>	71
Abbildung 7.7: Rekonstruktion durch Modell "r_cae_fm2x3_dim384" mit Latenter Dimension (2,3,64) = 384 und Reduktion von 99,90% .....	71
Abbildung 7.8: Rekonstruktion durch Modell "r_cae_fm2x3_dim192" mit Latenter Dimension (2,3,32) = 192 und Reduktion von 99,95% .....	71
Abbildung 7.9: Die Grafiken zeigen die Rekonstruktionen von jeweils einem DR-, Simple- und Real-Sample, zur Veranschaulichung, wie diese von der Trainings-Reihenfolge betroffen sind. Original-Sample, darunter gefolgt von Dekodierungen durch „_dr_si_se_re“, „_re_se_si_dr“, „_si_dr_se_re“. Die rot hervorgehobenen Bilder, stellen die Samples dar, die zum Datensatz gehören, der als erstes im Modell-Training verwendet wurde.....	79
Abbildung 7.10: Vergleich der Rekonstruktionen eines Real-Samples (links) mit Modellen: r_cae_on_base_comb_fm8x12_dim6144_real3500 (mittig) & r_cae_on_base_comb_fm8x12_dim3072_real3500 (rechts).....	80
Abbildung 7.11: MDS-Visualisierung der Datensatz-Repräsentanten.....	86
Abbildung 7.12: Feature-Vektor-Level-Distanzmatrix der Durchschnitts-Feature-Vektoren als Datensatz-Repräsentanten; Niedrige Distanzen sind hellgelb und hohe Distanzen dunkelblau hervorgehoben. ....	87
Abbildung 7.13: Ergebnisse der Cluster-Analyse durch UMAP auf den Datensatz-Repräsentanten ...	94

---

## Tabellenverzeichnis

Tabelle 6.1: Zielgrößen des Latenten Raums und Reduktionsstärke in %.....	53
Tabelle 6.2: Performance-Scores der Basis-Datensätze einer Objekterkennungsaufgabe .....	61
Tabelle 6.3: Domänenzugehörigkeiten der Varianten-Datensätze und Basis-Datensätze als Domänenrepräsentanten .....	62
Tabelle 7.1: SSI-Scores der sequenziell trainierten Modelle mit absteigender Dimensionsgröße der Vorstudie .....	73
Tabelle 7.2: MSE-Scores der sequenziell trainierten Modelle mit absteigender Dimensionsgröße der Vorstudie .....	74
Tabelle 7.3: SSI-Scores der sequenziell trainierten Modelle CAE und R-CAE der Zielgrößen 3.072 und 6.144.....	75
Tabelle 7.4: MSE-Scores der sequenziell trainierten Modelle CAE und R-CAE der Zielgrößen 3.072 und 6.144.....	76
Tabelle 7.5: SSI-Scores der sequenziell trainierten Modelle mit unterschiedlicher Trainings-Reihenfolge .....	77
Tabelle 7.6: MSE-Scores der sequenziell trainierten Modelle mit unterschiedlicher Trainings-Reihenfolge .....	78
Tabelle 7.7: SSI-Scores der kombiniert trainierten Modelle mit absteigender Zielgröße von 6.144 bis 192.....	80
Tabelle 7.8: MSE-Scores der kombiniert trainierten Modelle mit absteigender Zielgröße von 6.144 bis 192 .....	81
Tabelle 7.9: SSI- und MSE-Scores der kombiniert trainierten Modelle mit abnehmendem Real-Anteil .....	81
Tabelle 7.10: SSI-Scores der Modelle mit abnehmender und wachsender Filteranzahl .....	83
Tabelle 7.11: MSE-Scores der Modelle mit abnehmender und wachsender Filteranzahl.....	83
Tabelle 7.12: Ausschnitt der normalisierten Distanzmatrix der Basis-Datensätze.....	88
Tabelle 7.13: Natürliche Domänenbildung ohne Referenzwerte mit Schwellenwert von $\alpha = 0,1$ .....	89

---

---

Tabelle 7.14: Vergleich der Referenzwerte und der Rankings mit verschiedenen Bewertungskriterien .....	91
Tabelle 7.15: SVM-Ergebnisse, Vorhersagen für die Feature-Vektoren der Varianten-Datensätze.....	93
Auswahlverfahren 1: Referenzwerte der Objekterkennungsaufgabe .....	121
Auswahlverfahren 2: Domänenbildung nach Referenzwerte r_cae_on_base_comb_fm8x12_dim3072_real3500_norm .....	121
Auswahlverfahren 3: r_cae_on_base_comb_fm8x12_dim3072_real3500_norm mit Real-Nähe.....	122
Auswahlverfahren 4: r_cae_on_base_comb_fm8x12_dim3072_real3500_no_norm mit Real-Nähe	122
Auswahlverfahren 5: r_cae_on_base_comb_fm8x12_dim3072_real3500_norm mit Domänen und Real-Nähe.....	123
Auswahlverfahren 6: r_cae_on_base_comb_fm8x12_dim3072_real3500_no_norm mit Domänen und Real-Nähe.....	123
Auswahlverfahren 7: r_cae_on_base_comb_fm8x12_dim3072_real3500_norm mit Importance Score .....	124
Auswahlverfahren 8: r_cae_on_base_comb_fm8x12_dim3072_real3500_no_norm mit Importance Score.....	124
Auswahlverfahren 9: r_cae_on_base_comb_fm8x12_dim3072_real3500_norm mit Domänen und Importance Score.....	125
Auswahlverfahren 10: r_cae_on_base_comb_fm8x12_dim3072_real3500_no_norm mit Domänen und Importance Score.....	125
Auswahlverfahren 11: r_cae_on_base_comb_fm8x12_dim6144_real3500_norm mit Real-Nähe.....	126
Auswahlverfahren 12: r_cae_on_base_comb_fm8x12_dim6144_real3500_no_norm mit Real-Nähe	126
Auswahlverfahren 13: r_cae_on_base_comb_fm8x12_dim6144_real3500_norm mit Domänen und Real-Nähe.....	127
Auswahlverfahren 14: r_cae_on_base_comb_fm8x12_dim6144_real3500_no_norm mit Domänen und Real-Nähe.....	127

---

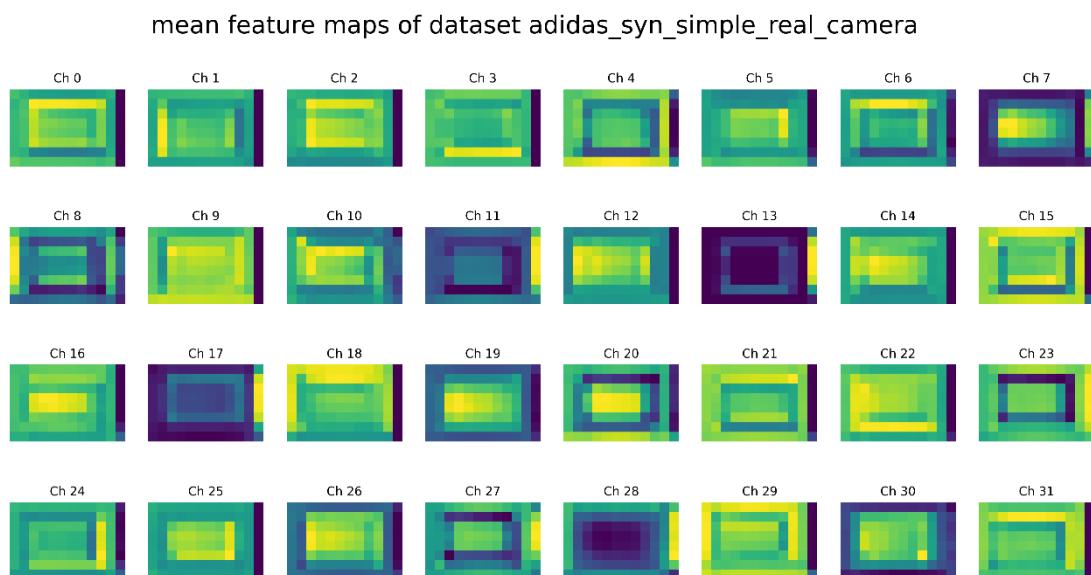
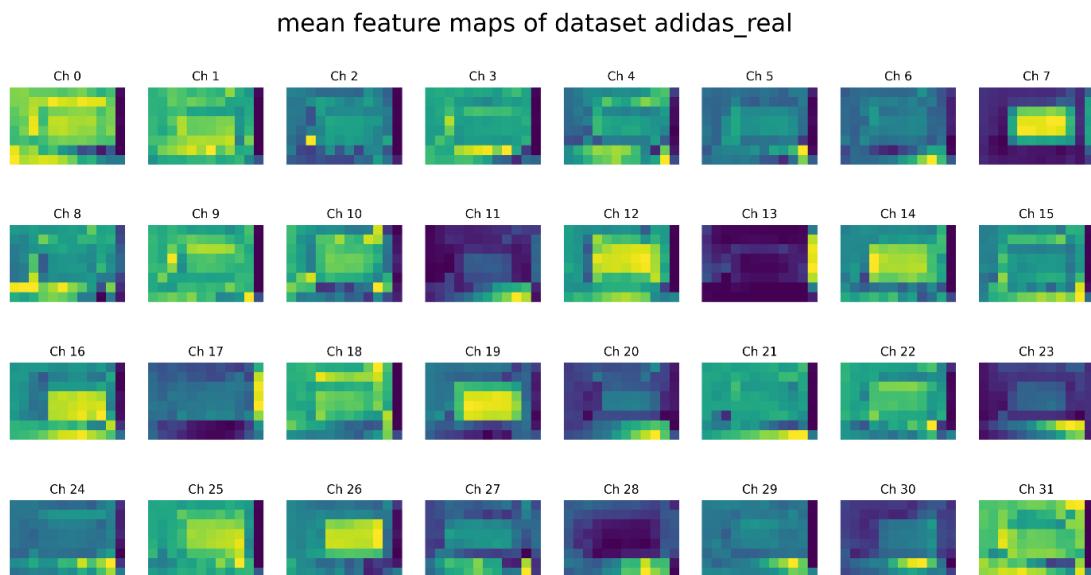
Auswahlverfahren 15: r_cae_on_base_comb_fm8x12_dim6144_real3500_norm mit Importance Score.....	128
Auswahlverfahren 16: r_cae_on_base_comb_fm8x12_dim6144_real3500_no_norm mit Importance Score.....	128
Auswahlverfahren 17: r_cae_on_base_comb_fm8x12_dim6144_real3500_norm mit Domänen und Importance Score.....	129
Auswahlverfahren 18: r_cae_on_base_comb_fm8x12_dim6144_real3500_no_norm mit Domänen und Importance Score.....	129
Auswahlverfahren 19: r_cae_on_base_comb_fm4x6_dim1536_real3500_norm mit Real-Nähe.....	130
Auswahlverfahren 20: r_cae_on_base_comb_fm4x6_dim1536_real3500_no_norm mit Real-Nähe.	130
Auswahlverfahren 21: r_cae_on_base_comb_fm4x6_dim1536_real3500_norm mit Domänen und Real-Nähe.....	131
Auswahlverfahren 22: r_cae_on_base_comb_fm4x6_dim1536_real3500_no_norm mit Domänen und Real-Nähe.....	131
Auswahlverfahren 23: r_cae_on_base_comb_fm4x6_dim1536_real3500_norm mit Importance Score .....	132
Auswahlverfahren 24: r_cae_on_base_comb_fm4x6_dim1536_real3500_no_norm mit Importance Score.....	132
Auswahlverfahren 25: r_cae_on_base_comb_fm4x6_dim1536_real3500_norm mit Domänen und Importance Score.....	133
Auswahlverfahren 26: r_cae_on_base_comb_fm4x6_dim1536_real3500_no_norm mit Domänen und Importance Score.....	133

---

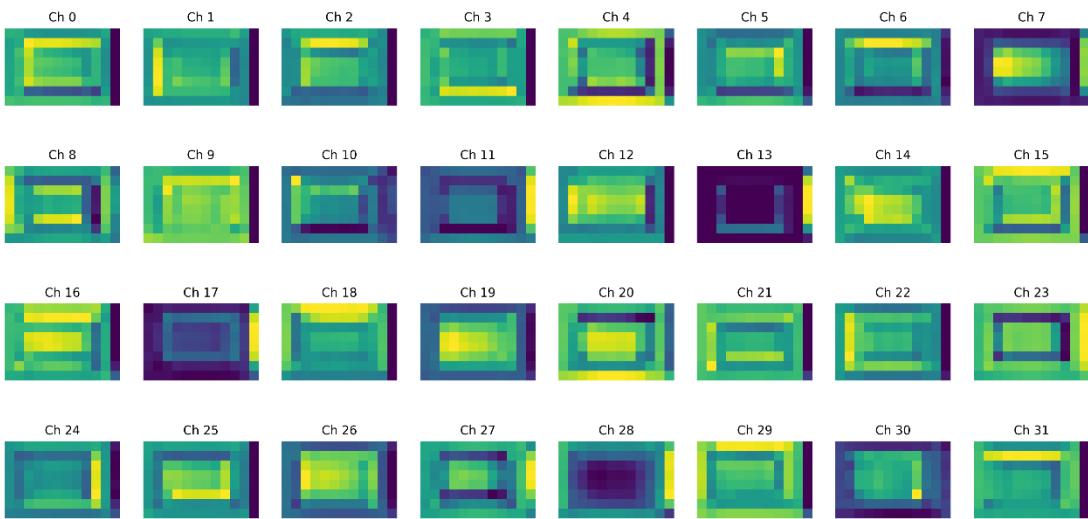
## Anhang

### A Latenter Raum mit „r\_cae\_on\_comb\_fm8x12\_dim3072\_real3500“

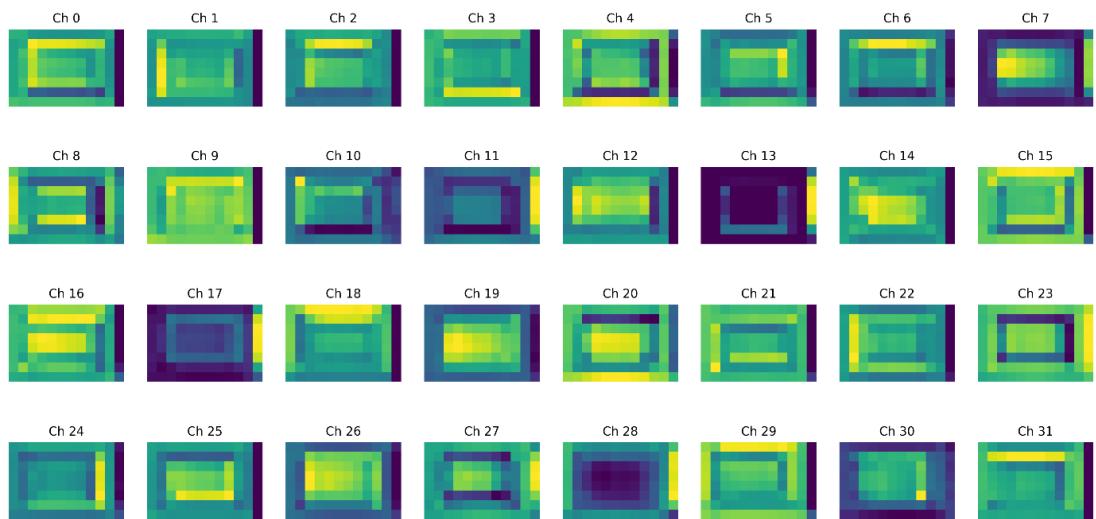
#### A.1 Durchschnitts-Feature-Vektoren



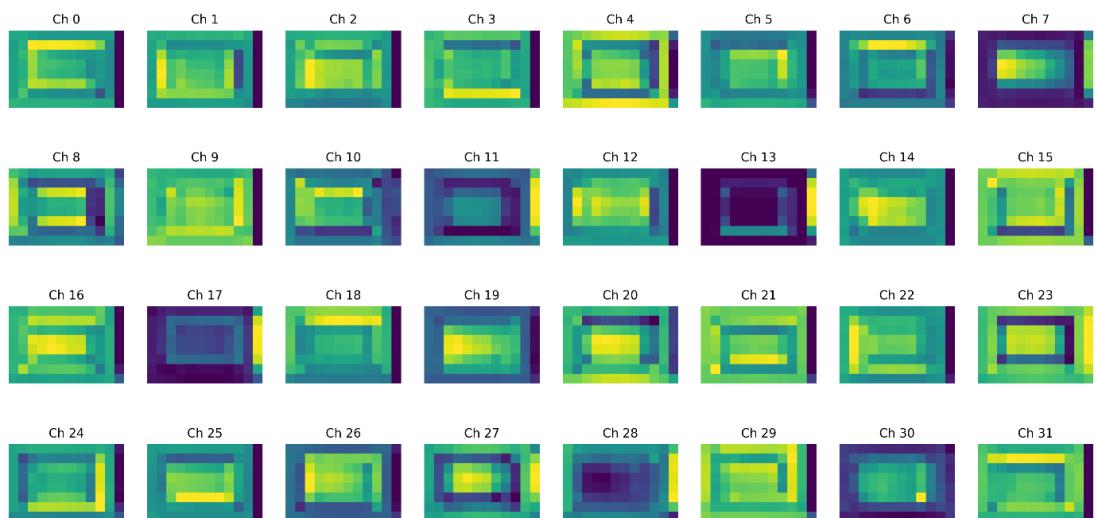
mean feature maps of dataset adidas\_syn\_simple



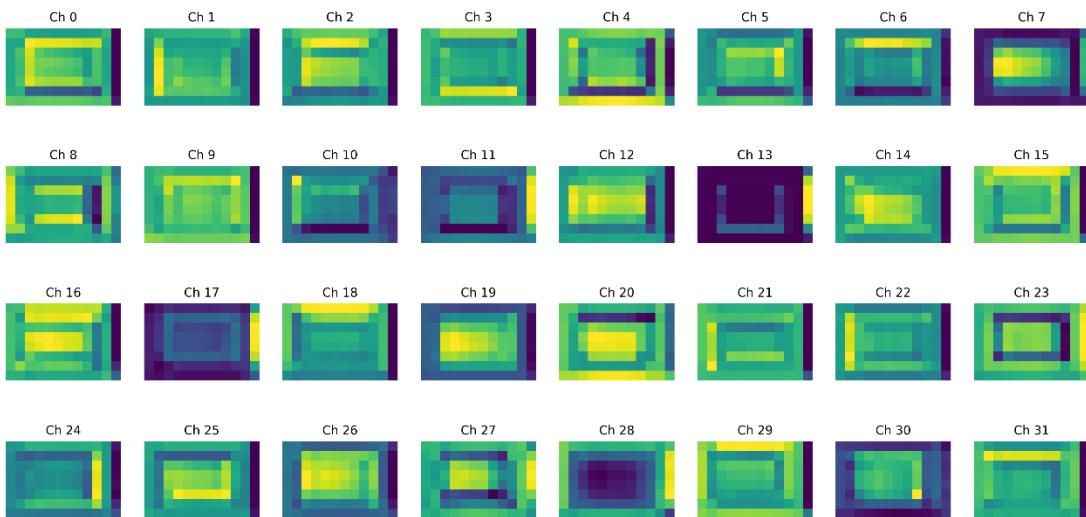
mean feature maps of dataset adidas\_syn\_simple\_real\_texture



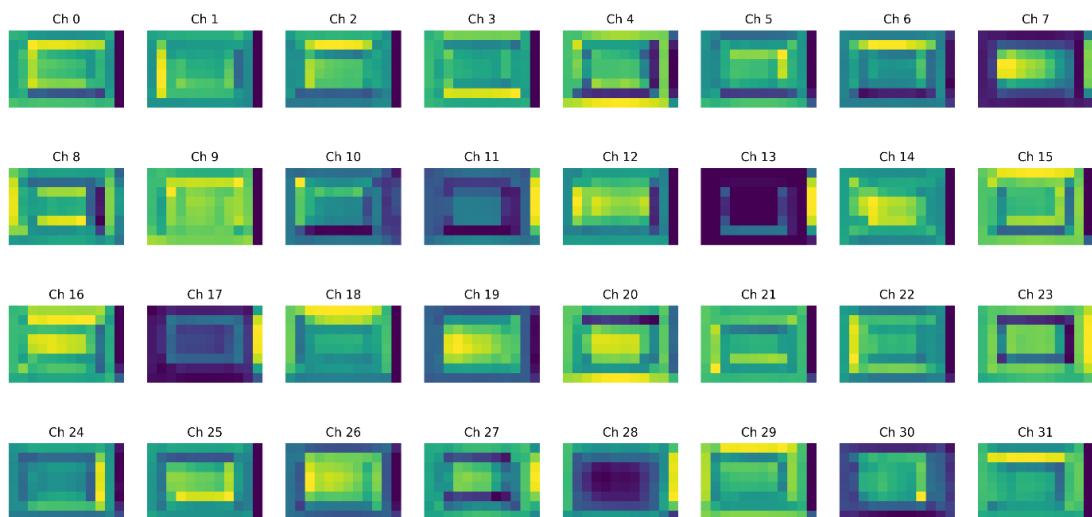
mean feature maps of dataset adidas\_syn\_simple\_random\_texture



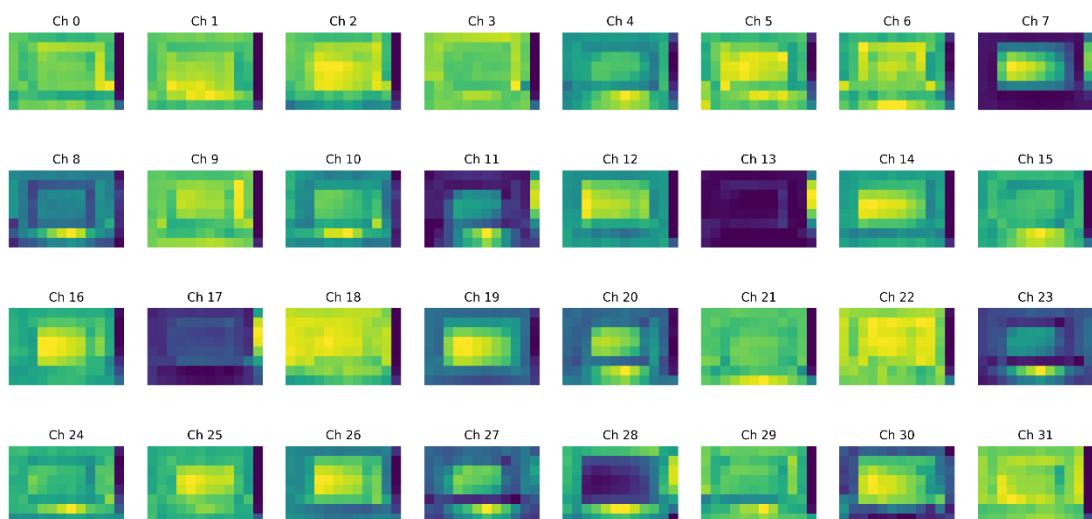
mean feature maps of dataset adidas\_syn\_simple\_random\_light



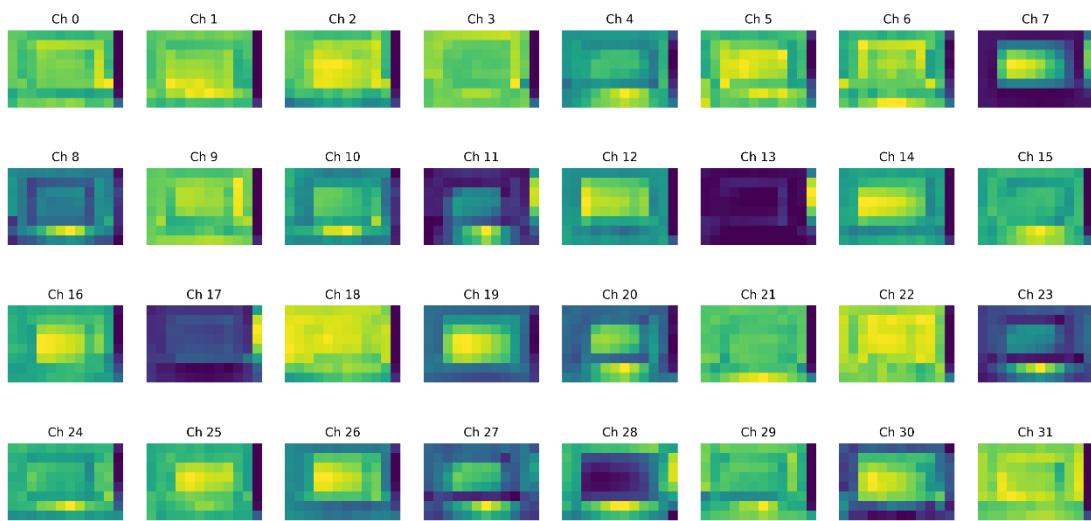
mean feature maps of dataset adidas\_syn\_simple\_real\_light



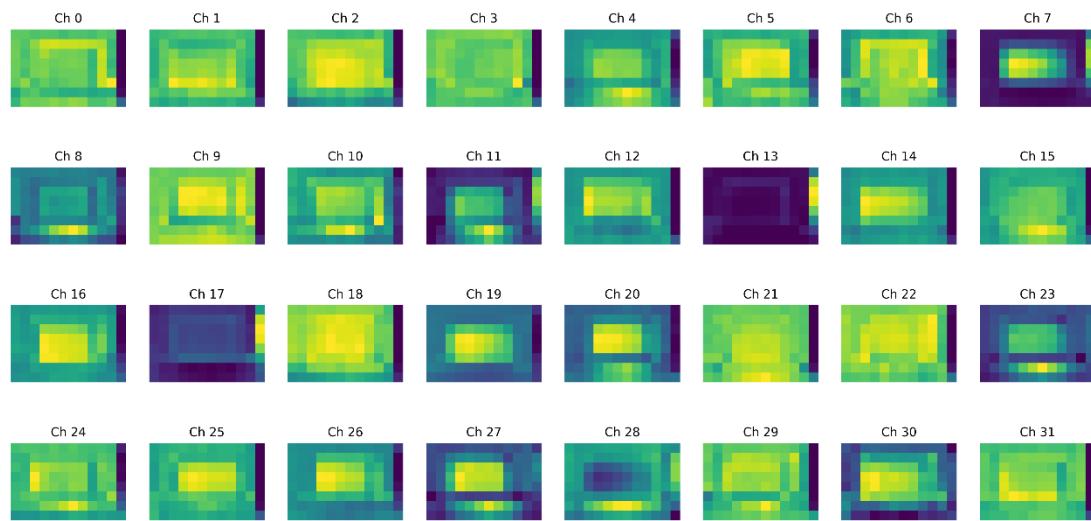
mean feature maps of dataset adidas\_syn\_se



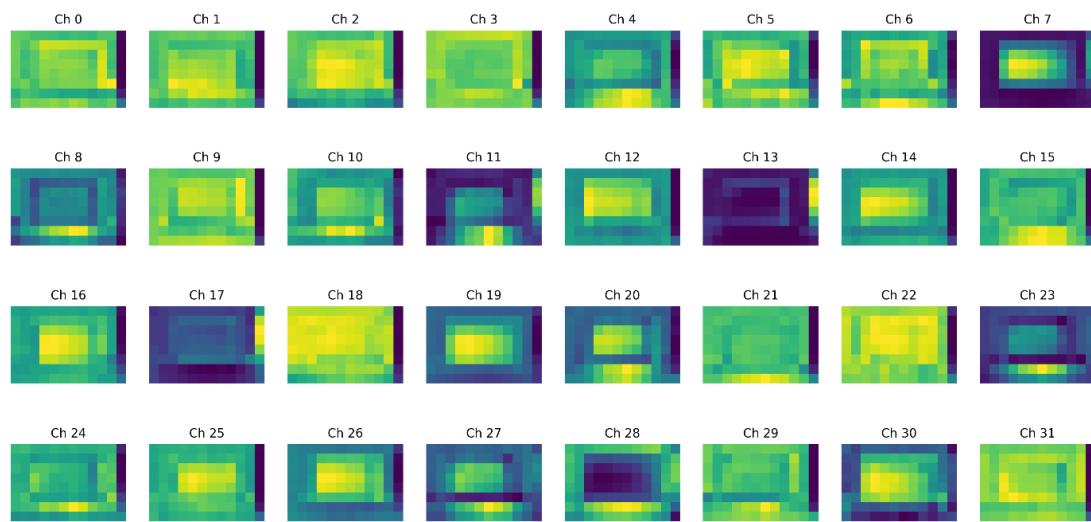
mean feature maps of dataset adidas\_syn\_se\_color\_texture



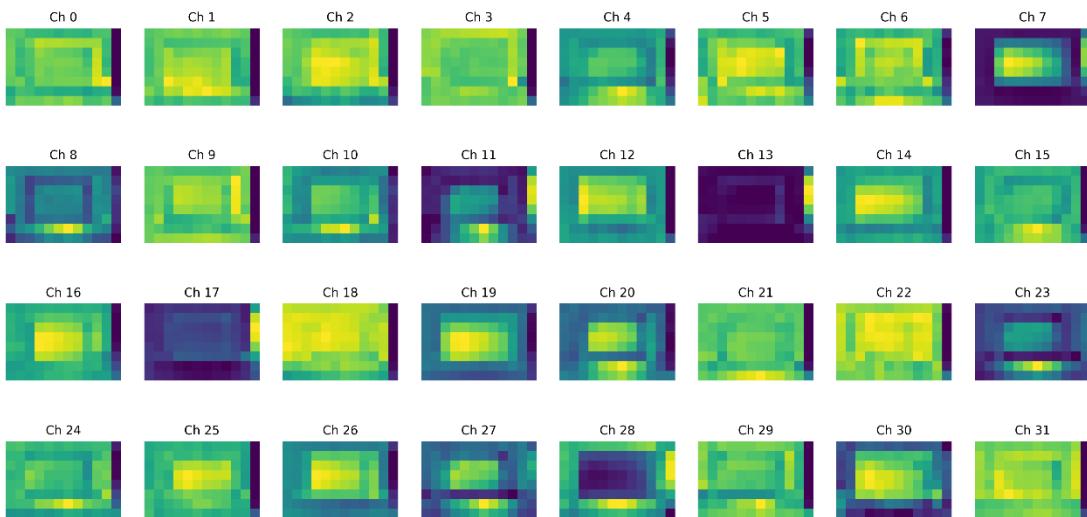
mean feature maps of dataset adidas\_syn\_se\_random\_texture



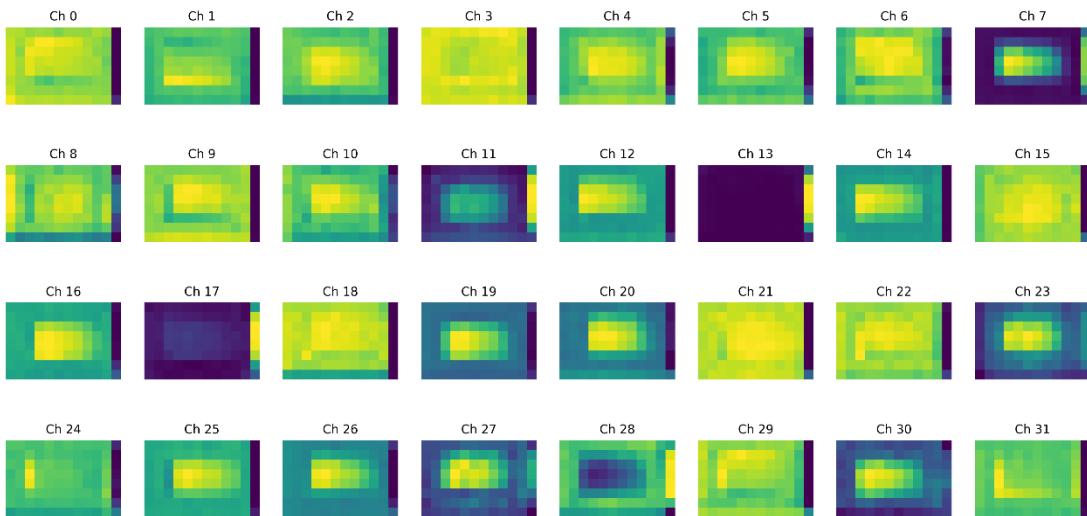
mean feature maps of dataset adidas\_syn\_se\_random\_light



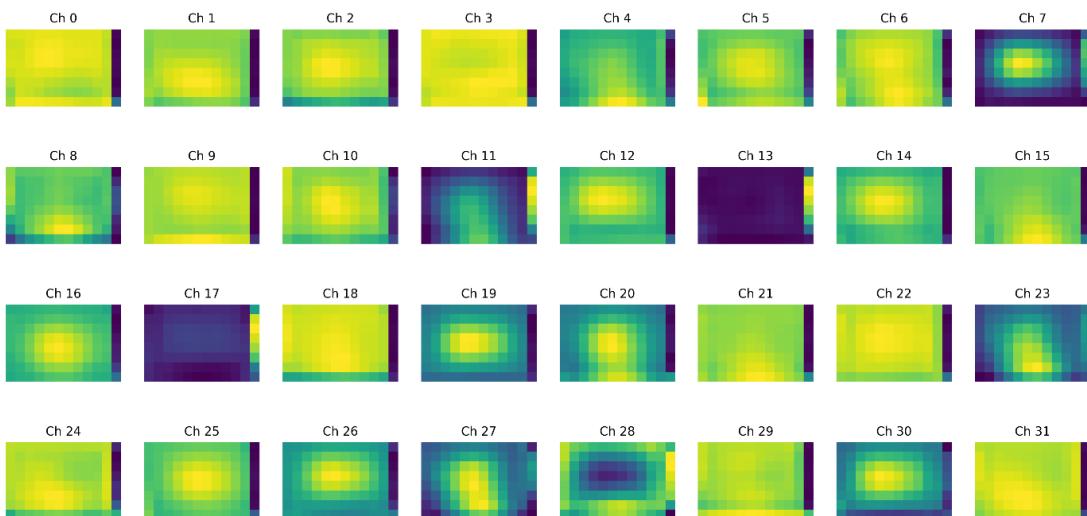
mean feature maps of dataset adidas\_syn\_se\_static\_light



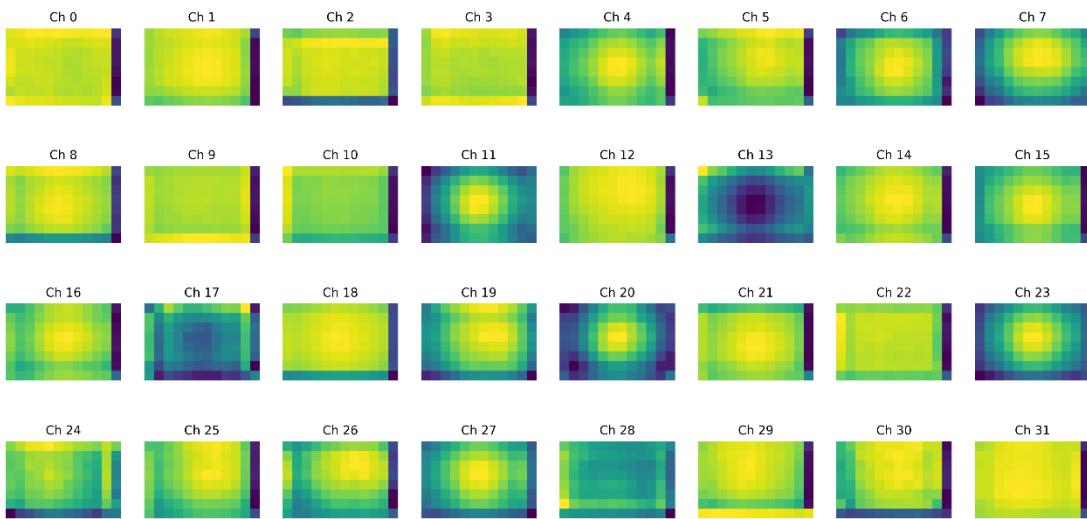
mean feature maps of dataset adidas\_syn\_se\_no\_lc\_table



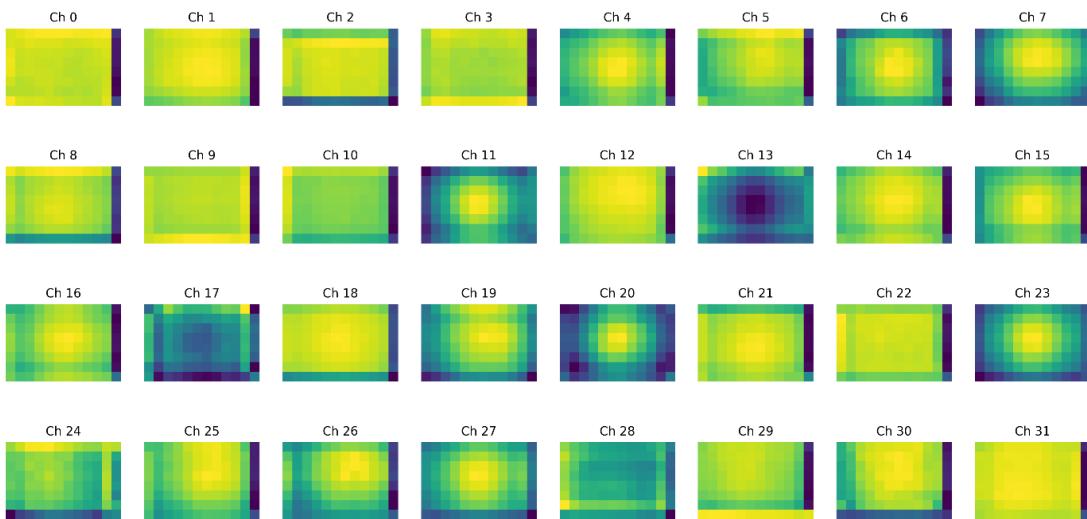
mean feature maps of dataset adidas\_syn\_se\_random\_camera



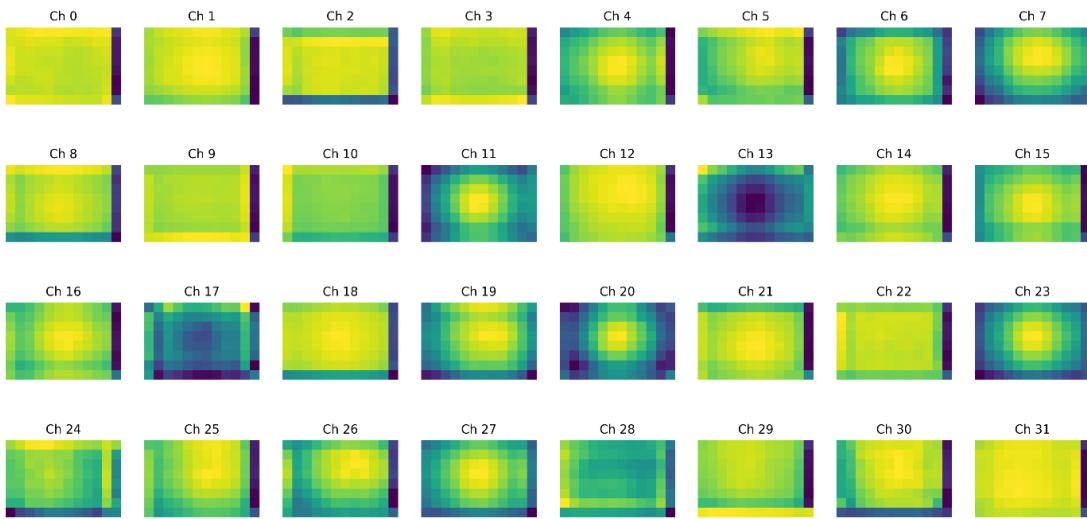
mean feature maps of dataset adidas\_syn\_dr



mean feature maps of dataset adidas\_syn\_dr\_real\_texture



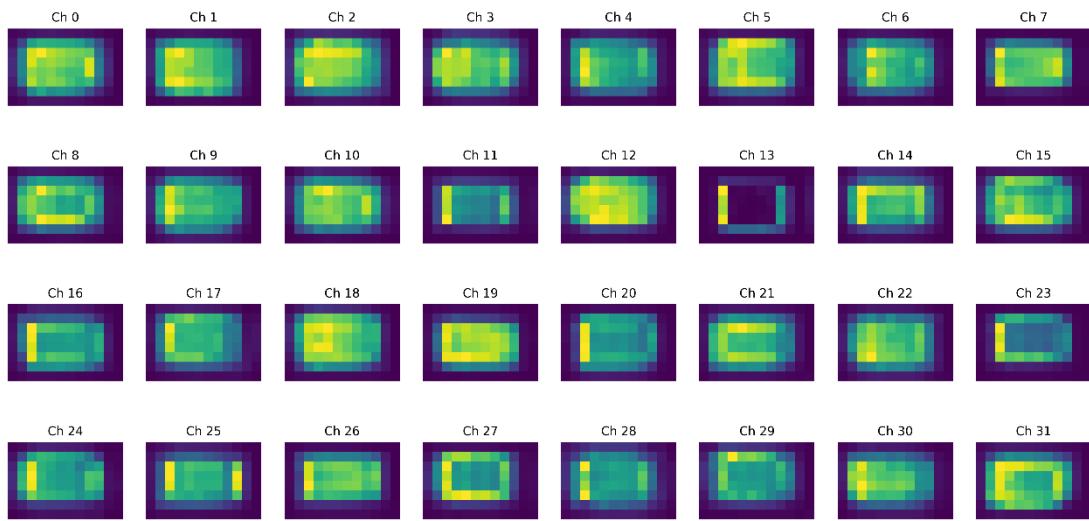
mean feature maps of dataset adidas\_syn\_dr\_color\_texture



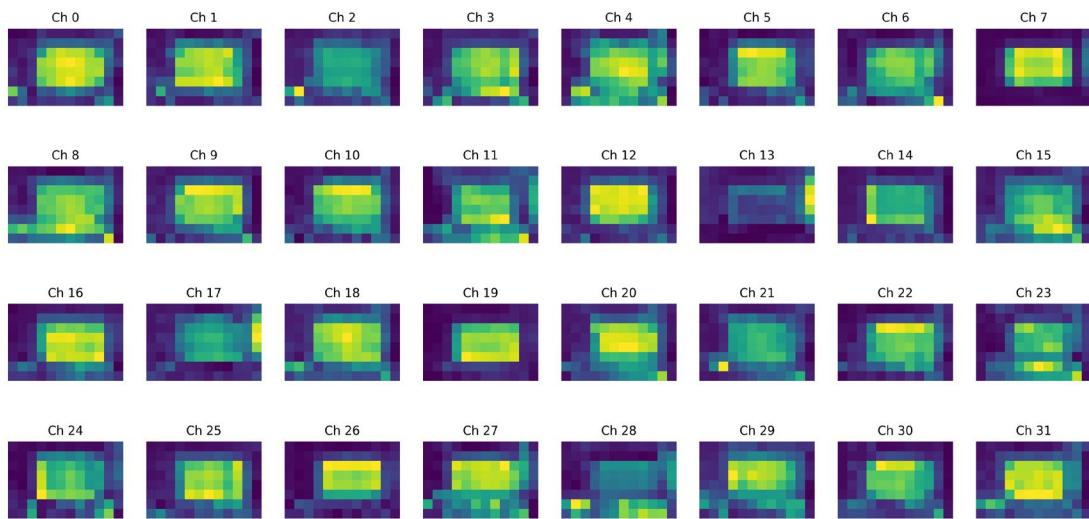
---

## A.2 Std-Dev-Feature-Vektoren

std\_dev feature maps of dataset adidas\_syn\_simple

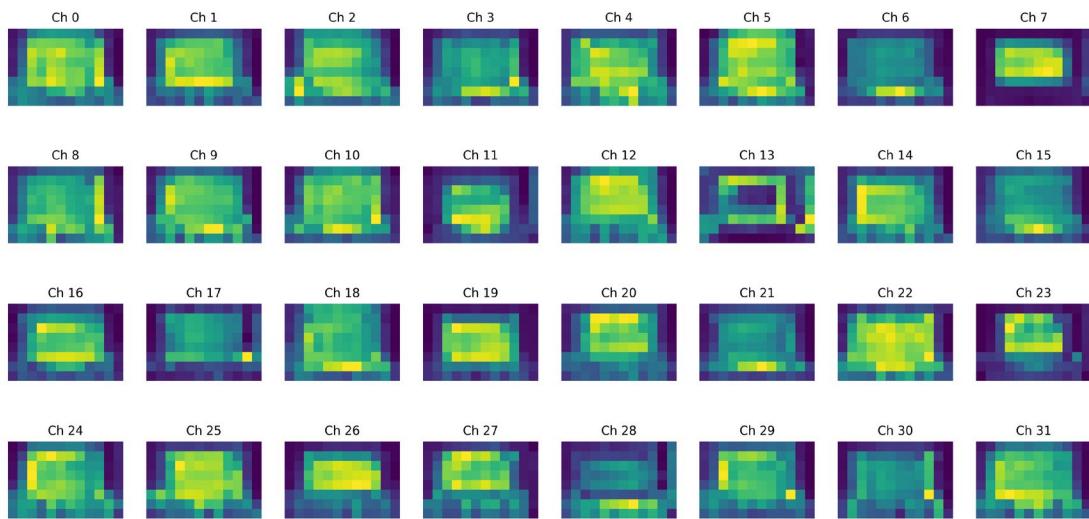


std\_dev feature maps of dataset adidas\_real

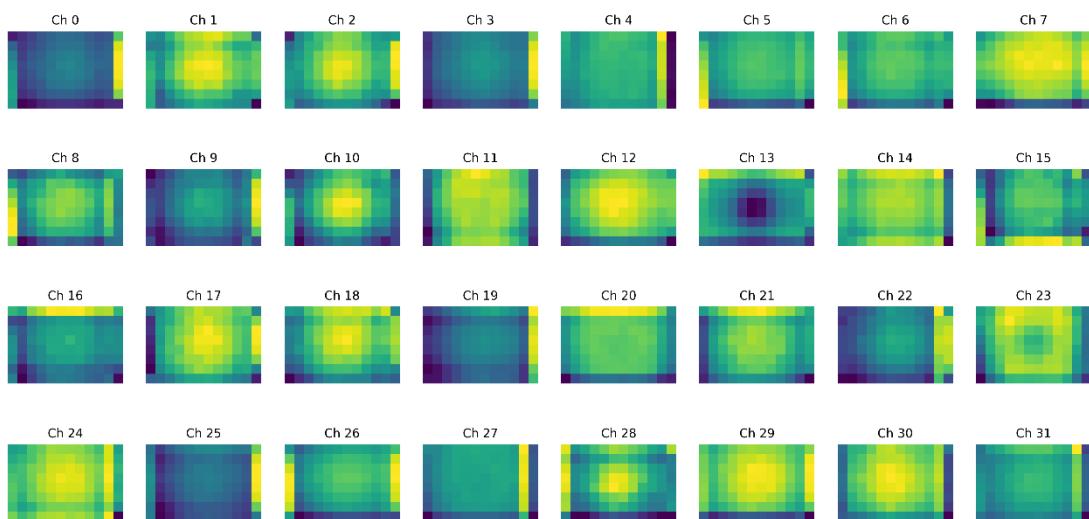


---

std\_dev feature maps of dataset adidas\_syn\_se



std\_dev feature maps of dataset adidas\_syn\_dr



## B Auswahlverfahren: r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500

### B.1 Referenzwerte der Objekterkennungsaufgabe

	adidas_real	adidas_syn_se_random_light	adidas_syn_se_random_camera	adidas_syn_se_random_texture	adidas_syn_simple_real_camera	adidas_syn_simple	adidas_syn_no_lc_table	adidas_syn_dr_real_texture	adidas_syn_dr
mAP	88,99	78,33	75,53	74,96	53,08	36,49	28,34	27,13	22,26
std_dev	0,37	0,74	1,15	1,14	3,48	2,40	0,79	5,79	3,68

Auswahlverfahren 1: Referenzwerte der Objekterkennungsaufgabe

### B.2 Domänenbildung nach Basis-Referenzwerte

Domain: real	adidas_real	adidas_syn_se	adidas_syn_simple	adidas_syn_dr
adidas_real	0,0000	0,7101	0,8896	0,9331

Domain: syn_dr	adidas_real	adidas_syn_se	adidas_syn_simple	adidas_syn_dr
adidas_syn_dr_color_texture	0.9236	0.8421	0.9931	0.0342
adidas_syn_dr_real_texture	0.9285	0.8511	0.9982	0.0525
adidas_syn_dr	0.9331	0.8411	0.9972	0.0000

Domain: syn_se	adidas_real	adidas_syn_se	adidas_syn_simple	adidas_syn_dr
didas_syn_se_random_light	0.6869	0.1547	0.7469	0.7976
adidas_syn_se_color_texture	0.7062	0,0149	0.7050	0.8497
adidas_syn_se	0.7107	0.0000	0.6974	0.8411
adidas_syn_se_static_light	0.7126	0.0164	0.6894	0.8430
adidas_syn_se_random_texture	0,7402	0.2682	0.7528	0.8328
adidas_syn_se_random_camera	0.7533	0.4192	0.7643	0.6494
adidas_syn_se_no_lc_table	0,8013	0.4096	0.7369	0.7900

Domain: syn_simple	adidas_real	adidas_syn_se	adidas_syn_simple	adidas_syn_dr
adidas_syn_simple_real_camera	0,8402	0,6107	0,1750	0,9888
adidas_syn_simple_random_light	0,8453	0,6467	0,1188	0,9738
adidas_syn_simple_random_texture	0,8455	0,6354	0,1856	0,9758
adidas_syn_simple_real_light	0,8603	0,6610	0,0105	0,9905
adidas_syn_simple	0,8618	0,6631	0,0000	0,9889
adidas_syn_simple_real_texture	0,8622	0,6637	0,0000	0,9873

Auswahlverfahren 2: Domänenbildung nach Referenzwerte  
r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500\_norm

---

### B.3 Ranking mit Real-Nähe

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	313,29
2	adidas_syn_se_random_light	adidas_syn_se	2	1	0,6869	89,83
3	adidas_syn_se_color_texture	adidas_syn_se	2	2	0,7062	77,49
4	adidas_syn_se	adidas_syn_se	2	3	0,7107	-313,29
5	adidas_syn_se_static_light	adidas_syn_se	2	4	0,7126	94,54
6	adidas_syn_se_random_texture	adidas_syn_se	2	5	0,7402	-19,33
7	adidas_syn_se_random_camera	adidas_syn_se	2	6	0,7533	53,57
8	adidas_syn_se_no_lc_table	adidas_syn_se	2	7	0,8013	89,43
9	adidas_syn_simple_real_camera	adidas_syn_simple	3	1	0,8357	98,39
10	adidas_syn_simple_random_texture	adidas_syn_simple	3	2	0,8594	32,85
11	adidas_syn_simple_random_light	adidas_syn_simple	3	3	0,8743	87,29
12	adidas_syn_simple_real_light	adidas_syn_simple	3	4	0,8871	74,18
13	adidas_syn_simple	adidas_syn_simple	3	5	0,8896	80,80
14	adidas_syn_simple_real_texture	adidas_syn_simple	3	6	0,8917	75,82
15	adidas_syn_dr_color_texture	adidas_syn_dr	4	1	0,9236	78,89
16	adidas_syn_dr_real_texture	adidas_syn_dr	4	2	0,9285	-294,69
17	adidas_syn_dr	adidas_syn_dr	4	3	0,9331	-301,86

Auswahlverfahren 3: r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500\_norm  
mit Real-Nähe

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	129,67
2	adidas_syn_se_random_light	adidas_syn_se	2	1	0,5643	24,74
3	adidas_syn_se_random_camera	adidas_syn_se	2	2	0,5782	22,34
4	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,5966	-129,67
5	adidas_syn_se	adidas_syn_se	2	4	0,602	26,75
6	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6073	-14,66
7	adidas_syn_se_no_lc_table	adidas_syn_se	2	6	0,6633	18,63
8	adidas_syn_se_random_texture	adidas_syn_se	2	7	0,6712	24,36
9	adidas_syn_dr_color_texture	adidas_syn_dr	3	1	0,7661	28,99
10	adidas_syn_dr_real_texture	adidas_syn_dr	3	2	0,7686	8,47
11	adidas_syn_dr	adidas_syn_dr	3	3	0,7711	25,35
12	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8031	27,49
13	adidas_syn_simple_random_light	adidas_syn_simple	4	2	0,8655	20,96
14	adidas_syn_simple_random_texture	adidas_syn_simple	4	3	0,874	21,25
15	adidas_syn_simple_real_light	adidas_syn_simple	4	4	0,8881	22,48
16	adidas_syn_simple	adidas_syn_simple	4	5	0,8941	-126,95
17	adidas_syn_simple_real_texture	adidas_syn_simple	4	6	0,8964	-127,67

Auswahlverfahren 4: r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500\_no\_norm  
mit Real-Nähe

## B.4 Domain-Based-Ranking mit Real-Nähe

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	313,29
2	adidas_syn_se_random_light	adidas_syn_se	2	1	0,6869	89,83
3	adidas_syn_se_color_texture	adidas_syn_se	2	2	0,7062	77,49
4	adidas_syn_se	adidas_syn_se	2	3	0,7107	-313,29
5	adidas_syn_se_static_light	adidas_syn_se	2	4	0,7126	94,54
6	adidas_syn_se_random_texture	adidas_syn_se	2	5	0,7402	-19,33
7	adidas_syn_se_random_camera	adidas_syn_se	2	6	0,7533	53,57
8	adidas_syn_se_no_lc_table	adidas_syn_se	2	7	0,8013	89,43
9	adidas_syn_simple_real_camera	adidas_syn_simple	3	1	0,8357	98,39
10	adidas_syn_simple_random_texture	adidas_syn_simple	3	2	0,8594	32,85
11	adidas_syn_simple_random_light	adidas_syn_simple	3	3	0,8743	87,29
12	adidas_syn_simple_real_light	adidas_syn_simple	3	4	0,8871	74,18
13	adidas_syn_simple	adidas_syn_simple	3	5	0,8896	80,80
14	adidas_syn_simple_real_texture	adidas_syn_simple	3	6	0,8917	75,82
15	adidas_syn_dr_color_texture	adidas_syn_dr	4	1	0,9236	78,89
16	adidas_syn_dr_real_texture	adidas_syn_dr	4	2	0,9285	-294,69
17	adidas_syn_dr	adidas_syn_dr	4	3	0,9331	-301,86

Auswahlverfahren 5: r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500\_norm  
mit Domänen und Real-Nähe

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	129,67
2	adidas_syn_se_random_light	adidas_syn_se	2	1	0,5643	24,74
3	adidas_syn_se_random_camera	adidas_syn_se	2	2	0,5782	22,34
4	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,5966	-129,67
5	adidas_syn_se	adidas_syn_se	2	4	0,602	26,75
6	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6073	-14,66
7	adidas_syn_se_no_lc_table	adidas_syn_se	2	6	0,6633	18,63
8	adidas_syn_se_random_texture	adidas_syn_se	2	7	0,6712	24,36
9	adidas_syn_dr_color_texture	adidas_syn_dr	3	1	0,7661	28,99
10	adidas_syn_dr_real_texture	adidas_syn_dr	3	2	0,7686	8,47
11	adidas_syn_dr	adidas_syn_dr	3	3	0,7711	25,35
12	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8031	27,49
13	adidas_syn_simple_random_light	adidas_syn_simple	4	2	0,8655	20,96
14	adidas_syn_simple_random_texture	adidas_syn_simple	4	3	0,874	21,25
15	adidas_syn_simple_real_light	adidas_syn_simple	4	4	0,8881	22,48
16	adidas_syn_simple	adidas_syn_simple	4	5	0,8941	-126,95
17	adidas_syn_simple_real_texture	adidas_syn_simple	4	6	0,8964	-127,67

Auswahlverfahren 6: r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500\_no\_norm  
mit Domänen und Real-Nähe

---

## B.5 Ranking mit Importance Score

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	313,29
2	adidas_syn_simple_real_camera	adidas_syn_simple	3	1	0,8357	98,39
3	adidas_syn_se_static_light	adidas_syn_se	2	4	0,7126	94,54
4	adidas_syn_se_random_light	adidas_syn_se	2	1	0,6869	89,83
5	adidas_syn_se_no_lc_table	adidas_syn_se	2	7	0,8013	89,43
6	adidas_syn_simple_random_light	adidas_syn_simple	3	3	0,8743	87,29
7	adidas_syn_simple	adidas_syn_simple	3	5	0,8896	80,80
8	adidas_syn_dr_color_texture	adidas_syn_dr	4	1	0,9236	78,89
9	adidas_syn_se_color_texture	adidas_syn_se	2	2	0,7062	77,49
10	adidas_syn_simple_real_texture	adidas_syn_simple	3	6	0,8917	75,82
11	adidas_syn_simple_real_light	adidas_syn_simple	3	4	0,8871	74,18
12	adidas_syn_se_random_camera	adidas_syn_se	2	6	0,7533	53,57
13	adidas_syn_simple_random_texture	adidas_syn_simple	3	2	0,8594	32,85
14	adidas_syn_se_random_texture	adidas_syn_se	2	5	0,7402	-19,33
15	adidas_syn_dr_real_texture	adidas_syn_dr	4	2	0,9285	-294,69
16	adidas_syn_dr	adidas_syn_dr	4	3	0,9331	-301,86
17	adidas_syn_se	adidas_syn_se	2	3	0,7107	-313,29

Auswahlverfahren 7: r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500\_norm  
mit Importance Score

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	129,67
2	adidas_syn_dr_color_texture	adidas_syn_dr	3	1	0,7661	28,99
3	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8031	27,49
4	adidas_syn_se	adidas_syn_se	2	4	0,602	26,75
5	adidas_syn_dr	adidas_syn_dr	3	3	0,7711	25,35
6	adidas_syn_se_random_light	adidas_syn_se	2	1	0,5643	24,74
7	adidas_syn_se_random_texture	adidas_syn_se	2	7	0,6712	24,36
8	adidas_syn_simple_real_light	adidas_syn_simple	4	4	0,8881	22,48
9	adidas_syn_se_random_camera	adidas_syn_se	2	2	0,5782	22,34
10	adidas_syn_simple_random_texture	adidas_syn_simple	4	3	0,874	21,25
11	adidas_syn_simple_random_light	adidas_syn_simple	4	2	0,8655	20,96
12	adidas_syn_se_no_lc_table	adidas_syn_se	2	6	0,6633	18,63
13	adidas_syn_dr_real_texture	adidas_syn_dr	3	2	0,7686	8,47
14	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6073	-14,66
15	adidas_syn_simple	adidas_syn_simple	4	5	0,8941	-126,95
16	adidas_syn_simple_real_texture	adidas_syn_simple	4	6	0,8964	-127,67
17	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,5966	-129,67

Auswahlverfahren 8: r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500\_no\_norm  
mit Importance Score

## B.6 Domain-Based-Ranking mit Importance Score

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	313,29
2	adidas_syn_se_static_light	adidas_syn_se	2	4	0,7126	94,54
3	adidas_syn_se_random_light	adidas_syn_se	2	1	0,6869	89,83
4	adidas_syn_se_no_lc_table	adidas_syn_se	2	7	0,8013	89,43
5	adidas_syn_se_color_texture	adidas_syn_se	2	2	0,7062	77,49
6	adidas_syn_se_random_camera	adidas_syn_se	2	6	0,7533	53,57
7	adidas_syn_se_random_texture	adidas_syn_se	2	5	0,7402	-19,33
8	adidas_syn_se	adidas_syn_se	2	3	0,7107	-313,29
9	adidas_syn_simple_real_camera	adidas_syn_simple	3	1	0,8357	98,39
10	adidas_syn_simple_random_light	adidas_syn_simple	3	3	0,8743	87,29
11	adidas_syn_simple	adidas_syn_simple	3	5	0,8896	80,80
12	adidas_syn_simple_real_texture	adidas_syn_simple	3	6	0,8917	75,82
13	adidas_syn_simple_real_light	adidas_syn_simple	3	4	0,8871	74,18
14	adidas_syn_simple_random_texture	adidas_syn_simple	3	2	0,8594	32,85
15	adidas_syn_dr_color_texture	adidas_syn_dr	4	1	0,9236	78,89
16	adidas_syn_dr_real_texture	adidas_syn_dr	4	2	0,9285	-294,69
17	adidas_syn_dr	adidas_syn_dr	4	3	0,9331	-301,86

Auswahlverfahren 9: r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500\_norm  
mit Domänen und Importance Score

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	129,67
2	adidas_syn_se	adidas_syn_se	2	4	0,602	26,75
3	adidas_syn_se_random_light	adidas_syn_se	2	1	0,5643	24,74
4	adidas_syn_se_random_texture	adidas_syn_se	2	7	0,6712	24,36
5	adidas_syn_se_random_camera	adidas_syn_se	2	2	0,5782	22,34
6	adidas_syn_se_no_lc_table	adidas_syn_se	2	6	0,6633	18,63
7	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6073	-14,66
8	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,5966	-129,67
9	adidas_syn_dr_color_texture	adidas_syn_dr	3	1	0,7661	28,99
10	adidas_syn_dr	adidas_syn_dr	3	3	0,7711	25,35
11	adidas_syn_dr_real_texture	adidas_syn_dr	3	2	0,7686	8,47
12	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8031	27,49
13	adidas_syn_simple_real_light	adidas_syn_simple	4	4	0,8881	22,48
14	adidas_syn_simple_random_texture	adidas_syn_simple	4	3	0,874	21,25
15	adidas_syn_simple_random_light	adidas_syn_simple	4	2	0,8655	20,96
16	adidas_syn_simple	adidas_syn_simple	4	5	0,8941	-126,95
17	adidas_syn_simple_real_texture	adidas_syn_simple	4	6	0,8964	-127,67

Auswahlverfahren 10: r\_cae\_on\_base\_comb\_fm8x12\_dim3072\_real3500\_no\_norm  
mit Domänen und Importance Score

---

## C Auswahlverfahren: r\_cae\_on\_base\_comb\_fm8x12\_dim6144\_real3500

### C.1 Ranking mit Real-Nähe

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	258,76
2	adidas_syn_se_random_camera	adidas_syn_se	2	1	0,5889	45,89
3	adidas_syn_se_random_light	adidas_syn_se	2	2	0,6018	60,96
4	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,6344	-258,76
5	adidas_syn_se	adidas_syn_se	2	4	0,6409	56,63
6	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6412	10,20
7	adidas_syn_se_random_texture	adidas_syn_se	2	6	0,659	32,09
8	adidas_syn_se_no_lc_table	adidas_syn_se	2	7	0,6646	44,00
9	adidas_syn_dr	adidas_syn_dr	3	1	0,793	58,34
10	adidas_syn_dr_color_texture	adidas_syn_dr	3	2	0,7994	14,30
11	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8065	75,02
12	adidas_syn_dr_real_texture	adidas_syn_dr	3	3	0,8091	56,14
13	adidas_syn_simple_random_texture	adidas_syn_simple	4	2	0,8372	55,43
14	adidas_syn_simple_random_light	adidas_syn_simple	4	3	0,8502	58,15
15	adidas_syn_simple_real_light	adidas_syn_simple	4	4	0,8647	64,13
16	adidas_syn_simple	adidas_syn_simple	4	5	0,8681	-236,42
17	adidas_syn_simple_real_texture	adidas_syn_simple	4	6	0,8726	-242,90

Auswahlverfahren 11: r\_cae\_on\_base\_comb\_fm8x12\_dim6144\_real3500\_norm mit Real-Nähe

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	152,70
2	adidas_syn_se_random_light	adidas_syn_se	2	1	0,5756	37,81
3	adidas_syn_se_random_camera	adidas_syn_se	2	2	0,5865	26,67
4	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,621	-152,70
5	adidas_syn_se	adidas_syn_se	2	4	0,6244	47,58
6	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6283	13,50
7	adidas_syn_se_no_lc_table	adidas_syn_se	2	6	0,6866	27,57
8	adidas_syn_se_random_texture	adidas_syn_se	2	7	0,7354	36,09
9	adidas_syn_dr_color_texture	adidas_syn_dr	3	1	0,7713	45,22
10	adidas_syn_dr_real_texture	adidas_syn_dr	3	2	0,7726	14,21
11	adidas_syn_dr	adidas_syn_dr	3	3	0,7782	35,53
12	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8316	25,40
13	adidas_syn_simple_random_light	adidas_syn_simple	4	2	0,9196	28,60
14	adidas_syn_simple_random_texture	adidas_syn_simple	4	3	0,9311	24,41
15	adidas_syn_simple_real_light	adidas_syn_simple	4	4	0,9362	28,06
16	adidas_syn_simple	adidas_syn_simple	4	5	0,9404	-139,34
17	adidas_syn_simple_real_texture	adidas_syn_simple	4	6	0,9434	-143,51

Auswahlverfahren 12: r\_cae\_on\_base\_comb\_fm8x12\_dim6144\_real3500\_no\_norm mit Real-Nähe

## C.2 Domain-Based-Ranking mit Real-Nähe

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	258,76
2	adidas_syn_se_random_camera	adidas_syn_se	2	1	0,5889	45,89
3	adidas_syn_se_random_light	adidas_syn_se	2	2	0,6018	60,96
4	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,6344	-258,76
5	adidas_syn_se	adidas_syn_se	2	4	0,6409	56,63
6	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6412	10,20
7	adidas_syn_se_random_texture	adidas_syn_se	2	6	0,659	32,09
8	adidas_syn_se_no_lc_table	adidas_syn_se	2	7	0,6646	44,00
9	adidas_syn_dr	adidas_syn_dr	3	1	0,793	58,34
10	adidas_syn_dr_color_texture	adidas_syn_dr	3	2	0,7994	14,30
11	adidas_syn_dr_real_texture	adidas_syn_dr	3	3	0,8091	56,14
12	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8065	75,02
13	adidas_syn_simple_random_texture	adidas_syn_simple	4	2	0,8372	55,43
14	adidas_syn_simple_random_light	adidas_syn_simple	4	3	0,8502	58,15
15	adidas_syn_simple_real_light	adidas_syn_simple	4	4	0,8647	64,13
16	adidas_syn_simple	adidas_syn_simple	4	5	0,8681	-236,42
17	adidas_syn_simple_real_texture	adidas_syn_simple	4	6	0,8726	-242,90

Auswahlverfahren 13: r\_cae\_on\_base\_comb\_fm8x12\_dim6144\_real3500\_norm  
mit Domänen und Real-Nähe

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	152,70
2	adidas_syn_se_random_light	adidas_syn_se	2	1	0,5756	37,81
3	adidas_syn_se_random_camera	adidas_syn_se	2	2	0,5865	26,67
4	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,621	-152,70
5	adidas_syn_se	adidas_syn_se	2	4	0,6244	47,58
6	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6283	13,50
7	adidas_syn_se_no_lc_table	adidas_syn_se	2	6	0,6866	27,57
8	adidas_syn_se_random_texture	adidas_syn_se	2	7	0,7354	36,09
9	adidas_syn_dr_color_texture	adidas_syn_dr	3	1	0,7713	45,22
10	adidas_syn_dr_real_texture	adidas_syn_dr	3	2	0,7726	14,21
11	adidas_syn_dr	adidas_syn_dr	3	3	0,7782	35,53
12	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8316	25,40
13	adidas_syn_simple_random_light	adidas_syn_simple	4	2	0,9196	28,60
14	adidas_syn_simple_random_texture	adidas_syn_simple	4	3	0,9311	24,41
15	adidas_syn_simple_real_light	adidas_syn_simple	4	4	0,9362	28,06
16	adidas_syn_simple	adidas_syn_simple	4	5	0,9404	-139,34
17	adidas_syn_simple_real_texture	adidas_syn_simple	4	6	0,9434	-143,51

Auswahlverfahren 14: r\_cae\_on\_base\_comb\_fm8x12\_dim6144\_real3500\_no\_norm  
mit Domänen und Real-Nähe

### C.3 Ranking mit Importance Score

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	258,76
2	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8065	75,02
3	adidas_syn_simple_real_light	adidas_syn_simple	4	4	0,8647	64,13
4	adidas_syn_se_random_light	adidas_syn_se	2	2	0,6018	60,96
5	adidas_syn_dr	adidas_syn_dr	3	1	0,793	58,34
6	adidas_syn_simple_random_light	adidas_syn_simple	4	3	0,8502	58,15
7	adidas_syn_se	adidas_syn_se	2	4	0,6409	56,63
8	adidas_syn_dr_real_texture	adidas_syn_dr	3	3	0,8091	56,14
9	adidas_syn_simple_random_texture	adidas_syn_simple	4	2	0,8372	55,43
10	adidas_syn_se_random_camera	adidas_syn_se	2	1	0,5889	45,89
11	adidas_syn_se_no_lc_table	adidas_syn_se	2	7	0,6646	44,00
12	adidas_syn_se_random_texture	adidas_syn_se	2	6	0,659	32,09
13	adidas_syn_dr_color_texture	adidas_syn_dr	3	2	0,7994	14,30
14	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6412	10,20
15	adidas_syn_simple	adidas_syn_simple	4	5	0,8681	-236,42
16	adidas_syn_simple_real_texture	adidas_syn_simple	4	6	0,8726	-242,90
17	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,6344	-258,76

Auswahlverfahren 15: r\_cae\_on\_base\_comb\_fm8x12\_dim6144\_real3500\_norm  
mit Importance Score

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	152,70
2	adidas_syn_se	adidas_syn_se	2	4	0,6244	47,58
3	adidas_syn_dr_color_texture	adidas_syn_dr	3	1	0,7713	45,22
4	adidas_syn_se_random_light	adidas_syn_se	2	1	0,5756	37,81
5	adidas_syn_se_random_texture	adidas_syn_se	2	7	0,7354	36,09
6	adidas_syn_dr	adidas_syn_dr	3	3	0,7782	35,53
7	adidas_syn_simple_random_light	adidas_syn_simple	4	2	0,9196	28,60
8	adidas_syn_simple_real_light	adidas_syn_simple	4	4	0,9362	28,06
9	adidas_syn_se_no_lc_table	adidas_syn_se	2	6	0,6866	27,57
10	adidas_syn_se_random_camera	adidas_syn_se	2	2	0,5865	26,67
11	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8316	25,40
12	adidas_syn_simple_random_texture	adidas_syn_simple	4	3	0,9311	24,41
13	adidas_syn_dr_real_texture	adidas_syn_dr	3	2	0,7726	14,21
14	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6283	13,50
15	adidas_syn_simple	adidas_syn_simple	4	5	0,9404	-139,34
16	adidas_syn_simple_real_texture	adidas_syn_simple	4	6	0,9434	-143,51
17	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,621	-152,70

Auswahlverfahren 16: r\_cae\_on\_base\_comb\_fm8x12\_dim6144\_real3500\_no\_norm  
mit Importance Score

#### C.4 Domain-Based-Ranking mit Importance Score

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	258,76
2	adidas_syn_se_random_light	adidas_syn_se	2	2	0,6018	60,96
3	adidas_syn_se	adidas_syn_se	2	4	0,6409	56,63
4	adidas_syn_se_random_camera	adidas_syn_se	2	1	0,5889	45,89
5	adidas_syn_se_no_lc_table	adidas_syn_se	2	7	0,6646	44,00
6	adidas_syn_se_random_texture	adidas_syn_se	2	6	0,659	32,09
7	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6412	10,20
8	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,6344	-258,76
9	adidas_syn_dr	adidas_syn_dr	3	1	0,793	58,34
10	adidas_syn_dr_real_texture	adidas_syn_dr	3	3	0,8091	56,14
11	adidas_syn_dr_color_texture	adidas_syn_dr	3	2	0,7994	14,30
12	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8065	75,02
13	adidas_syn_simple_real_light	adidas_syn_simple	4	4	0,8647	64,13
14	adidas_syn_simple_random_light	adidas_syn_simple	4	3	0,8502	58,15
15	adidas_syn_simple_random_texture	adidas_syn_simple	4	2	0,8372	55,43
16	adidas_syn_simple	adidas_syn_simple	4	5	0,8681	-236,42
17	adidas_syn_simple_real_texture	adidas_syn_simple	4	6	0,8726	-242,90

Auswahlverfahren 17: r\_cae\_on\_base\_comb\_fm8x12\_dim6144\_real3500\_norm  
mit Domänen und Importance Score

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	152,70
2	adidas_syn_se	adidas_syn_se	2	4	0,6244	47,58
3	adidas_syn_se_random_light	adidas_syn_se	2	1	0,5756	37,81
4	adidas_syn_se_random_texture	adidas_syn_se	2	7	0,7354	36,09
5	adidas_syn_se_no_lc_table	adidas_syn_se	2	6	0,6866	27,57
6	adidas_syn_se_random_camera	adidas_syn_se	2	2	0,5865	26,67
7	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6283	13,50
8	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,621	-152,70
9	adidas_syn_dr_color_texture	adidas_syn_dr	3	1	0,7713	45,22
10	adidas_syn_dr	adidas_syn_dr	3	3	0,7782	35,53
11	adidas_syn_dr_real_texture	adidas_syn_dr	3	2	0,7726	14,21
12	adidas_syn_simple_random_light	adidas_syn_simple	4	2	0,9196	28,60
13	adidas_syn_simple_real_light	adidas_syn_simple	4	4	0,9362	28,06
14	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8316	25,40
15	adidas_syn_simple_random_texture	adidas_syn_simple	4	3	0,9311	24,41
16	adidas_syn_simple	adidas_syn_simple	4	5	0,9404	-139,34
17	adidas_syn_simple_real_texture	adidas_syn_simple	4	6	0,9434	-143,51

Auswahlverfahren 18: r\_cae\_on\_base\_comb\_fm8x12\_dim6144\_real3500\_no\_norm  
mit Domänen und Importance Score

## D Auswahlverfahren: r\_cae\_on\_base\_comb\_fm4x6\_dim1536\_real3500

### D.1 Ranking mit Real-Nähe

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	43,54
2	adidas_syn_se_random_light	adidas_syn_se	2	1	0,6426	9,96
3	adidas_syn_se_color_texture	adidas_syn_se	2	2	0,6676	10,55
4	adidas_syn_se	adidas_syn_se	2	3	0,6689	-43,54
5	adidas_syn_se_static_light	adidas_syn_se	2	4	0,6732	8,93
6	adidas_syn_se_random_camera	adidas_syn_se	2	5	0,675	-8,41
7	adidas_syn_se_random_texture	adidas_syn_se	2	6	0,7083	6,31
8	adidas_syn_simple_real_camera	adidas_syn_simple	3	1	0,7251	10,13
9	adidas_syn_simple_random_texture	adidas_syn_simple	3	2	0,7352	10,37
10	adidas_syn_simple_random_light	adidas_syn_simple	3	3	0,7421	1,33
11	adidas_syn_simple_real_light	adidas_syn_simple	3	4	0,7426	10,25
12	adidas_syn_simple_real_texture	adidas_syn_simple	3	5	0,7434	10,95
13	adidas_syn_simple	adidas_syn_simple	3	6	0,7449	10,64
14	adidas_syn_se_no_lc_table	adidas_syn_se	2	7	0,7772	10,19
15	adidas_syn_dr	adidas_syn_dr	4	1	0,8841	10,55
16	adidas_syn_dr_color_texture	adidas_syn_dr	4	2	0,8867	-39,54
17	adidas_syn_dr_real_texture	adidas_syn_dr	4	3	0,8919	-41,53

Auswahlverfahren 19: r\_cae\_on\_base\_comb\_fm4x6\_dim1536\_real3500\_norm mit Real-Nähe

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	31,62
2	adidas_syn_se_random_camera	adidas_syn_se	2	1	0,6172	6,96
3	adidas_syn_se_random_light	adidas_syn_se	2	2	0,6221	5,59
4	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,6548	-31,62
5	adidas_syn_se	adidas_syn_se	2	4	0,6584	5,81
6	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6629	-2,08
7	adidas_syn_se_no_lc_table	adidas_syn_se	2	6	0,6983	8,16
8	adidas_syn_se_random_texture	adidas_syn_se	2	7	0,7383	7,00
9	adidas_syn_dr_color_texture	adidas_syn_dr	3	1	0,7884	7,06
10	adidas_syn_dr	adidas_syn_dr	3	2	0,7918	5,97
11	adidas_syn_dr_real_texture	adidas_syn_dr	3	3	0,7941	5,20
12	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8161	7,23
13	adidas_syn_simple_random_light	adidas_syn_simple	4	2	0,8253	5,70
14	adidas_syn_simple_real_light	adidas_syn_simple	4	3	0,8529	5,47
15	adidas_syn_simple	adidas_syn_simple	4	4	0,8651	5,57
16	adidas_syn_simple_real_texture	adidas_syn_simple	4	5	0,8668	-30,66
17	adidas_syn_simple_random_texture	adidas_syn_simple	4	6	0,8763	-30,81

Auswahlverfahren 20: r\_cae\_on\_base\_comb\_fm4x6\_dim1536\_real3500\_no\_norm mit Real-Nähe

## D.2 Domain-Based Ranking mit Real-Nähe

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	43,54
2	adidas_syn_se_random_light	adidas_syn_se	2	1	0,6426	9,96
3	adidas_syn_se_color_texture	adidas_syn_se	2	2	0,6676	10,55
4	adidas_syn_se	adidas_syn_se	2	3	0,6689	-43,54
5	adidas_syn_se_static_light	adidas_syn_se	2	4	0,6732	8,93
6	adidas_syn_se_random_camera	adidas_syn_se	2	5	0,675	-8,41
7	adidas_syn_se_random_texture	adidas_syn_se	2	6	0,7083	6,31
8	adidas_syn_se_no_lc_table	adidas_syn_se	2	7	0,7772	10,19
9	adidas_syn_simple_real_camera	adidas_syn_simple	3	1	0,7251	10,13
10	adidas_syn_simple_random_texture	adidas_syn_simple	3	2	0,7352	10,37
11	adidas_syn_simple_random_light	adidas_syn_simple	3	3	0,7421	1,33
12	adidas_syn_simple_real_light	adidas_syn_simple	3	4	0,7426	10,25
13	adidas_syn_simple_real_texture	adidas_syn_simple	3	5	0,7434	10,95
14	adidas_syn_simple	adidas_syn_simple	3	6	0,7449	10,64
15	adidas_syn_dr	adidas_syn_dr	4	1	0,8841	10,55
16	adidas_syn_dr_color_texture	adidas_syn_dr	4	2	0,8867	-39,54
17	adidas_syn_dr_real_texture	adidas_syn_dr	4	3	0,8919	-41,53

Auswahlverfahren 21: r\_cae\_on\_base\_comb\_fm4x6\_dim1536\_real3500\_norm  
mit Domänen und Real-Nähe

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	31,62
2	adidas_syn_se_random_camera	adidas_syn_se	2	1	0,6172	6,96
3	adidas_syn_se_random_light	adidas_syn_se	2	2	0,6221	5,59
4	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,6548	-31,62
5	adidas_syn_se	adidas_syn_se	2	4	0,6584	5,81
6	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6629	-2,08
7	adidas_syn_se_no_lc_table	adidas_syn_se	2	6	0,6983	8,16
8	adidas_syn_se_random_texture	adidas_syn_se	2	7	0,7383	7,00
9	adidas_syn_dr_color_texture	adidas_syn_dr	3	1	0,7884	7,06
10	adidas_syn_dr	adidas_syn_dr	3	2	0,7918	5,97
11	adidas_syn_dr_real_texture	adidas_syn_dr	3	3	0,7941	5,20
12	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8161	7,23
13	adidas_syn_simple_random_light	adidas_syn_simple	4	2	0,8253	5,70
14	adidas_syn_simple_real_light	adidas_syn_simple	4	3	0,8529	5,47
15	adidas_syn_simple	adidas_syn_simple	4	4	0,8651	5,57
16	adidas_syn_simple_real_texture	adidas_syn_simple	4	5	0,8668	-30,66
17	adidas_syn_simple_random_texture	adidas_syn_simple	4	6	0,8763	-30,81

Auswahlverfahren 22: r\_cae\_on\_base\_comb\_fm4x6\_dim1536\_real3500\_no\_norm  
mit Domänen und Real-Nähe

### D.3 Ranking mit Importance Score

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	43,54
2	adidas_syn_simple_real_texture	adidas_syn_simple	3	5	0,7434	10,95
3	adidas_syn_simple	adidas_syn_simple	3	6	0,7449	10,64
4	adidas_syn_dr	adidas_syn_dr	4	1	0,8841	10,55
5	adidas_syn_se_color_texture	adidas_syn_se	2	2	0,6676	10,55
6	adidas_syn_simple_random_texture	adidas_syn_simple	3	2	0,7352	10,37
7	adidas_syn_simple_real_light	adidas_syn_simple	3	4	0,7426	10,25
8	adidas_syn_se_no_lc_table	adidas_syn_se	2	7	0,7772	10,19
9	adidas_syn_simple_real_camera	adidas_syn_simple	3	1	0,7251	10,13
10	adidas_syn_se_random_light	adidas_syn_se	2	1	0,6426	9,96
11	adidas_syn_se_static_light	adidas_syn_se	2	4	0,6732	8,93
12	adidas_syn_se_random_texture	adidas_syn_se	2	6	0,7083	6,31
13	adidas_syn_simple_random_light	adidas_syn_simple	3	3	0,7421	1,33
14	adidas_syn_se_random_camera	adidas_syn_se	2	5	0,675	-8,41
15	adidas_syn_dr_color_texture	adidas_syn_dr	4	2	0,8867	-39,54
16	adidas_syn_dr_real_texture	adidas_syn_dr	4	3	0,8919	-41,53
17	adidas_syn_se	adidas_syn_se	2	3	0,6689	-43,54

Auswahlverfahren 23: r\_cae\_on\_base\_comb\_fm4x6\_dim1536\_real3500\_norm  
mit Importance Score

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	31,62
2	adidas_syn_se_no_lc_table	adidas_syn_se	2	6	0,6983	8,16
3	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8161	7,23
4	adidas_syn_dr_color_texture	adidas_syn_dr	3	1	0,7884	7,06
5	adidas_syn_se_random_texture	adidas_syn_se	2	7	0,7383	7,00
6	adidas_syn_se_random_camera	adidas_syn_se	2	1	0,6172	6,96
7	adidas_syn_dr	adidas_syn_dr	3	2	0,7918	5,97
8	adidas_syn_se	adidas_syn_se	2	4	0,6584	5,81
9	adidas_syn_simple_random_light	adidas_syn_simple	4	2	0,8253	5,70
10	adidas_syn_se_random_light	adidas_syn_se	2	2	0,6221	5,59
11	adidas_syn_simple	adidas_syn_simple	4	4	0,8651	5,57
12	adidas_syn_simple_real_light	adidas_syn_simple	4	3	0,8529	5,47
13	adidas_syn_dr_real_texture	adidas_syn_dr	3	3	0,7941	5,20
14	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6629	-2,08
15	adidas_syn_simple_real_texture	adidas_syn_simple	4	5	0,8668	-30,66
16	adidas_syn_simple_random_texture	adidas_syn_simple	4	6	0,8763	-30,81
17	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,6548	-31,62

Auswahlverfahren 24: r\_cae\_on\_base\_comb\_fm4x6\_dim1536\_real3500\_no\_norm  
mit Importance Score

## D.4 Domain-Based Ranking mit Importance Score

	Dataset	57	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	43,54
2	adidas_syn_se_color_texture	adidas_syn_se	2	2	0,6676	10,55
3	adidas_syn_se_no_lc_table	adidas_syn_se	2	7	0,7772	10,19
4	adidas_syn_se_random_light	adidas_syn_se	2	1	0,6426	9,96
5	adidas_syn_se_static_light	adidas_syn_se	2	4	0,6732	8,93
6	adidas_syn_se_random_texture	adidas_syn_se	2	6	0,7083	6,31
7	adidas_syn_se_random_camera	adidas_syn_se	2	5	0,675	-8,41
8	adidas_syn_se	adidas_syn_se	2	3	0,6689	-43,54
9	adidas_syn_simple_real_texture	adidas_syn_simple	3	5	0,7434	10,95
10	adidas_syn_simple	adidas_syn_simple	3	6	0,7449	10,64
11	adidas_syn_simple_random_texture	adidas_syn_simple	3	2	0,7352	10,37
12	adidas_syn_simple_real_light	adidas_syn_simple	3	4	0,7426	10,25
13	adidas_syn_simple_real_camera	adidas_syn_simple	3	1	0,7251	10,13
14	adidas_syn_simple_random_light	adidas_syn_simple	3	3	0,7421	1,33
15	adidas_syn_dr	adidas_syn_dr	4	1	0,8841	10,55
16	adidas_syn_dr_color_texture	adidas_syn_dr	4	2	0,8867	-39,54
17	adidas_syn_dr_real_texture	adidas_syn_dr	4	3	0,8919	-41,53

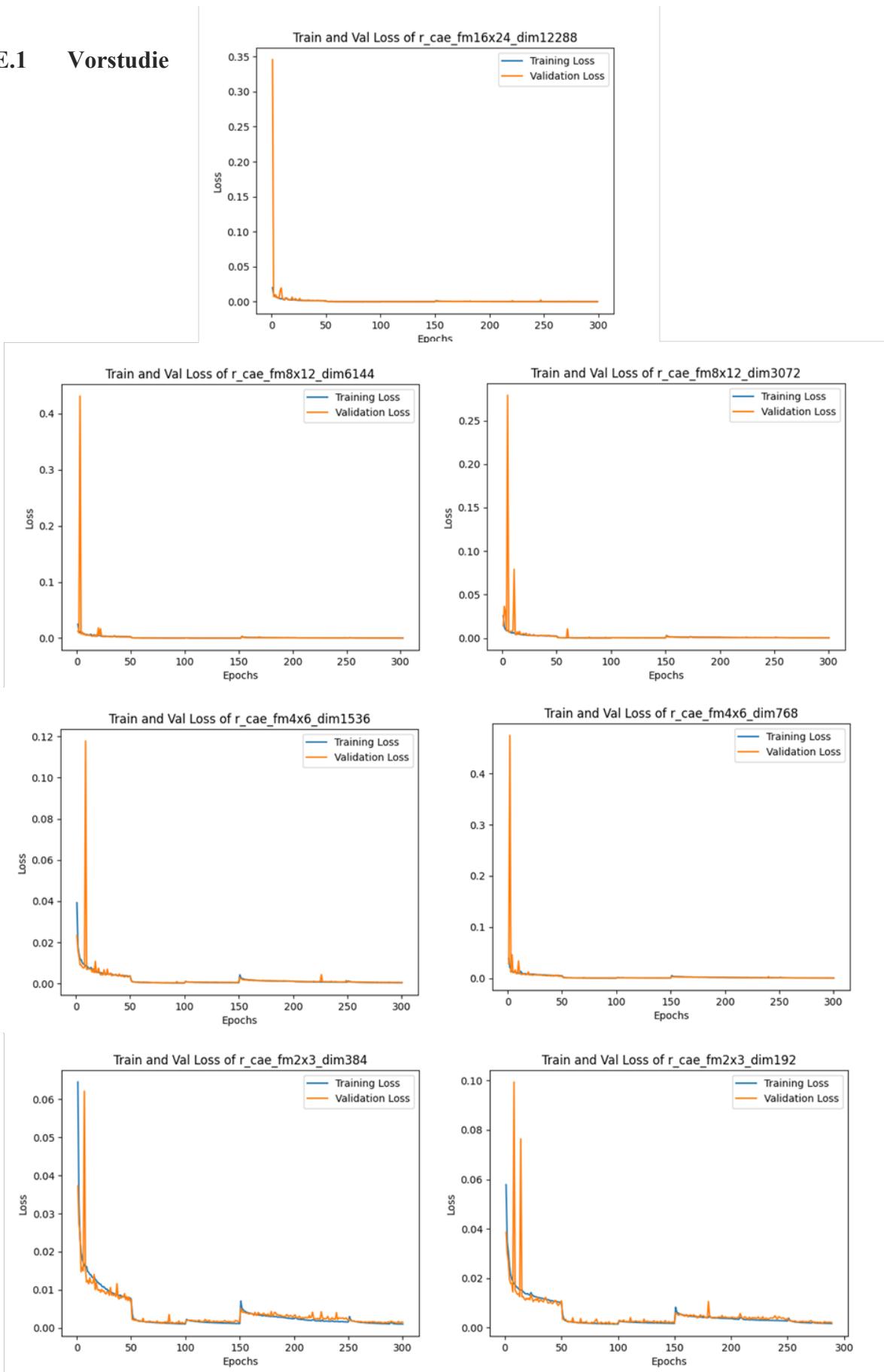
Auswahlverfahren 25: r\_cae\_on\_base\_comb\_fm4x6\_dim1536\_real3500\_norm  
mit Domänen und Importance Score

	Dataset	Domain	Rank	Rank In Domain	Distance To adidas_real	Score
1	adidas_real	adidas_real	1	1	0	31,62
2	adidas_syn_se_no_lc_table	adidas_syn_se	2	6	0,6983	8,16
3	adidas_syn_se_random_texture	adidas_syn_se	2	7	0,7383	7,00
4	adidas_syn_se_random_camera	adidas_syn_se	2	1	0,6172	6,96
5	adidas_syn_se	adidas_syn_se	2	4	0,6584	5,81
6	adidas_syn_se_random_light	adidas_syn_se	2	2	0,6221	5,59
7	adidas_syn_se_static_light	adidas_syn_se	2	5	0,6629	-2,08
8	adidas_syn_se_color_texture	adidas_syn_se	2	3	0,6548	-31,62
9	adidas_syn_dr_color_texture	adidas_syn_dr	3	1	0,7884	7,06
10	adidas_syn_dr	adidas_syn_dr	3	2	0,7918	5,97
11	adidas_syn_dr_real_texture	adidas_syn_dr	3	3	0,7941	5,20
12	adidas_syn_simple_real_camera	adidas_syn_simple	4	1	0,8161	7,23
13	adidas_syn_simple_random_light	adidas_syn_simple	4	2	0,8253	5,70
14	adidas_syn_simple	adidas_syn_simple	4	4	0,8651	5,57
15	adidas_syn_simple_real_light	adidas_syn_simple	4	3	0,8529	5,47
16	adidas_syn_simple_real_texture	adidas_syn_simple	4	5	0,8668	-30,66
17	adidas_syn_simple_random_texture	adidas_syn_simple	4	6	0,8763	-30,81

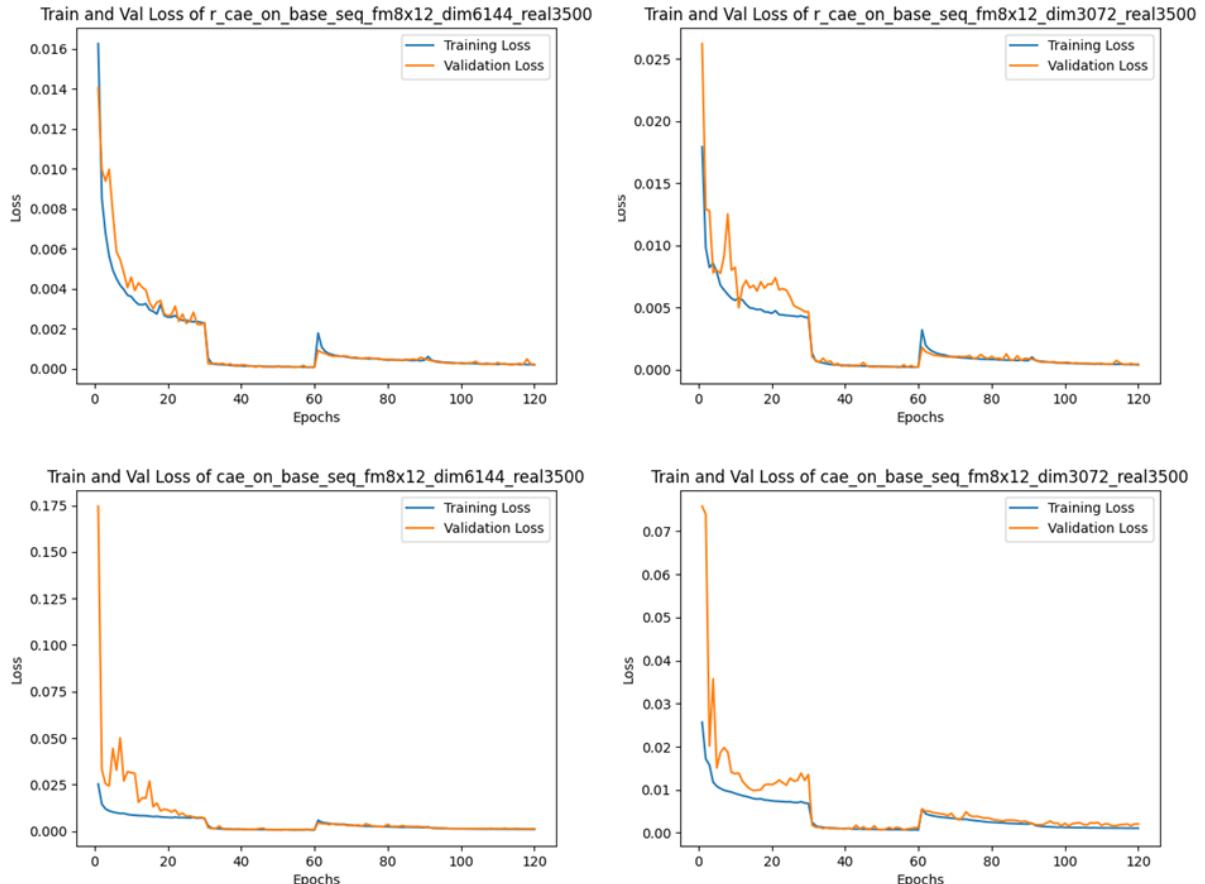
Auswahlverfahren 26: r\_cae\_on\_base\_comb\_fm4x6\_dim1536\_real3500\_no\_norm  
mit Domänen und Importance Score

## E Validation Loss

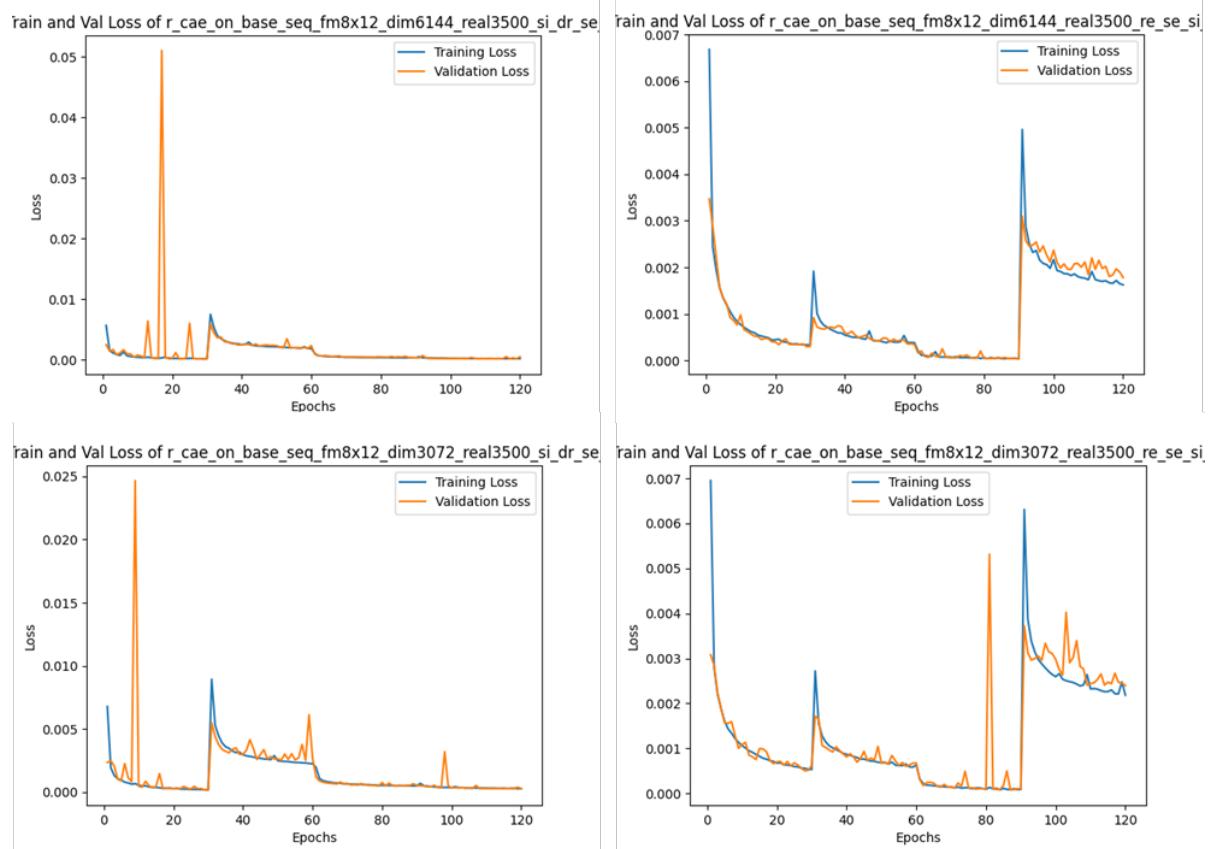
### E.1 Vorstudie



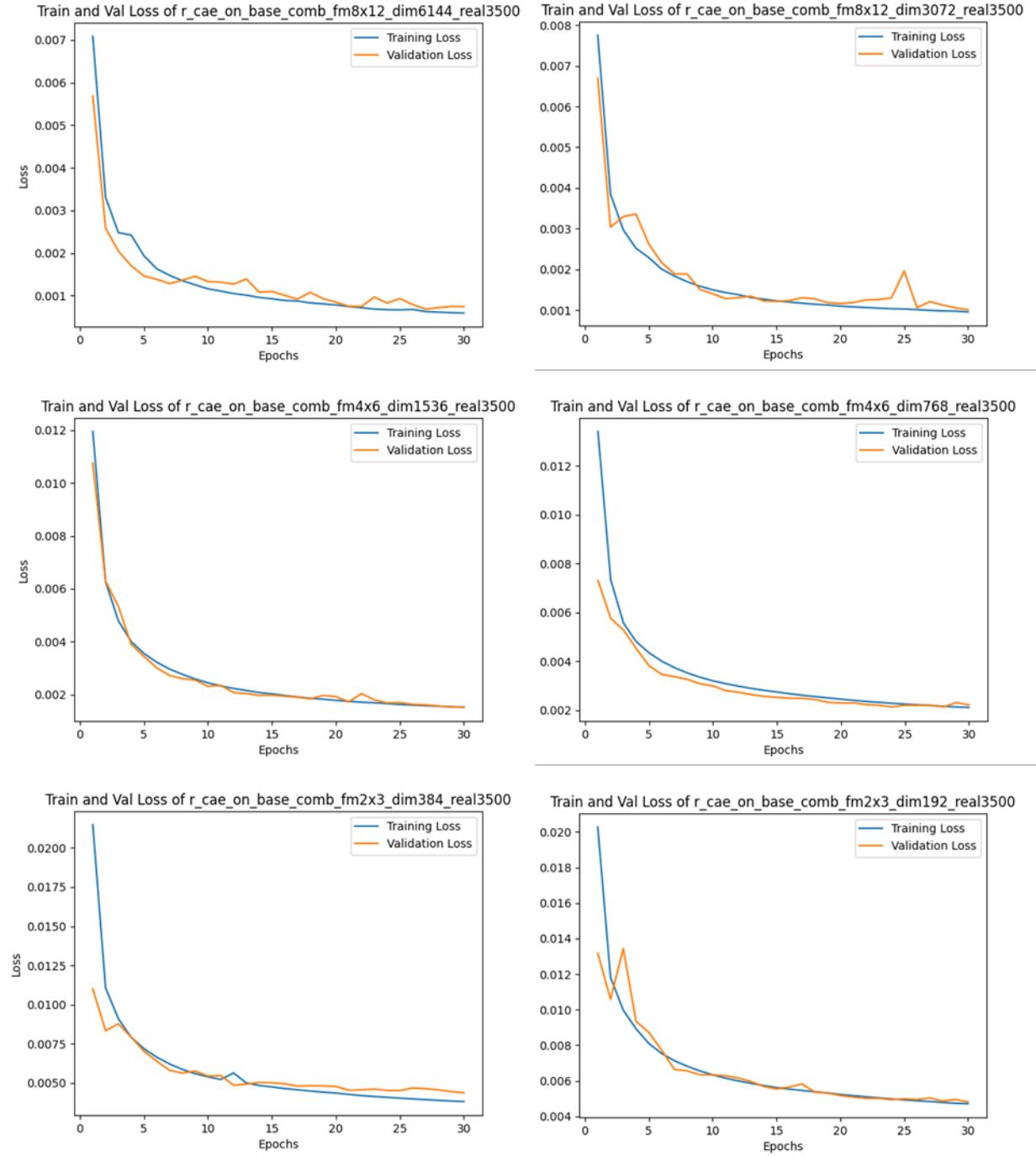
## E.2 Sequenziell: CAE vs. R-CAE



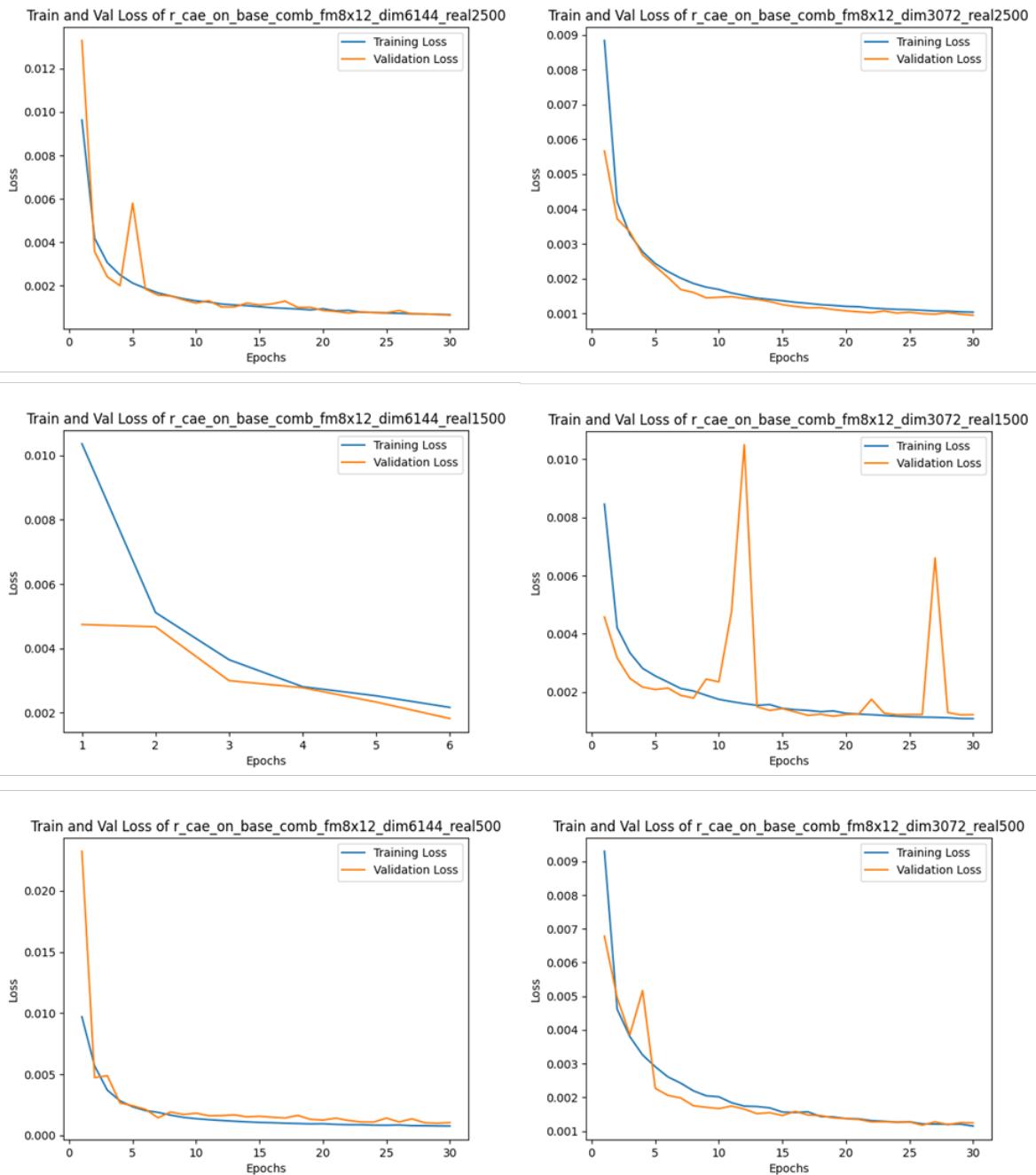
## E.3 Sequenziell: Datensatz-Reihenfolge



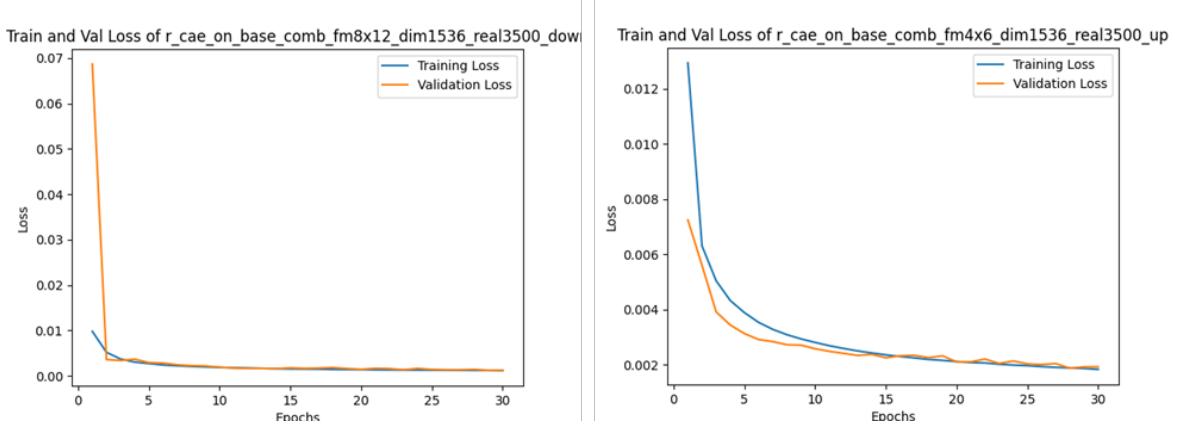
## E.4 Kombiniert: R-CAE mit kleinen Zielgrößen



## E.5 Kombiniert: Abnahme der Real-Trainingsdaten



## E.6 Kombiniert: Filteranzahl-Strategie





---

## **Eidesstattliche Erklärung**

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Quellen und Hilfsmittel angefertigt habe. Alle Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Quellen entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegen.

Karlsruhe, den 18.03.2024

  
\_\_\_\_\_  
Ilian Kohl