Hochschule Düsseldorf
University of Applied Sciences

Fachbereich Elektro- und Informationstechnik
Faculty of Electrical Engineering and Information Technology

Vergleich von datengetriebenen Clusteringmethoden im CIFAR10- und MNIST-Datensatz

Projektarbeit für das Mastermodul **„Künstliche Intelligenz“** PO2022

von Emre Kaplan (873889), Bünyamin Budak (816659) und Maxim Speczyk (1035069)

14.12.2025

Betreuende Professorin

Prof. Dr. -Ing. Dorothea Schwung

Fachbereich Elektro- und Informationstechnik

Münsterstr. 156

40476 Düsseldorf

dorothea.schwung@hs-duesseldorf.de

Inhaltsverzeichnis

Abbildungsverzeichnis

Formelverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

In der heutigen industriellen Produktion gewinnt die Digitalisierung zunehmend an Bedeutung. Ein zentraler Baustein für den effizienten Betrieb moderner Anlagen ist der Einsatz intelligenter Datenanalysetools. Der wirtschaftliche Druck, Produktionsprozesse zu optimieren, ist dabei enorm gestiegen.

Wie relevant robuste Analysesysteme sind, verdeutlicht der Bericht „The True Cost of Downtime 2024“ von Siemens. Demnach verlieren die Fortune Global 500-Unternehmen jährlich rund **1,5 Billionen US-Dollar** durch ungeplante Maschinenstillstände. Besonders drastisch zeigt sich dies in der Automobilindustrie, wo eine einzige Stunde Stillstand durchschnittliche Kosten von **2,3 Millionen US-Dollar** verursacht, ein Wert, der sich seit 2019 mehr als verdoppelt hat [1].

Um diese enormen Kosten zu vermeiden, ist der Übergang von einer reaktiven Wartung hin zu einer vorausschauenden Instandhaltung (*Predictive Maintenance*) unabdingbar. Ziel ist es, drohende Ausfälle zu erkennen, bevor sie zum Stillstand führen.

Die praktische Umsetzung scheitert jedoch oft an der Datenlage: Zwar sammeln Unternehmen riesige Mengen an Sensor- und Bilddaten, doch liegt der Großteil davon unbeschriftet vor. Da das manuelle Labeln dieser Datenmengen wirtschaftlich kaum darstellbar ist, reichen klassische überwachte Lernverfahren (*Supervised Learning*) oft nicht aus. Es soll evaluiert werden, wie gut moderne Clustering-Methoden in der Lage sind, eigenständig Strukturen in komplexen Bilddaten zu erkennen, ohne auf menschliche Vorarbeit angewiesen zu sein.

1.2 Problemstellung und Zielsetzung

Obwohl die Vorteile der Datenanalyse theoretisch bekannt sind, stellt die praktische Umsetzung auf komplexen Datensätzen eine Herausforderung dar. Im Rahmen dieser Arbeit soll das Clustering – ein Verfahren des unüberwachten Lernens – auf den Benchmark-Datensätzen **CIFAR10** und **MNIST** untersucht werden. Diese Datensätze dienen als standardisierte Testumgebungen, um die Leistungsfähigkeit verschiedener Algorithmen objektiv beurteilen zu können.

Das Kernproblem besteht darin, sinnvolle Strukturen in unbeschrifteten Bilddaten zu finden. Hierfür sollen verschiedene datengetriebene Methoden des maschinellen Lernens angewendet und verglichen werden:

- Deep Neural Networks und Auto-Encoder (AE) werden eingesetzt, um eine effiziente Merkmalsextraktion aus den Rohdaten durchzuführen.

- Zusätzlich soll untersucht werden, wie sich die Clustering-Ergebnisse verhalten, wenn sie auf generierten Bildern eines trainierten **Generative Adversarial Network (GAN)** basieren.
- Klassische Verfahren wie K-Means, dichtebasiertes Clustering und hierarchisches Clustering dienen dabei als Vergleichsgrundlage.

Die Zielsetzung der Arbeit ist es, ein Machine-Learning-Framework in Python zu entwickeln, das diese unterschiedlichen Algorithmen implementiert. Durch die Definition und Anwendung typischer Leistungsmetriken aus der Literatur sollen die Ergebnisse validiert und ein detaillierter Vergleich zwischen den Ansätzen ermöglicht werden. Letztlich dient das Projekt dem Erwerb tiefgehender Kompetenzen im Umgang mit modernen Data Science Verfahren und komplexen Datenstrukturen.

2 Grundlagen und Stand der Technik

Das folgende Kapitel legt das theoretische Fundament, indem es die Eigenschaften der verwendeten Benchmark-Datensätze MNIST und CIFAR10 detailliert beschreibt. Darüber hinaus werden die Funktionsweisen der angewandten Clustering-Algorithmen sowie die Architekturkonzepte von Auto-Encodern und Generative Adversarial Networks (GANs) erläutert, um ein tiefgreifendes Verständnis der methodischen Umsetzung zu ermöglichen.

2.1 Datensätze MNIST und CIFAR10

Zur Validierung der im Projekt eingesetzten Clustering-Verfahren, Autoencoder und GAN-Modelle werden standardisierte Datensätze verwendet. Hierzu kommt unter anderem der Datensatz MNIST (Modified National Institute of Standards and Technology) zum Einsatz, der häufig als Benchmark für Verfahren der Mustererkennung und des Clustering genutzt wird. Er bietet eine kontrollierte Umgebung zur vergleichenden Evaluierung lernbasierter Algorithmen.

Der MNIST-Datensatz umfasst insgesamt 70.000 handgeschriebene Ziffern, die in 60.000 Trainings- und 10.000 Testbeispiele unterteilt sind. Jede Instanz liegt als Graustufenbild mit einer Auflösung von 28 × 28 Pixeln vor. Die Intensitätswerte der Pixel entstehen durch ein Anti-Aliasing-Verfahren, das aus ursprünglich binären Vorlagen feinere Kantenstrukturen ableitet und dadurch eine realistischere Darstellung der Ziffern ermöglicht.

Der CIFAR-10-Datensatz (Canadian Institute for Advanced Research) stellt einen anspruchsvollen Benchmark für die Evaluierung von Verfahren der Objekterkennung und des unüberwachten Clusterings dar. Im Vergleich zu MNIST weist er eine deutlich höhere Komplexität auf, da er auf natürlichen Farbbildern anstelle isolierter Symbole basiert. Der Datensatz umfasst insgesamt 60.000 Bilder, die in 50.000 Trainings- und 10.000 Testbeispiele unterteilt sind.

Die Bilder liegen im RGB-Farbraum mit einer Auflösung von 32 × 32 Pixeln vor, was einer Eingabedimension von 3.072 Merkmalen pro Bild entspricht. CIFAR-10 ist in zehn disjunkte Objektklassen unterteilt, wobei jede Klasse jeweils 6.000 Instanzen enthält. Eine besondere Herausforderung für lernbasierte Verfahren ergibt sich aus der hohen Intra-Klassen-Varianz, die durch unterschiedliche Objektposen, Beleuchtungsbedingungen und komplexe Hintergründe verursacht wird.[13][14]

2.2 Schichttypen in Deep Neural Networks

Die Wahl der Schichttypen in neuronalen Netzen beeinflusst maßgeblich die Art der Merkmalsrepräsentation und damit die Eignung der extrahierten Features für nachgelagerte Clustering-Verfahren. In dieser Arbeit sind insbesondere vollständig

verbundene Schichten und Faltungsschichten von Bedeutung, da sie die Grundlage für die eingesetzten Autoencoder- und GAN-Architekturen bilden.

2.2.1 Vollständig verbundene Schichten

Wenn die Neuronen mit allen Neuronen der vorherigen Schicht verbunden sind, bilden sie eine vollständig verbundene Schicht. Man sagt auch „Fully Connected“. Jedes Neuron erhält alle Ausgabewerte als Eingabe und jede Verbindung besitzt ein eigenes Gewicht. So kann man mathematisch ausdrücken:

$$y = f\left(\sum W \cdot x + b\right)$$

Formel 1 Fully Connected"-Schicht

Wobei y den Ausgabevektor, f die Aktivierungsfunktion, W die Gewichtsmatrix, x den Eingabevektor und b den Bias-Vektor darstellt.

Sie wird häufig am Ende eines Netzwerks wie „Convolutional neuronal network“ (CNN) oder „Recurrent neuronal network“ (RNN) eingesetzt, um endgültige Entscheidungen zu treffen. Jedoch bringen sie den Nachteil, dass sie viele Parameter erfordern. [2]

2.2.2 Faltungsschichten

Ein weiterer Typ ist die sogenannte „Faltungsschicht“. Sie unterscheidet sich sehr im Vergleich zur vollständig verbundenen Schicht, da ein Neuron nur mit einem kleinen Bereich der vorherigen Schicht verbunden ist. Da geht ein kleiner Filter über die Eingabedaten und wertet lokale Zusammenhänge aus. Mathematisch lässt sich folgende Formel herleiten mit dem Faltungssatz. Das vereinfachte Ergebnis lautet:

$$Y(i,j) = \sum X(i+m, j+n) \cdot K(m,n)$$

Formel 2 Faltungsschicht

Wobei Y das Ergebnis als „Feature Map“, i den Zeilenindex des Ergebnisses, j den Spaltenindex des Ergebnisses, X die Eingabematrix, K den Filter, m den Zeilenindex und n den Spaltenindex des Filters kennzeichnet. [3]

In Einsatz kommen zahlreiche Filter, welche verschiedene Merkmale extrahieren sollen. Ein Filter kann Ecken identifizieren und ein anderer Filter Kanten. Deshalb finden diese Schichten oft Gebrauch in der Merkmalsextraktion.

2.3 Hyperparameter

Diese Algorithmen benötigen Parameter, die man extern vor dem Start des Trainingsprozesses einstellen muss, die auch „hyperparameters“ heißen. Sie unterscheiden sich von Modellparametern, die während des Trainings aus dem Daten gelernt werden. Die Konfigurationsvariablen beeinflussen dann das Verhalten und

werden meist bei der Gewichtung des neuronalen Netzes (auch bekannt als „Lernrate“). Es ist wichtig diese Parameter optimal auszulegen, indem man einen sinnvollen Suchraum definiert. Bei falscher Auslegung kann es zum „Underfitting“ (Modell kann schlecht die Struktur der Daten erfassen) oder „Overfitting“ (Modell hat zu viele Parameter auf die Muster und generalisiert schlecht). [4]

2.4 Datenvorbereitung – Feature Engineering

Nicht alle Daten sind geeignet für das maschinelle Lernen. Wie beim menschlichen Gehirn werden Informationen nach Kriterien wie Aufmerksamkeit, Arbeitsgedächtnis und selektive Verarbeitung gefiltert, sodass sie den Menschen nicht überfordern. Für die Leistungssteigerung gehen die Maschinen einige Prozesse wie Datenreduktion und Merkmalsextraktion durch, welche ein Teil des „Feature Engineering“ sind.

Es ist sinnvoll, zuerst zu extrahieren, dann zu reduzieren, weil man erst relevante Informationen aus den Rohdaten gewinnen muss, bevor man sie effizient verdichten kann.

2.4.1 Merkmalsextraktion

Bei komplexen Rohdaten verliert man schnell den Überblick. Häufig fällt die Klassifizierung sehr schwer, um sie zu analysieren. Bei der Frage „Was sind gute Beschreibungen meiner Daten?“ spielt die Merkmalsextraktion eine entscheidende Rolle. Typischerweise werden hier Faltungsschichten und Filter verwendet.

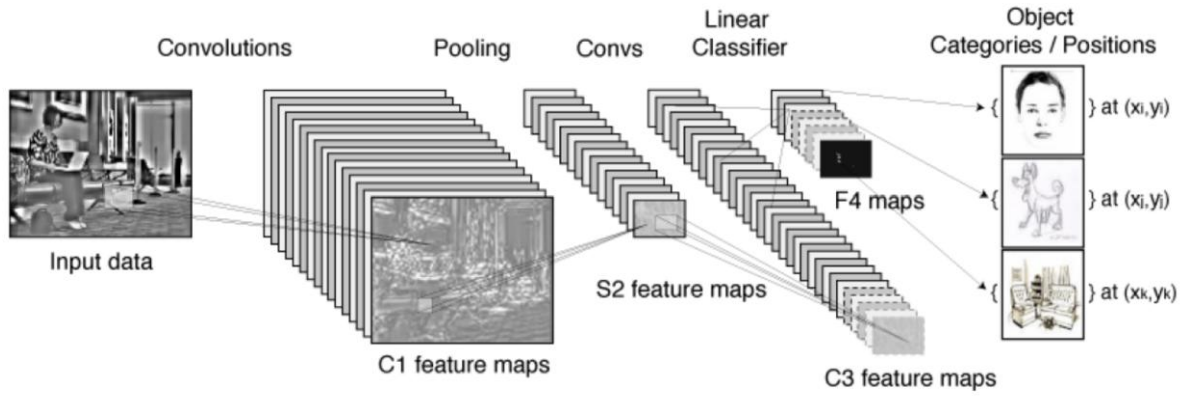

Abbildung 1 Schema Merkmalsextraktion aus dem Skript "CNN" (Teil 10)

2.4.2 Dimensionsreduktion (PCA & Auto-Encoder)

Die Datensätze können hohe Dimensionen (D) erreichen, was die Signalverarbeitung erschwert (durch Rauschen oder Verlangsamung wegen riesigen Mengen an Informationen). Bei der Frage „Wie kann ich mit weniger Zahlen auskommen?“ ist die Reduzierung der Dimensionen (d) ist notwendig, um die Visualisierung der Daten zu

vereinfachen (Muster und Trends leichter zu identifizieren) und so viele Informationen wie möglich beizubehalten. [5]

Eine einfache und weitverbreitete Methode ist „Principal components analysis“ (PCA), auch Hauptkomponentenanalyse. Es wird eine Vielzahl statistischer Variablen durch eine geringere Zahl möglichst kompakter und aussagekräftiger Linearkombinationen genähert. Durch das Transformieren der Datensätze werden die Daten auf ein anderes Koordinatensystem (meist 2D- oder 3D, da es für den Menschen vorstellbar ist) projiziert. Die Berechnung erfolgt durch die Bildung der Kovarianz-Matrix (Stärke der linearen Abhängigkeit) aus Daten mit höchster Abweichung oder die normierte Korrelationsmatrix und der Eigenwerte (Parameter). [6] Die Varianzen werden miteinander verglichen und die kleinste Varianz soll ausgeblendet werden, sodass die Reduktion mit minimalem Informationsverlust behaftet ist.

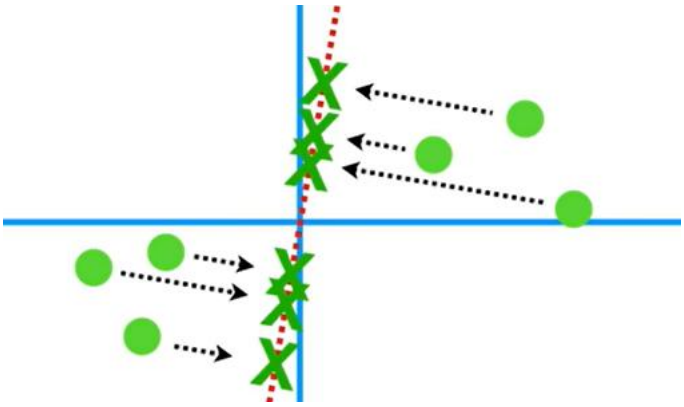

Abbildung 2 Projektion der Punkte auf neues Koordinatensystem

Wenn die Linearität des PCAs nicht genügt, verwendet man eine flexiblere Methode „Auto-Encoder“. Es ist ein neuronales Netz, das lernt, Daten zu komprimieren und in einen kompakten latenten Raum abzubilden. Die Formel für den latenten Raum lautet:

$$z = h(W \cdot x + b)$$

Formel 3 Berechnung latenter Raum

Wobei h die Aktivierungsfunktion darstellt. [2]

Aufgrund der Nichtlinearität kann der ursprüngliche Raum nicht wiederhergestellt werden, sodass es zu einem Verlust von Informationen kommt:

$$\mathbb{R}^D \rightarrow \mathbb{R}^d \rightarrow \mathbb{R}^{D\#}$$

Formel 4 verzerrter Dimensionswechsel

Der Fehler soll so klein wie möglich gehalten werden, sodass man diesen wieder ins Modell zurückführt, bis er eine definierte Toleranz unterschritten hat. [7] Die allgemeine mathematische Definition lautet:

$$\mathcal{L} = \sum \iota(x, g\big(h(x)\big), wobei\ \iota = |x - \hat{x}|^2$$

Formel 5 Fehlerberechnung Autoencoder

Es lassen sich drei Schichten im Modell bilden: „Encoder“ (Kompression der Daten), „Bottleneck“ (veränderte Darstellung für die nachfolgende Datenverarbeitung wie Clustering in Abbildung 2) und „Decoder“ (Rekonstruktion der Daten). Der Grund für die Namensgebung „Bottleneck“ für den latenten Raum ist der, dass die Informationen durch die engste Stelle im Netzwerk hindurchmüssen.

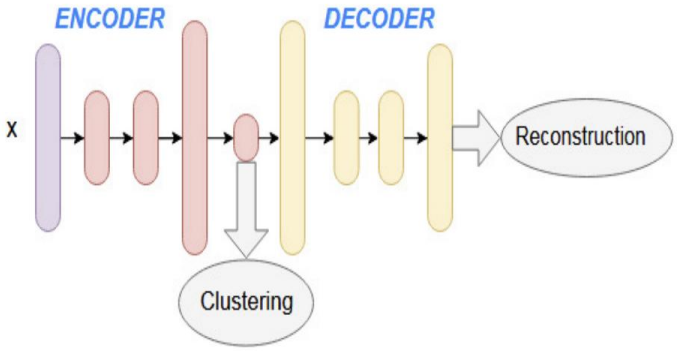

Abbildung 3 Schema Auto-Encoder aus "Deep Clustering with Dynamic Autoencoder" von Nairouz Mrabah 2019

2.4.2.1 Linearer Auto-Encoder (LAE)

Für die MNIST-Daten wird ein linearer Autoencoder genutzt, dessen Aktivierungsfunktion einer Geraden entspricht. Ähnlich wie die PCA lernt er eine lineare Abbildung, was ihn besonders effizient und rechensparend macht.

2.4.2.2 Convolutional Auto-Encoder (CAE)

Für CIFAR10 wird hingegen ein Convolutional Autoencoder eingesetzt, der Faltungsschichten zur Kompression und Rekonstruktion nutzt. Er extrahiert lokale Muster, bewahrt die räumliche Struktur und findet primär bei Bild- oder Videodaten Anwendung.

2.5 Unsupervised learning

Algorithmen des Unsupervised Learning analysieren ausschließlich Eingabedaten ohne vorgegebene Labels (Zielwerte). Sie agieren eigenständig, um Muster zu identifizieren, wodurch sich verborgene Strukturen in den Daten aufdecken lassen.

2.5.1 Clusteranalyse

Die Clusteranalyse ist ein Verfahren zur Entdeckung von Ähnlichkeitsstrukturen, bei dem Datentupel in Gruppen (Cluster) unterteilt werden. Dabei soll die Ähnlichkeit innerhalb eines Clusters maximiert und die Ähnlichkeit zwischen verschiedenen Clustern minimiert werden.

Die Ähnlichkeit wird mittels einer Distanzfunktion gemessen, wobei der Abstand das Maß für die Ähnlichkeit darstellt. Diese trägt den Namen „Minkowski-Distanz“ und sie stellt eine Möglichkeit dar, den geraden oder gekrümmten Weg zwischen zwei Punkten zu

messen, abhängig von einem gewählten Parameter k, der die Form beeinflusst. Diese wird durch folgende Formel beschrieben:

$$d_k(\vec{x}, \vec{y}) = \left(\sum_{i=1}^{n} (x_i - y_i)^k \right)^{\frac{1}{k}}$$

Formel 6 Minkowski-Distanz

Bekannte Spezialfälle dieser Familie sind [8]:

- $k = 1$: Manhattan- oder Cityblock-Distanz
- $k = 2$: Euklidische Distanz
- $k \to \infty$: Maximum-Distanz oder auch „Tschebyscheff-Abstand“ genannt

Mit steigendem k sinkt der Minkowski-Abstand, da höhere Werte die größte Differenz stärker gewichten und kleinere Unterschiede vernachlässigen. Im Abbild 3 ist ein Beispiel mit zwei Punkten dargestellt:

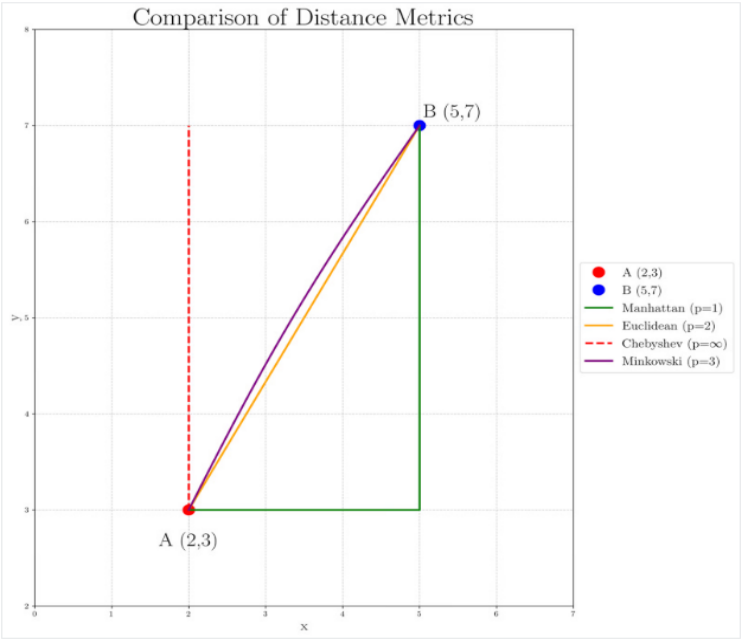

Abbildung 4 Arten von Minowski-Distanz

Die Manhattan-Distanz ($k = 1$), dargestellt durch den grünen Pfad, ergibt den längsten Pfad, da er streng dem Raster folgt, während die Tschebyscheff-Distanz ($k \to \infty$), dargestellt durch die rot gestrichelte Linie, sich ausschließlich auf die größte Koordinatendifferenz konzentriert (hier y). Diese Berechnung des Abstands wird in unterschiedlichen Cluster-Methoden angewandt.

2.5.1.1 K-Means-Clustering

Das „k-Means-Clustering“ ist ein unbeaufsichtigter Lernalgorithmus für Daten-Clustering. Es ist ein iterativer Prozess zur Minimierung der Summe der Abstände zwischen den Datenpunkten und ihren Cluster-Mittelpunkten.

Anfangs muss man eine Anzahl k von Clustern angeben, die zufällig initialisiert werden soll. Danach wird zufällig ein Clusterzentrum erstellt. Anschließend erfolgen zwei Schritte, die sich abwechselnd wiederholen:

- Datenpunktzuweisung: jedem Datenpunkt x wird das Clusterzentrum i zugewiesen, das ihm am nächsten ist. Dadurch verändert sich Schwerpunkt μ^i.
- Aktualisierung der Clusterzentren: neue Clusterzentren i werden als Mittelwert-Vektoren der zugeordneten Datenpunkte berechnet, wobei i den Index für der Schwerpunkt μ^i ist.

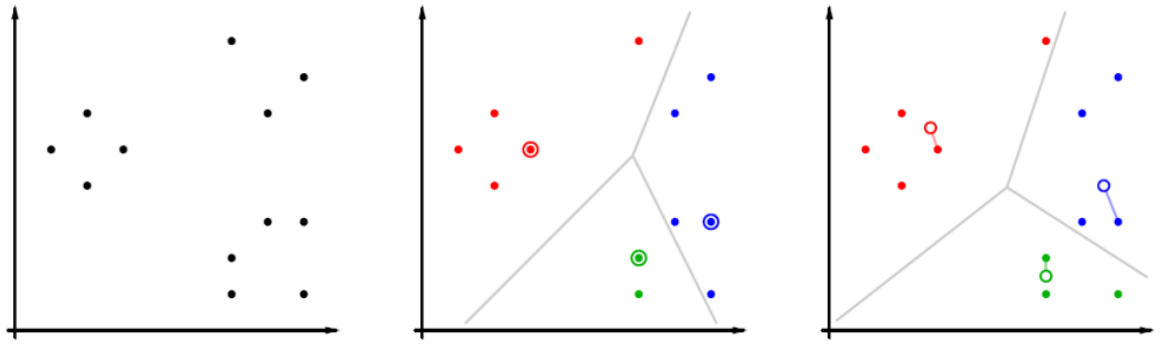

Abbildung 5 Schema Clusteranalyse

Dies geschieht bis sich die Clusterzentren nicht mehr ändern. Die Konvergenz ist normalerweise schnell, wenn gut initialisiert – wenn anfangs viele Punkte zum Mittelpunkt des Clusters einen ähnlichen Abstand haben – wird. Bei schlechter Initialisierung kann das Clustering fehlschlagen und in einem lokalen Minimum hängen.

Wenn der Datenpunkt p und Referenzvektor r das gleiche Cluster haben, rechnet man:

$$\vec{r}_{new} = \vec{r}_{old} + \mu(\vec{p} - \vec{r}_{old})$$

Formel 7 Anziehungsregel

Sie ziehen sich an (Anziehungsregel). Die Lernrate μ gibt die Gewichtung der Änderung an. Eine feste Lernrate kann zu Oszillationen führen, weshalb man sie öfter zeitlich verändert. Wenn der Datenpunkt p und Referenzvektor r unterschiedliche Cluster haben, rechnet man:

$$\vec{r}_{new} = \vec{r}_{old} - \mu(\vec{p} - \vec{r}_{old})$$

Formel 8 Abstoßungsregel

Sie stoßen sich ab (Abstoßungsregel). [8]

2.5.1.2 Hierarchisches agglomeratives Clustering

Im Unterschied zum k-Means-Clustering erhält beim hierarchischen agglomerativen Clustering („Bottom-Up-Ansatz“) jeder Datenpunkt ein eigenes Cluster, auch „Singletones“ genannt. Schritt für Schritt werden nebeneinanderstehende Cluster zusammengeführt, bis alle Datenpunkte in einem Cluster enthalten sind.

Wenn Cluster mehrere Datenpunkte enthalten, dann können verschiedene Abstände berechnet werden:

- Centroid-Linkage: Abstand zwischen den Schwerpunkten beider Cluster
- Single-Linkage: Abstand zwischen den beiden nächstgelegenen Punkten beider Cluster
- Complete-Linkage: Abstand zwischen den beiden am weitesten entfernten Punkten beider Cluster (führt zu kompakten Clustern)
- Ward-Linkage: die Varianz der beiden Punkte bleibt minimal

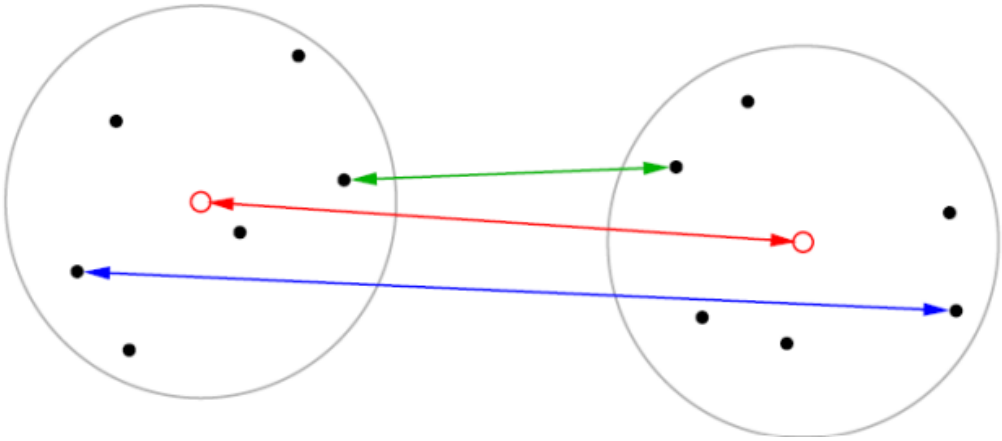

Abbildung 6 Schema Hierarchisches agglomeratives Clustering

Die Implementierung erfolgt durch eine Matrix, die die paarweisen Abstände der Datenpunkte enthält. In jedem Schritt werden die Zeilen und Spalten gelöscht, die den beiden Clustern entsprechen, die am nächsten beieinander liegen.

Eine neue Zeile und Spalte, die dem Cluster entspricht, das durch das Zusammenführen dieser Cluster gebildet wird, wird der Matrix hinzugefügt:

$$d_{k*} = d_{*k} = \alpha_i d_{ik} + \alpha_j d_{jk} + \beta d_{ij} + \gamma \left| d_{ik} - d_{jk} \right|$$

Formel 9 Zusammenführen der Cluster (hierarchisch)

Wobei i und j die Indizes der zusammengeführten Cluster, k das Indiz des alten Clusters, $*$ der Index des neuen Clusters, α_i, α_j, β_i und β_j die Parameter speziell für die Methode darstellen. Der einfachste Ansatz ist folgender:

- Gewünschten Mindestabstand zwischen Cluster angeben
- Das Zusammenführen von Clustern soll gestoppt werden, wenn die nächsten zwei Cluster weiter als dieser Abstand voneinander entfernt sind

Methode	α_i	α_j	β_i	β_j
Centroid-Linkage	$\frac{n_i}{n_i + n_j}$	$\frac{n_j}{n_i + n_j}$	$-\frac{n_i\, n_j}{n_i + n_i}$	0
Single-Linkage	½	½	0	-½

Complete-Linkage	½	½	0	+½
Ward-Linkage	$\frac{n_i + n_k}{n_i + n_j + n_k}$	$\frac{n_j + n_k}{n_i + n_j + n_k}$	0	0

Tabelle 1: Formeln für verschiedene Verbindungsarten (hierarchisch)

2.5.1.2.1 Ward-Linkage

Speziell für diese Arbeit nutzte man „Ward-Linkage“, wo man die Summe der quadrierten Abweichungen innerhalb der Cluster minimiert. Man verbindet die Punkte zu einem Cluster, die innerhalb der Cluster-Varianz sind, wodurch rundere Cluster entstehen.

2.5.1.3 Dichtebasiertes Clustering (DBSCAN)

Bei diesem Ansatz werden Cluster als Regionen hoher Datendichte definiert. Isolierte Punkte werden dabei als Ausreißer eingestuft und fließen nicht in die Clusterbildung ein. Dabei gelten folgende Regeln:

- Als Nachbarschaft eines Punktes x gelten alle Punkte innerhalb eines festgelegten Radius ∈.
- Punkt p kann als Clusterkern fungieren, wenn ihre Nachbarschaft eine Mindestgröße überschreitet – man nennt es „dicht“
- Punkt p, der auf dem Rand eines Clusters liegt, gehört zum Cluster, wenn es mindestens einen Kernpunkt x gibt – p ist von x aus dichte-erreichbar
- Punkt p und Punkt q sind dichte-verbunden, wenn es einen Kernpunkt o gibt, so dass p von o aus dichte-erreichbar ist und q von o aus dichte-erreichbar ist
- Cluster C ist eine Teilmenge aller Datenpunkte
- Punkte e, die zu keinem Cluster gehören, sind Rauschen

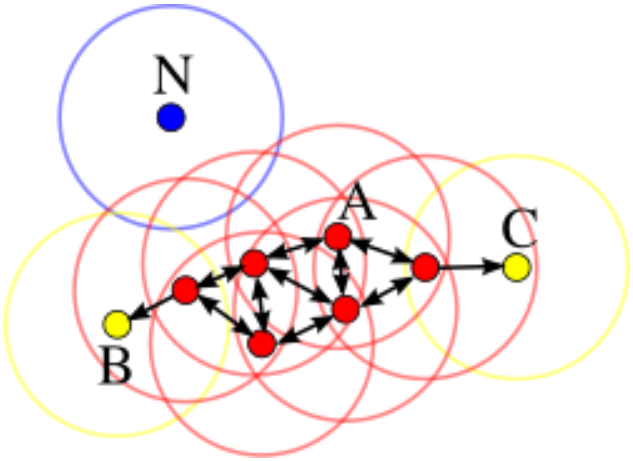

Abbildung 7 Schema DBSCAN

Das Hauptproblem dieser Clustering-Methode ist die unterschiedliche Dichte der Cluster, welche separate Hyperparameter für jedes Cluster erfordert. [8]

2.5.2 Generatives Modell

Generative Modelle werden für das „Generative Lernen“ benötigt, um über oberflächliche statistische Regelmäßigkeiten bestehender Daten hinauszugehen. In anderen Worten soll das Modell mit großen Mengen an Daten neue Inhalte generieren, die den Trainingsdaten ähneln.

Durch das Generieren neuer Daten sind sowohl unerwartete Ereignisse erkennbar als auch Datenabweichungen verständlicher. Es „ebnet den Weg zur Argumentation und Entscheidungsfindung“. [7]

2.5.2.1 Generative Adversarial Network (GAN)

Das Generative Adversarial Network (GAN) ist ein Architektur-Konzept, bei dem zwei aufeinanderliegende Netzwerke das „Generatornetzwerk“ und „Diskriminatornetzwerk“, dass Sampling von Daten direkt modellieren sollen. Während der Generator (oft als Dieb gekennzeichnet) versucht, den Diskriminator (Polizei) zu täuschen und andersrum nicht täuschen zu lassen. Hierbei soll die Bedingung gelten, dass die Hälfte der Samples echt und die andere Hälfte generiert ist. [7]

Einer der vielen Vorteile am „Adversial model“ im Vergleich zu anderen generativen Modellen ist der, dass kein MCMC-Sampling („Markov Chain Monte Carlo“, stochastisches Modell zur Modellierung von Zustandsübergangen) notwendig ist, wo neue Samples schrittweise erzeugt und aneinandergereiht werden. Das verlangsamt das Training und erschwert deutlich die Skalierbarkeit. [9]

Da Generatornetzwerk G und Diskriminatornetzwerk D vollständig verbundene Schichten sind, werden räumliche Strukturen der Bilder ignoriert.

2.5.2.2 Deep Convolutional GAN (DCGAN)

Somit wurde das GAN weiterentwickelt und nutzt „Convolational Layers“, auch Faltungsschichten statt vollständig verbundene Schichten. Sie wenden lokale Filter (Kernel) auf die Eingabedaten an, damit räumliche Muster erkannt werden. Zuerst wird der Filter über einen Teil und anschließend über das gesamte Bild „geschoben“ und es entsteht eine „Feature Map“ (gewichtete Summe, Downsampling), ganz nach der Frage „wie stark ähnelt dieser lokale Ausschnitt meinem Filter?“. [10]

Während der Diskriminator D diese spezielle Form verwendet, um lokale und globale Muster zu erkennen, setzt der Generator G eine umgekehrte Form „Transposed Convolution“ ein. Es erzeugt ein größeres Bild durch kleine Eingaben (Upsampling) und somit lokale Muster, die die gefälschten Bilder deutlicher machen. Das Zusammenspiel von beiden führt zu realistischen Bildern. [11]

2.6 Supervised learning

„Supervised learning algorithms“ beinhaltet das Beobachten von Outputs auf zufällig gewählte Inputs, um das Verhalten des Systems zu beschreiben. Dafür berechnet es eine

Funktion, die dieses Verhalten annähert. „Deep learning“ ist eine spezielle Form vom überwachten Lernen, wo mehr Schichten und Einheiten dem neuronalen Netz hinzugefügt werden, was zu höherer Komplexität führt. Der Grund für die Etablierung der „deep learning algorithms“ ist das Problem, dass einfache Algorithmen bei kleineren Dimensionen von Daten bestimmte Aufgaben erledigen können, aber schlechte Resultate bei allgemeineren Aufgaben erzielt. [2]

2.7 KI-Metriken

Nachdem man ein KI-Modell gebaut hat, kann man ihn beliebig einsetzen und testen. Um zu bewerten, wie die Leistung bei speziellen Aufgaben ist, führt man ein Maß ein, welches im Bereich der künstlichen Intelligenz auch „Metrik“ genannt wird. Sie ermöglicht einen standardisierten Vergleich und fundierte Beurteilung der Modellqualität.

2.7.1 Adjusted Rand Index (ARI)

Der „Adjusted Rand Index“ ist eine Kennzahl zur Bewertung der Übereinstimmung zwischen zwei Cluster-Zuordnungen, wobei sie den Zufallseffekt herausrechnet. Sie basiert auf dem Rand Index (RI), der den Anteil von Datenpaaren misst, die in beiden Partitionen entweder gemeinsam oder getrennt sind. ARI korrigiert diesen Wert, indem der erwartete Zufallswert subtrahiert wird. Der Wertbereich liegt zwischen −1 und 1, wobei 1 bedeutet perfekte Übereinstimmung und 0 entspricht rein zufälliger Übereinstimmung, negative Werte sind schlechter als Zufall. Die Metrik ist symmetrisch und unabhängig von der Anzahl der Cluster. [12]

2.7.2 Normalized Mutual Information (NMI)

Die „Normalized Mutual Information“ misst, wie viel Information die eine Partition über die andere enthält, und normalisiert diesen Wert, um ihn unabhängig von der Anzahl der Cluster und der Größe der Daten zu machen. Die Berechnung erfolgt über die gegenseitige Information (Mutual Information) und die Entropien der beiden Cluster. Der resultierende Wert liegt zwischen 0 und 1, wobei 1 eine perfekte Übereinstimmung und 0 keine gemeinsame Information bedeutet. NMI ist symmetrisch und unabhängig von der Anzahl der Cluster. [12]

2.7.3 Silhouette Score

Die Silhouette-Kennzahl misst, wie gut ein Punkt zu seinem eigenen Cluster passt im Vergleich zu anderen Clustern. Für jeden Punkt wird der Wert aus der durchschnittlichen Distanz zu Punkten im eigenen Cluster und der nächstgelegenen anderen Clustergruppe berechnet. Der Silhouette-Wert liegt zwischen −1 und 1: Werte nahe 1 bedeuten eine klare Zuordnung, 0 deutet auf eine Grenzlage hin, und negative Werte zeigen eine mögliche Fehlzuordnung. Der Durchschnitt aller Silhouette-Werte dient als Indikator für die Gesamtqualität des Clusters und wird häufig genutzt, um die optimale Clusteranzahl zu bestimmen. [12]

3 Methodik & Implementierung

3.1 Software-Stack & Framework

Für die Implementierung der neuronalen Netze (Autoencoder und DCGAN) kam das Deep-Learning-Framework **PyTorch** zum Einsatz. Die Datenvorverarbeitung sowie das Laden der Bilddatensätze erfolgte über **torchvision**. Die Clustering-Verfahren sowie die verwendeten Bewertungsmetriken (z. B. K-Means, DBSCAN, ARI, NMI und Silhouette Score) wurden mit Hilfe der Bibliothek **scikit-learn** umgesetzt. Zur numerischen Verarbeitung und effizienten Handhabung größerer Datenmengen wurde zusätzlich **NumPy** eingesetzt. Visualisierungen, Zwischenergebnisse und qualitative Analysen der Clusterstrukturen wurden mit **Matplotlib** und **Seaborn** erstellt.

3.2 Datenvorverarbeitung

Die Bilddaten der Datensätze MNIST und CIFAR-10 wurden vor der Weiterverarbeitung normalisiert, skaliert und in eine einheitliche Repräsentationsform überführt. Diese Schritte dienen sowohl der Stabilisierung des Trainingsprozesses der Deep-Learning-Modelle als auch der Verbesserung der Vergleichbarkeit in der anschließenden Clusteranalyse.

Die ursprünglichen Pixelwerte liegen im Bereich von 0 bis 255 vor und werden zunächst skaliert, um numerische Stabilität im Trainings- und Clusteringprozess zu gewährleisten. Für die Deep-Learning-Modelle, insbesondere Autoencoder und DCGAN, erfolgt eine Normalisierung auf den symmetrischen Wertebereich $[-1,1]$. Diese Skalierung unterstützt den Einsatz symmetrischer Aktivierungsfunktionen und trägt zu einem stabileren Gradienten verhalten während des Trainings bei.

Für die Clustering-Verfahren werden die Bilddaten hingegen standardisiert, sodass alle Merkmale einen Mittelwert von null und eine Varianz von eins aufweisen. Dadurch wird verhindert, dass einzelne Pixel aufgrund höherer Streuung einen überproportionalen Einfluss auf die Distanzberechnung und damit auf die Clusterbildung erhalten.

Um eine einheitliche Verarbeitung mit identischen Netzwerkarchitekturen zu ermöglichen, werden Bilddaten mit abweichender Auflösung auf ein gemeinsames Format gebracht. In diesem Zusammenhang werden Bilder mit geringerer Auflösung entsprechend hochskaliert, ohne die semantische Struktur der Bildinhalte wesentlich zu verändern.

3.3 Repräsentations- und Feature-Ebene

Die Leistungsfähigkeit von Clustering-Verfahren hängt entscheidend von der gewählten Datenrepräsentation ab. Insbesondere bei hochdimensionalen Bilddaten beeinflussen Dimensionalität und Abstraktionsgrad der Features maßgeblich die Trennbarkeit

semantisch ähnlicher Klassen. Ziel dieses Kapitels ist es, die in dieser Arbeit untersuchten Repräsentationsebenen überblicksartig darzustellen. Untersucht werden vier unterschiedliche Feature-Ebenen:

1. Rohpixel als Baseline
2. PCA-basierte Repräsentationen
3. Autoencoder-basierte Latent-Repräsentationen
4. DCGAN-basierte Features und synthetische Daten

Die spezifischen Eigenschaften sowie die methodische Umsetzung dieser Ebenen werden in den folgenden Unterabschnitten detailliert erläutert.

3.3.1 Rohdaten (Baseline)

Als Ausgangspunkt (Baseline) dienen die unverarbeiteten Pixelwerte der Bilder. Jedes Bild wird dabei als flacher Vektor betrachtet.

- MNIST: Ein Bild besteht aus 28x28 Pixeln, was zu 784 Vektoren führt.
- CIFAR-10: Ein Farbbild besteht aus 32x32 Pixeln, was zu 3072 Vektoren entspricht.

Herausforderung: Diese Darstellung leidet unter dem Curse of Dimensionality. Der euklidische Abstand (L2-Norm), auf den Algorithmen wie K-Means und DBSCAN basieren, verliert in hochdimensionalen Räumen an Aussagekraft. Semantisch ähnliche Objekte (z. B. zwei Autos in unterschiedlichen Farben oder Positionen) können im Pixelraum einen großen Abstand haben, während semantisch verschiedene Objekte sich pixelweise sehr ähnlich sein können.

3.3.2 PCA-Features

Um die Dimensionalität zu reduzieren und die Rechenkomplexität insbesondere bei der Berechnung der Distanzmatrizen für DBSCAN und Hierarchischem Clustering handhabbar zu machen, wurde die PCA angewendet. Das in PCA genutzte Verfahren wurde im Vorherigen Kapitel beleuchtet. In der Umsetzung wurde für den MNIST-Datensatz eine Reduktion auf 50 Komponenten vorgenommen. Bei den komplexeren Farbbildern des CIFAR-10-Datensatzes wurde die Anzahl der Komponenten hingegen so definiert, dass 95 % der ursprünglichen Varianz erhalten bleiben, was einer Dimension von etwa 200 Komponenten entspricht. So werden in diesem Schritt Redundanzen und Korrelationen zwischen benachbarten Pixeln entfernt.

3.3.3 Autoencoder-Latents

Aufbauend auf den theoretischen Grundlagen (siehe Kapitel 2.4.2) werden Autoencoder genutzt, um die Bilddaten in einen nicht-linearen, komprimierten Merkmalsraum zu

transformieren. Für das Clustering werden je nach Komplexität des Datensatzes zwei unterschiedliche Repräsentationstypen generiert:

MNIST (Linearer AE): Aufgrund der einfachen geometrischen Strukturen der Ziffern wird eine Repräsentation durch einen Fully-Connected Autoencoder erzeugt. Dieser komprimiert den 784-dimensionalen Input auf einen 32-dimensionalen Latent-Code.

CIFAR-10 (Convolutional AE): Für die komplexeren, natürlichen Farbbilder wird ein Convolutional Autoencoder (CAE) eingesetzt. Dieser nutzt Faltungsschichten, um räumliche Hierarchien und Texturen in einem 128-dimensionalen Vektor zu abstrahieren.

Der entscheidende Vorteil dieser Ebene liegt in der semantischen Robustheit. Durch die Minimierung des Rekonstruktionsfehlers wird das Modell gezwungen, nur die für die Bildstruktur essenziellen Merkmale im Bottleneck zu speichern, was die Trennbarkeit der Cluster in den nachfolgenden Verfahren signifikant erleichtert.

3.3.4 GAN-generierte Bilder

Ergänzend zu den realen Datensätzen werden synthetische Stichproben sowie interne Repräsentationen eines Deep Convolutional GAN (DCGAN) untersucht. Im Gegensatz zu den vorherigen Ebenen werden hier zwei unterschiedliche Ansätze verfolgt, um die Clustering-Verfahren zu evaluieren:

- Synthetische Bilddaten (Generator-Output): Hierbei werden vom Generator erzeugte Bilder als zusätzliche Datenpunkte in den bestehenden Pixelraum integriert. Ziel ist eine Form der Data Augmentation, um die Datendichte in spärlich besetzten Regionen zu erhöhen und die Stabilität von Algorithmen wie DBSCAN zu prüfen.
- Diskriminator-Features: Parallel dazu werden die im Diskriminator gelernten Merkmalskarten (Feature Maps) als hochabstrahierte Repräsentationen für das Clustering extrahiert. Dies dient der Robustheitsanalyse, um festzustellen, ob diese gelernten Merkmale eine bessere Trennung semantischer Klassen ermöglichen als klassische Verfahren.

3.4 Autoencoder-Training

Zur Merkmalsextraktion wurden zwei spezifische Autoencoder trainiert, um die hochdimensionalen Rohdaten auf kompakte Feature-Vektoren (Latents) zu reduzieren.

3.4.1 Architektur

Für die Datensätze MNIST und CIFAR-10 wurden aufgrund ihrer unterschiedlichen Komplexität zwei differenzierte Netzwerkarchitekturen implementiert:

Merkmal	Linearer Autoencoder (MNIST)	Convolutional Autoencoder (CIFAR-10)
Eingangsdimension	784 (28x28 Pixel, flach)	3x32x32 (RGB-Bilder)
Encoder-Struktur	Fully-Connected Layers (784 → 256 → 32)	3 Convolutional Layers mit ReLU-Aktivierung
Bottleneck	32 Dimensionen	128 Dimensionen
Decoder-Struktur	Symmetrisch zum Encoder mit Sigmoid-Output	3 Transposed Convolutional Layers mit Sigmoid-Output

Tabelle 2 Autoencoder-Training

3.4.2 Trainingskonfiguration und Hyperparameter

Das Training erfolgte mit dem Adam-Optimierer (η = 0.001) unter Minimierung des Mean Squared Error (MSE), um die Differenz zwischen Original und Rekonstruktion zu verringern. Die weiteren Hyperparameter wurden wie folgt festgelegt:

- Batch Size: Pro Iteration wurden 128 Bilder verarbeitet, um eine stabile Gradientenschätzung zu gewährleisten.
- Trainingsdauer (MNIST): Aufgrund der einfachen Struktur der Zifferndaten erreichte das Modell bereits nach 30 Epochen eine hinreichende Konvergenz.
- Trainingsdauer (CIFAR-10): Für die komplexeren Farbbilder wurden 50 Epochen angesetzt, um sicherzustellen, dass auch feine Details und Texturen vom Modell erfasst werden.

3.4.3 Speicherung der Latents

Nach dem Training wurden die Decoder verworfen und die Modelle in den Inferenz-Modus versetzt. Die Datensätze wurden vollständig durch die Encoder geleitet, um die komprimierten Repräsentationen zu extrahieren. Die resultierenden Latent-Vektoren dienen als optimierter Input für die nachfolgenden Cluster-Analysen.

3.4.4 Trainingsverlauf und Validierung

Die Qualität der extrahierten Latent-Features hängt direkt von der Fähigkeit der Autoencoder ab, die Eingabedaten zu komprimieren und wiederherzustellen. Zur

Validierung wurden sowohl die quantitativen Fehlerkurven als auch die qualitative visuelle Rekonstruktion analysiert.

Analyse der Lernkurven (Loss): Die Überwachung der Verlustfunktion (Mean Squared Error) während des Trainings zeigt für beide Datensätze ein stabiles Konvergenzverhalten ohne Anzeichen von Overfitting (siehe Abbildung 8 und 9.

MNIST (Linearer AE): Wie in Abbildung 8 zu sehen ist, fällt der Fehler bereits in den ersten 3 Epochen massiv ab (von ca. 0.040 auf 0.010). Ab Epoche 10 stagniert der Loss asymptotisch bei ca. 0.005. Dies bestätigt, dass die simple Architektur des linearen Autoencoders ausreicht, um die niedrig-komplexe Struktur der Ziffern effizient zu lernen. 20 Epochen wären hierfür auch völlig ausreichend gewesen.

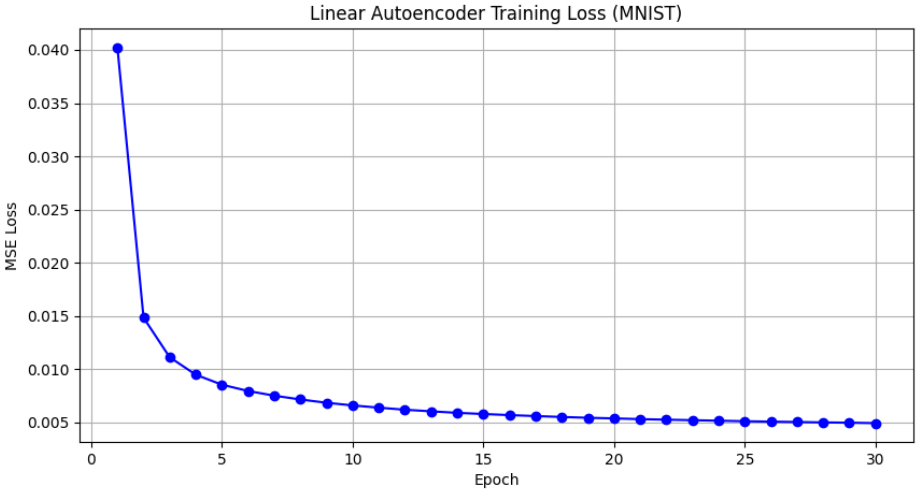

Abbildung 8 Linear Autoencoder Training Loss (MNIST)

CIFAR-10 (Convolutional AE): Der Verlauf in Abbildung 9 zeigt einen stetigen, aber langsameren Abfall des Fehlers über 50 Epochen. Im Gegensatz zu MNIST ist hier kein abruptes "Einrasten" zu sehen, sondern eine kontinuierliche Optimierung der Filtergewichte. Der Loss sinkt von anfänglich >0.0225 auf unter 0.005. Der glatte Verlauf deutet auf eine gute Lernrate von η =0.001 hin.

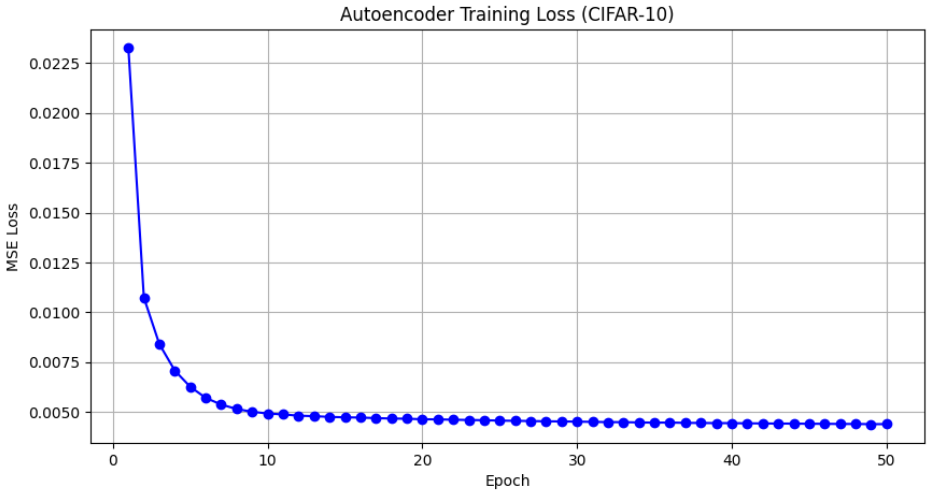

Abbildung 9 Autoencoder Training Loss (CIFAR-10)

Zur qualitativen Analyse wurden die Rekonstruktionen des Decoders visuell begutachtet. Ziel war es zu verifizieren, dass das Modell tatsächlich strukturelle Merkmale extrahiert, statt lediglich Bildrauschen zu übernehmen.

Wie die Abbildung 10 zu MNIST zeigt, ist hier eine nahezu perfekte Rekonstruktion der Ziffern. Die Konturen sind scharf und Artefakte sind kaum vorhanden. Das Modell hat gelernt, die Ziffern "2", "6", "8" etc. klar voneinander zu trennen und im Latent-Space (32 Dimensionen) präzise abzubilden.

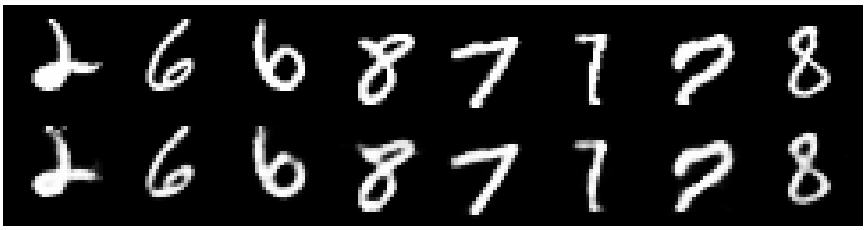

Abbildung 10 MNIST

Die Rekonstruktionen bei den CIFAR-10-Daten in Abbildung 11 zeigen ein für das Clustering wichtiges Phänomen die Bilder wirken nämlich leicht unscharf ("blurry"). So erkennt man zwar Globale Merkmale wie die Form des blauen LKWs (unten rechts) oder die Farbe des Himmels beim Flugzeug bleiben erhalten. Hochfrequente Details (z. B. Grashalme oder Fellstrukturen) werden jedoch geglättet.

Dies ist jedoch ein gewünschter Effekt des MSE-Loss. Der Autoencoder fungiert als Rauschfilter. Er zwingt das Modell, sich auf die grobe Form und Farbe (den "Inhalt") zu konzentrieren und ignoriert Pixel-Rauschen. Für das anschließende Clustering ist genau diese Abstraktion vorteilhaft, da sie den Fokus auf die Objektklasse legt und nicht auf irrelevante Details.

Abbildung 11 CIFAR-10

3.5 DCGAN-Training

Um synthetische Bilddaten für das Clustering zu erzeugen, wurde ein DCGAN implementiert. Ziel ist es, die Robustheit der Algorithmen gegenüber künstlich erzeugten Daten zu prüfen und die Datendichte in den Feature-Räumen zu erhöhen.

3.5.1 Warum DCGAN?

Die Wahl fiel auf das DCGAN, da diese Architektur durch den Einsatz von Faltungsschichten eine stabilere Konvergenz bei der Bildsynthese bietet als herkömmliche lineare Modelle. Um eine einheitliche Prozesskette zu gewährleisten, wurden die MNIST-Bilder von 28x28 auf 32x32 Pixel hochskaliert und somit an die native Auflösung von CIFAR-10 angepasst. Während der Generator lernt, abstrakte Informationen aus einem 100-dimensionalen Rauschvektor räumlich anzuordnen, unterscheiden sich die Anforderungen der Datensätze deutlich: Bei MNIST liegt der Fokus auf der Topologie einfacher geometrischer Formen, wohingegen das Modell bei CIFAR-10 die zusätzliche Komplexität der Drei-Kanal-Synthese (RGB) bewältigen muss, um die hohe interne Varianz und die Texturen realer Objekte konsistent abzubilden.

3.5.2 Netzwerkarchitektur des DCGAN

Die Implementierung folgt dem klassischen DCGAN-Design, wobei sowohl der Generator als auch der Diskriminator auf tiefen Faltungsstrukturen basieren. Im Gegensatz zum Autoencoder werden hier keine vollständig verbundenen Schichten genutzt, um die räumlichen Abhängigkeiten der Bilder (32x32 Pixel) optimal zu verarbeiten.

Merkmal	**Generator (NetG)**	**Diskriminator (NetD)**
Input	100-dimensionaler Rauschvektor z	32x32 Bild (nc=1 für MNIST, nc=3 für CIFAR)

Merkmal	Generator (NetG)	Diskriminator (NetD)
Kern-Operation	Transposed Convolution (Upsampling)	Strided Convolution (Downsampling)
Aktivierung	ReLU (Hidden) & Tanh (Output)	LeakyReLU (0.2) & Sigmoid (Output)
Stabilität	Batch Normalization in allen Schichten	Batch Normalization (außer im Input-Layer)
Feature Maps	Startet mit 512 (64x8) Filtern	Startet mit 64 Filtern

Tabelle 3 Netzwerkarchitektur des DCGAN

3.5.3 Trainingsparameter und Datenumfang

Das DCGAN-Training wurde als kompetitives Minimax-Spiel mit der Binary Cross Entropy (BCE) als Verlustfunktion implementiert. Die Wahl der Hyperparameter orientiert sich dabei eng an der Fachliteratur von Radford et al. [11], um die Stabilität des adversarialen Prozesses zu gewährleisten. Zur Optimierung beider Netzwerke dient der Adam-Optimierer mit einer Lernrate von 2X10^-4 und einem angepassten Impulsparameter von $\beta_1 = 0{,}5$. Das Training erfolgte über 50 Epochen für MNIST sowie 100 Epochen für CIFAR-10 bei einer Batch-Größe von 128. Die kontinuierliche visuelle Überwachung durch einen festen Rauschvektor sicherte dabei die Bestimmung des optimalen Konvergenzpunktes für die spätere Merkmalsextraktion.

3.5.4 Generierung und Speicherung der Daten

Nach Abschluss des adversarialen Trainings wird der Generator isoliert und in den Inferenz-Modus versetzt, um die für die Arbeit benötigten Testdaten zu erzeugen. Dieser Prozess umfasst die systematische Erstellung von zwei Datentypen:

- Synthetische Bilddaten: Aus 100-dimensionalen Rauschvektoren z, die einer Standardnormalverteilung folgen, generiert das Modell neue Stichproben. Diese werden vom ursprünglichen Wertebereich [-1, 1] auf das Intervall [0, 1] denormalisiert, um sie mit der Rohpixel-Baseline kompatibel zu machen.
- Latent-Vektoren: Parallel dazu werden die zugrunde liegenden Vektoren z als NumPy-Dateien gespeichert. Diese dienen als zusätzliche Repräsentationsebene, um die Abgrenzbarkeit der gelernten Strukturen im latenten Raum zu untersuchen.

Die Speicherung erfolgt dabei automatisiert in Batches von beispielsweise 10.000 Samples.

3.5.5 Trainingsverlauf und Validierung

Zunächst erfolgt die Analyse des GAN-Trainings für die MNIST Datensatz dessen Entwicklung lässt sich besonders gut durch den Vergleich von drei markanten Zeitpunkten, während der 50 Trainingsepochen verdeutlichen:

In der Initialphase bei Epoche 1 zeigt die Ausgabe ein vollkommen unstrukturiertes Bildrauschen, da das Netzwerk zu diesem Zeitpunkt noch keine Informationen über die räumliche Verteilung der MNIST-Daten besitzt.

Abbildung 12 MNIST Epoche 1

Nach 25 Epochen lässt sich eine signifikante Lernkurve dokumentieren: Der Generator beginnt, die grundlegende Topologie der Ziffern zu erfassen. Es entstehen schemenhafte Formen, bei denen bereits zwischen kreisförmigen Strukturen (wie bei der „0“ oder „8“) und vertikalen Linien (wie bei der „1“) unterschieden werden kann, wenngleich die Ränder noch diffus wirken.

Abbildung 13 MNIST Epoche 25

Mit Erreichen der Epoche 50 ist die Konvergenz des Minimax-Spiels zwischen Generator und Diskriminator weit fortgeschritten. Die generierten Ziffern weisen nun scharfe Konturen und eine klare Abgrenzung zum Hintergrund auf, was die erfolgreiche Nutzung der auf 32x32 Pixel erhöhten Auflösung bestätigt. Dass in den Previews eine hohe Vielfalt an Schreibstilen sichtbar ist, ohne dass sich Muster identisch wiederholen, dient als Indiz für ein stabiles Training ohne *Mode Collapse*.

Abbildung 14 MNIST Epoche 50

Die visuelle Analyse des Lernfortschritts für den CIFAR-10-Datensatz verdeutlicht die im Vergleich zu MNIST deutlich höhere Komplexität, da der Generator neben geometrischen Formen auch präzise Farbkompositionen und Texturen koordinieren muss. In der Initialphase bei Epoche 1 erzeugt das Netzwerk lediglich unstrukturiertes Farbrauschen ohne jegliche Objektdefinition.

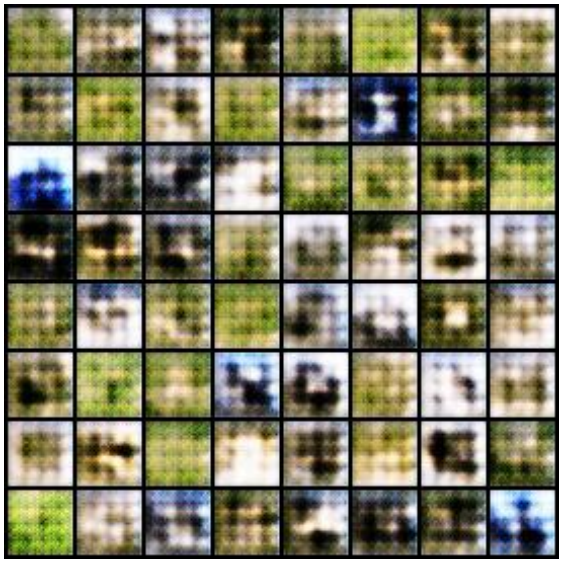

Abbildung 15 CIFAR-10 Epoche 1

Nach 35 Epochen lässt sich eine frühe Lernphase dokumentieren, in der das Modell beginnt, klassenspezifische Hintergründe zu assoziieren – wie beispielsweise blaue Flächen für maritime Motive –, während die eigentlichen Konturen noch diffus wirken.

Abbildung 16 CIFAR-10 Epoche 35

Dieser Prozess festigt sich bis zur Epoche 70 erheblich, wobei Umrisse von Fahrzeugen und Tieren zunehmend differenzierbar werden und die Farbübergänge an Natürlichkeit gewinnen.

Abbildung 17 CIFAR-10 Epoche 70

Das abschließende 8x8-Stichprobenraster der Epoche 100 belegt schließlich eine hohe Variabilität sowie eine stabile Konvergenz ohne Anzeichen eines Mode Collapse. Trotz einer konstruktionsbedingten Unschärfe im Vergleich zu den Originaldaten bleiben, die für das Clustering entscheidenden strukturellen Merkmale erhalten, was die ausreichende Repräsentationskraft des 100-dimensionalen Latent-Spaces für die dreidimensionale RGB-Synthese bestätigt.

Abbildung 18 CIFAR- 10 Epoche 100

3.6 Clustering-Algorithmen und Parameterwahl

Nachdem die verschiedenen Repräsentationsebenen extrahiert wurden, erfolgt im nächsten Schritt die Anwendung der Clustering-Verfahren. Zur Gewährleistung der Vergleichbarkeit werden alle Datenrepräsentationen zunächst mittels **StandardScaler** normalisiert. Dies stellt sicher, dass Merkmale mit unterschiedlichen Wertebereichen die Distanzberechnungen nicht verzerren.

Um eine objektive und umfassende Analyse der Datenstrukturen zu ermöglichen, kommen drei komplementäre Clustering-Ansätze zum Einsatz:

Algorithmus	Kernparameter	Methodik / Zielsetzung
K-Means	8, 10 und 12	Zentroid-basierte Baseline zur Prüfung der linearen Separierbarkeit.
Hierarchisch	8, 10,12 und 15	Ward-Linkage zur Varianzminimierung; Subsampling auf 30.000 Instanzen.

DBSCAN	min_samples = 15 Epsilon ist variabel	Dichte-basiert; Identifikation von Rauschen und unregelmäßigen Clustern.

Tabelle 4 Clustering-Algorithmen und Parameterwahl

Die Die Wahl der Clusterzahlen orientiert sich an der bekannten Klassenanzahl der Datensätze (10 Klassen), wobei Abweichungen (8,12 und15) bewusst gewählt wurden, um die Stabilität der Algorithmen gegenüber Über- oder Untersegmentierung zu testen.

Die Bewertung der Clustering-Güte erfolgt durch eine Kombination aus quantitativen Metriken und qualitativen Analysen:

- Quantitative Metriken: Der Adjusted Rand Index (ARI) und die Normalized Mutual Information (NMI) messen die Übereinstimmung mit den Ground-Truth-Labels, während der Silhouette Score die interne Kompaktheit der Cluster ohne externe Labels bewertet.
- Qualitative Visualisierung: Heatmaps der Konfusionsmatrizen visualisieren die Zuordnung der tatsächlichen Klassen zu den identifizierten Clustern und machen Fehlklassifikationen sowie semantische Überschneidungen sichtbar.

4 Experimente und Ergebnisse

4.1 Baseline: Clustering auf Rohdaten

Um eine fundierte Bewertung der späteren Merkmalsextraktion zu ermöglichen, wurden zunächst die Rohdaten beider Datensätze (MNIST und CIFAR-10) nach einer PCA-Reduktion geclustert

4.1.1 K-Means auf MNIST (Rohdaten)

Die Auswertung der K-Means-Ergebnisse zeigt, dass die Struktur der MNIST-Daten im PCA-reduzierten Pixelraum grundsätzlich erfasst wird. Bei einer Clusteranzahl von k=10, welche der tatsächlichen Klassenanzahl entspricht, werden ein ARI von 0,3120 sowie eine NMI von 0,4284 erzielt. Diese Ergebnisse bilden die Baseline für die weiteren Experimente und dienen als Referenz für den Vergleich mit den Deep-Learning-basierten Repräsentationen. Der niedrige Silhouette-Score von 0,0680 deutet jedoch darauf hin, dass die Klassen im 50-dimensionalen PCA-Raum nur unzureichend separierbar sind. Die resultierenden Cluster zeigen eine deutliche Überlappung, sodass die Gruppengrenzen unscharf ausgeprägt sind.

Anzahl Cluster (k) MNIST	ARI	NMI	Silhouette Score
8	0,2864	0,3993	0,0586
10 (Optimum)	0,3120	0,4284	0,0680
12	0,3126	0,4367	0,0773

Tabelle 5 Clustering Metriken - MNIST

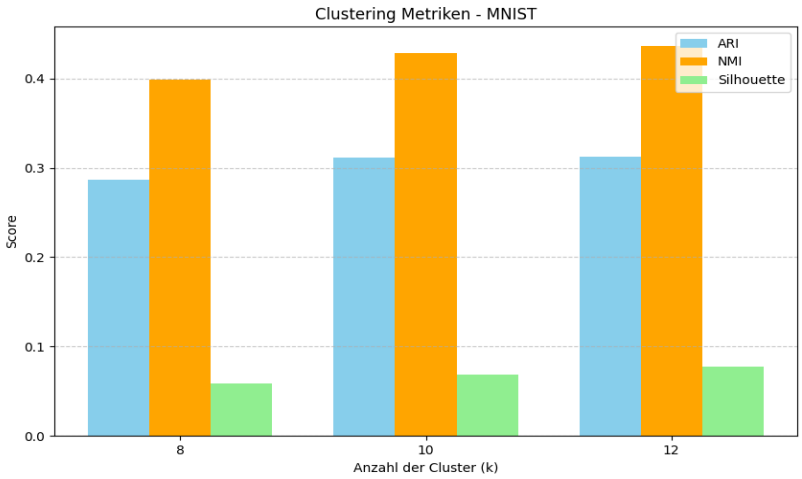

Abbildung 19 Clustering Metriken - MNIST

Die qualitative Analyse der Konfusionsmatrix gibt tiefere Einblicke in die Trennbarkeit der einzelnen Ziffern. Besonders deutlich sticht **Cluster 1** hervor, der mit 6.537 korrekt zugewiesenen Instanzen fast ausschließlich die Ziffer „1“ repräsentiert. Auch die Ziffern „0“ (Cluster 2) und „6“ (Cluster 8) lassen sich vergleichsweise stabil gruppieren. Im Gegensatz dazu offenbart die Heatmap erhebliche Schwierigkeiten bei topologisch ähnlichen Zeichen: Die Ziffern „4“, „7“ und „9“ verteilen sich diffus über mehrere Cluster-IDs, was die Grenzen der rein pixelbasierten Merkmalsextraktion ohne tiefe Repräsentationslernen-Verfahren aufzeigt.

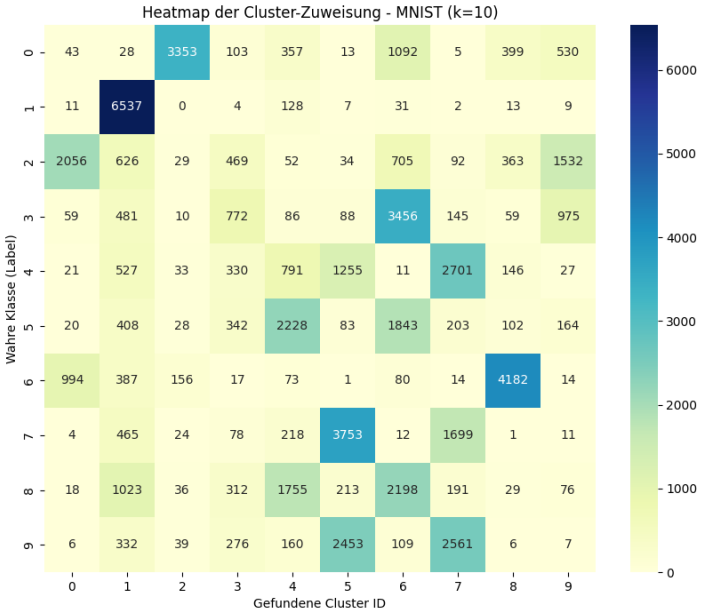

Abbildung 20 Heatmap der Cluster-Zuweisung – MNIST (k=10)

4.1.2 K-Means auf CIFAR-10 (Rohdaten)

Die Auswertung der K-Means-Baseline für CIFAR-10 verdeutlicht die immense Komplexität von Farbbildern im Vergleich zu einfachen Graustufen-Ziffern. Die quantitativen Ergebnisse zeigen bei der natürlichen Klassenanzahl von k=10 einen ARI von lediglich 0,0419 und einen NMI von 0,0802.

Anzahl Cluster (k) CIFAR10	ARI	NMI	Silhouette Score
8	0,0395	0,0735	0,0606
10 (Optimum)	**0,0419**	**0,0802**	**0,0591**
12	0,0414	0,0826	0,0477

Tabelle 6 Clustering Metriken – CIFAR-10

Diese niedrigen Werte belegen, dass im rohen Pixel- bzw. PCA-Raum nahezu keine semantisch sinnvolle Trennung der Objektklassen möglich ist. Auch der Silhouette Score von 0,0591 bestätigt, dass die Datenpunkte im 200-dimensionalen Raum keine kompakten, isolierten Cluster bilden, sondern eine stark überlappende, diffuse Wolke darstellen.

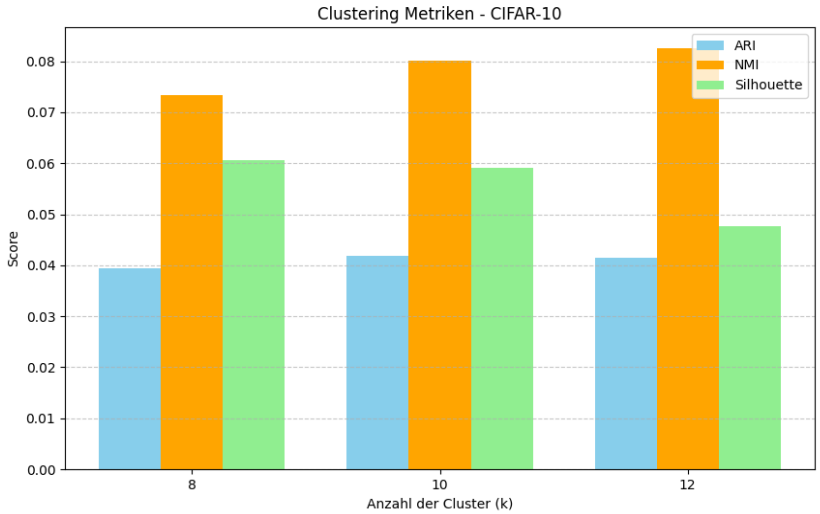

Abbildung 21 Clustering Metriken – CIFAR-10

Qualitativ bestätigt die Heatmap die mangelnde Trennschärfe auf der Rohpixel-Ebene: Während MNIST klare Schwerpunkte zeigt, verteilen sich die CIFAR-10-Klassen fast gleichmäßig über alle Cluster-IDs. Punktuelle Konzentrationen, wie bei der Klasse „Schiff“ (Label 8) in Cluster 1, bieten keine verlässliche Basis für eine Objektkategorisierung. Dieses Ergebnis unterstreicht die Notwendigkeit für fortgeschrittene Repräsentationslernen-Verfahren wie Autoencoder oder GANs, um semantisch relevante Merkmale aus komplexen Bilddaten zu extrahieren.

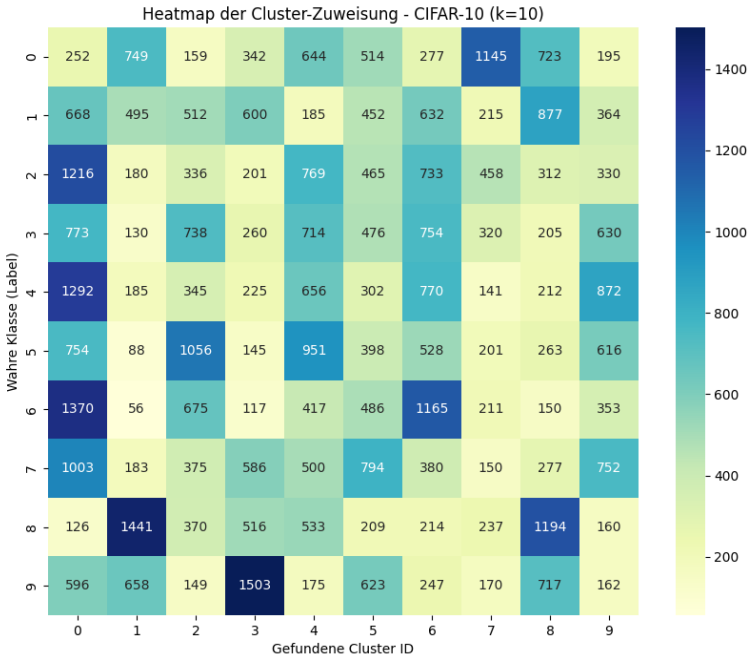

Abbildung 22 Heatmap der Cluster-Zuweisung – CIFAR-10 (k=10)

4.1.3 DBSCAN auf MNIST (Rohdaten)

Die Auswertung des Dichtebasierten Clusterings zeigt ein lokales Optimum der Trenngüte bei ε =7,0, wobei der ARI jedoch bei einem niedrigen Wert von 0,0583 stagniert. Besonders prägnant sind die negativen Silhouette-Scores, die bei ε =7,0 einen Wert von -0,1956 aufweisen. Mathematisch belegt dies das Fehlen distinkter Dichtezentren: Die stark überlappende Struktur führt dazu, dass Instanzen im Durchschnitt näher an benachbarten Clustern liegen als an der eigenen Gruppe.

Epsilon (ϵ) für MNIST	Gefundene Cluster	ARI	NMI	Silhouette Score
4,0	2	0,0422	0,2200	-0,1556
5,0	14	0,0467	0,2166	-0,2495
6,0	12	0,0542	0,1719	-0,2355
7,0 (Optimum)	**6**	**0,0583**	**0,1315**	**-0,1956**
8,0	4	0,0511	0,1145	-0,1056

Tabelle 7 DBSCAN Metriken – MNIST (FULL DATASET)

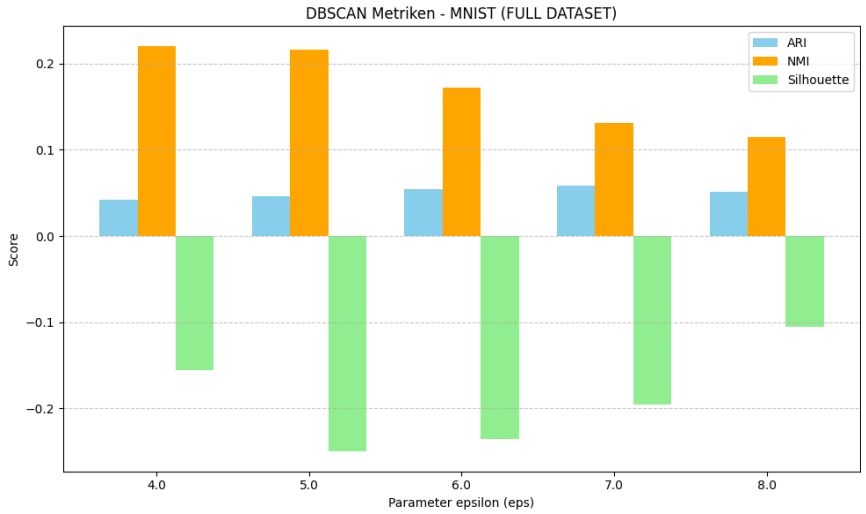

Abbildung 23 DBSCAN Metriken – MNIST (FULL DATASET)

Die Heatmap bei ε=7,0 offenbart die Bildung zweier massiver Cluster, welche die semantische Varianz der Ziffern fast vollständig ignorieren. Trotz eines technisch bedingten Index-Offsets auf der Y-Achse (Ziffer „0“ entspricht Zeile 1) wird deutlich, dass nahezu alle Klassen in den IDs 0 und 1 verschmelzen. Lediglich die Ziffer „1“ erfährt in Cluster 1 eine geringfügige Isolation, bleibt dort jedoch massiv mit anderen Klassen wie der „0“ oder „9“ vermischt. Zusammenfassend scheitert DBSCAN an der zu homogenen

Punktdichte des Pixelraums, wodurch semantisch relevante Lücken zwischen den Klassen unkenntlich bleiben.

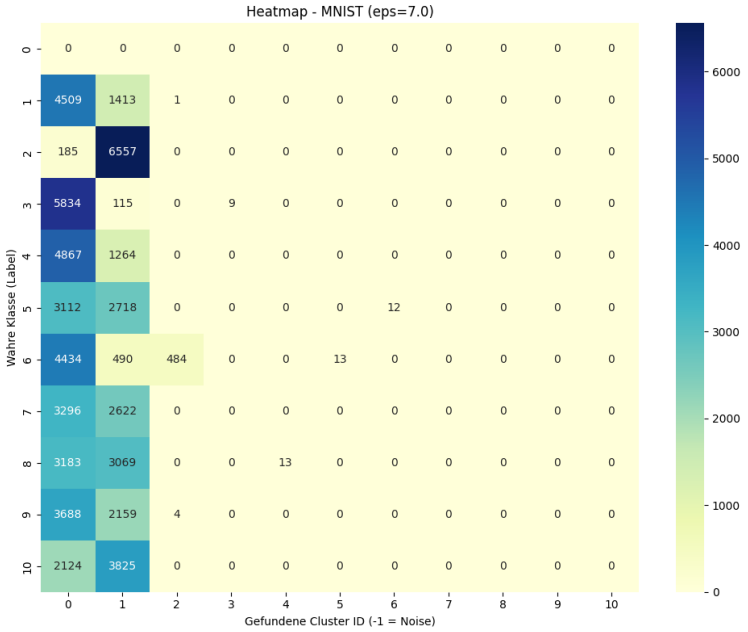

Abbildung 24 Heatmap – MNIST (eps=7.0)

4.1.4 DBSCAN auf CIFAR-10 (Rohdaten)

Die Untersuchung des CIFAR-10-Daten zeigt eine nahezu vollständige Abwesenheit separierbarer Dichtestrukturen. Die quantitative Auswertung liefert beiε =35,0 zwar das rechnerische Optimum, wobei die Werte für den ARI (0,0324) und den NMI (0,0601) jedoch faktisch eine Ununterscheidbarkeit von einer Zufallsklassifikation belegen. Die negativen Silhouette-Scores (bis zu -0,1322) belegen eine diffuse, stark überlappende Datenstruktur ohne klare klassenspezifische Konzentrationen.

Epsilon (ϵ)	Gefundene Cluster	ARI	NMI	Silhouette Score
25,0	2	0,0040	0,0351	-0,1322
30,0	2	0,0189	0,0577	-0,0712
35,0 (Optimum)	**3**	**0,0324**	**0,0601**	**-0,0220**
40,0	1	0,0143	0,0353	-
45,0	1	0,0023	0,0163	-

Tabelle 8 DBSCAN Metriken – CIFAR-10 (FULL DATASET)

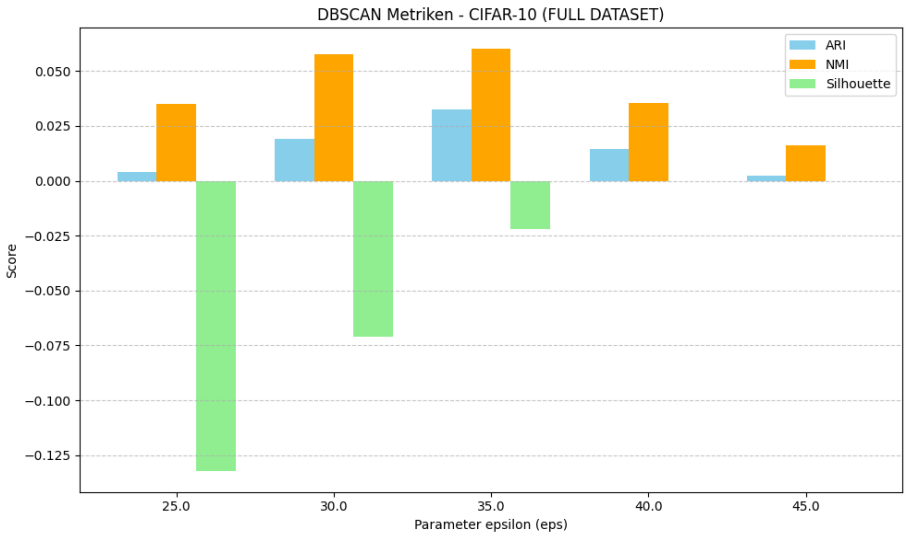

Abbildung 25 DBSCAN Metriken – CIFAR-10 (FULL DATASET)

Die Heatmap für CIFAR-10 ε =35, spiegelt das strukturelle Scheitern wie bei MNIST wieder: Die Bilddaten kollabieren fast vollständig in zwei massive Super-Cluster (IDs 0 und 1). Auch hier ist der technische Index-Offset auf der Y-Achse zu beobachten, welcher die Klassen 0 bis 9 in die Zeilen 1 bis 10 verschiebt.

Selbst markante Klassen wie „LKW“ (Zeile 10) entwickeln keine eigenständige dichte basierte Identität, sondern verteilen sich ohne semantische Trennung auf die beiden Hauptcluster. Dies belegt, dass die hohe Varianz der Bildinhalte (Hintergründe, Posen) die euklidischen Distanzen im PCA-Raum so stark dominiert, dass DBSCAN auf den Rohdaten keine klassenspezifischen Merkmale isolieren kann.

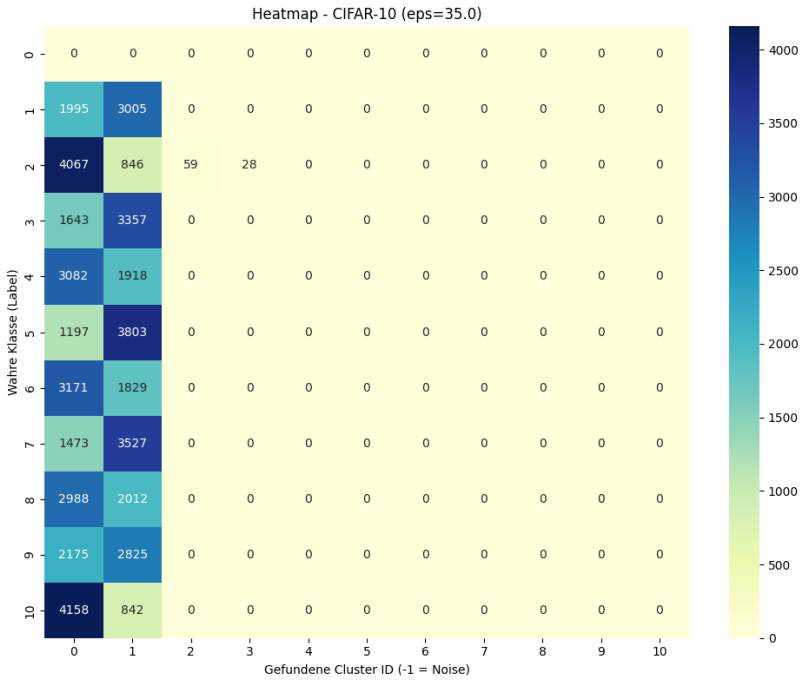

Abbildung 26 Heatmap – CIFAR-10 (eps=35.0)

4.1.5 Hierarchisches Clustern für MNIST Rohdaten

Die Die Untersuchung mittels hierarchischem Clustering (Ward-Linkage) belegt eine deutliche Steigerung der Trenngüte im Vergleich zur K-Means-Baseline. Während K-Means bei der Referenzgröße k=10 einen ARI von 0,3120 erzielt, übertrifft das hierarchische Verfahren diesen Wert bereits mit 0,3963. Das Maximum wird bei k=15 mit einem ARI von 0,4707 erreicht. Dieser kontinuierliche Anstieg der Metriken verdeutlicht, dass die Minimierung der Varianz innerhalb der Cluster feine Unterstrukturen und unterschiedliche Schreibstile im 50-dimensionalen PCA-Raum präziser erfasst als rein zentroid-basierte Ansätze.

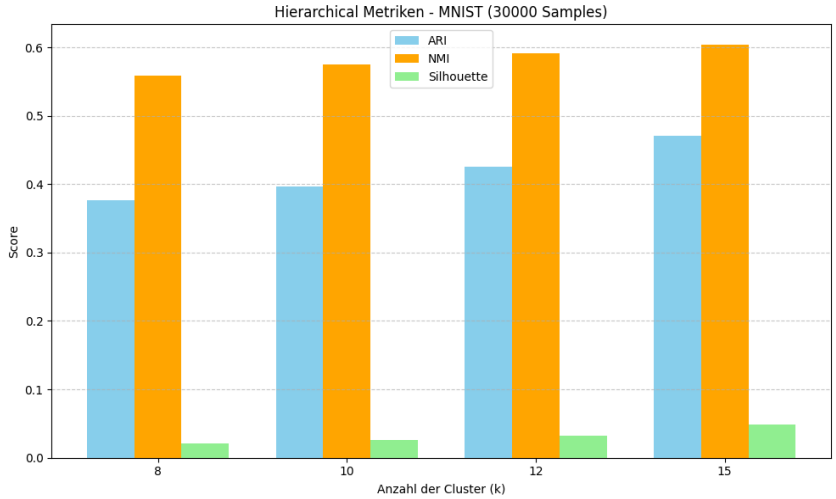

Abbildung 27 Hierarchical Metriken – MNIST (30k Samples)

Clusteranzahl (k) MNIST	ARI	NMI	Silhouette Score
8	0,3771	0,5592	0,0213
10 (Referenz)	**0,3963**	**0,5745**	**0,0255**
12	0,4258	0,5916	0,0318
15	0,4707	0,6034	0,0487

Tabelle 9 Hierarchical Metriken – MNIST (30k Samples)

Die Konfusionsmatrix bei k=15 belegt eine im Vergleich zu anderen Baselines gesteigerte Trennschärfe. Besonders deutlich isoliert der Algorithmus die Ziffer „1“ in Cluster 3 (3.288 Instanzen) sowie die „6“ in Cluster 0 (2.476 Instanzen). Die Verbesserung der Metriken bei höherem k resultiert maßgeblich aus der Aufspaltung strukturell komplexer Klassen: So verteilt sich die Ziffer „0“ primär auf die Cluster 8 und 12, während die „7“ in den Clustern 10 und 14 abgebildet wird. Diese Differenzierung deutet darauf hin, dass das Verfahren verschiedene Schreibstile erfolgreich als eigenständige strukturelle Gruppen identifiziert.

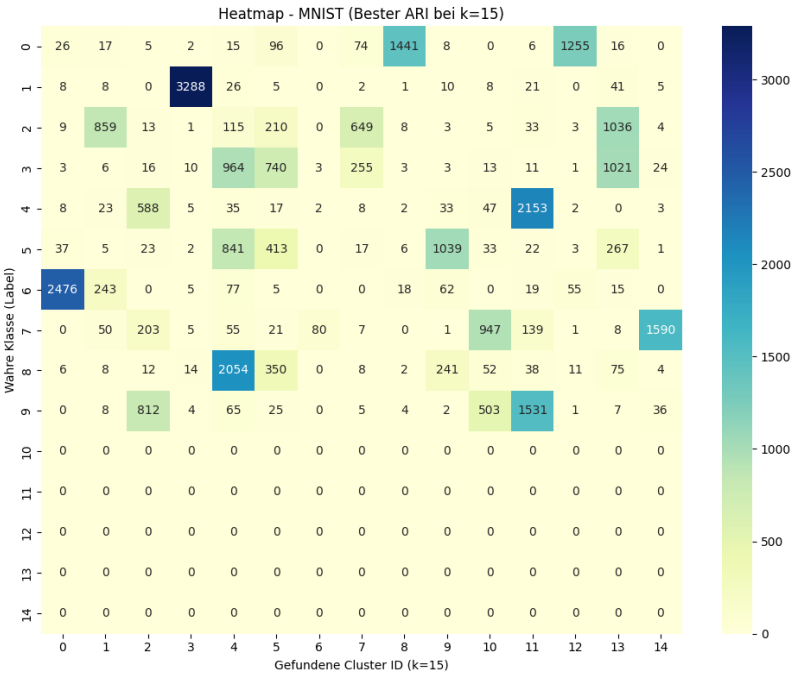

Abbildung 28 Heatmap – MNIST (Bester ARI bei k=15)

4.1.6 Hierarchisches Clustern für CIFAR-10 Rohdaten

Die hierarchische Clusteranalyse auf die CIFAR-10-Datren verdeutlicht, dass selbst varianzminimierende Verfahren wie das Ward-Linkage im hochdimensionalen PCA-Raum keine semantisch kohärenten Gruppen isolieren können. Die Ergebnisse zeigen bei der Referenzgröße von k=10 einen ARI von lediglich 0,0361 und eine NM) von 0,0692.

Clusteranzahl (k)	ARI	NMI	Silhouette Score
8	0,0303	0,0614	0,0270
10 (Referenz)	**0,0361**	**0,0692**	**0,0173**
12	0,0332	0,0733	0,0168
15	0,0314	0,0783	-0,0095

Tabelle 10 Hierarchical Metriken – CIFAR-10 (30k Samples)

Der bei k=15 negative Silhouette-Score (-0,0095) belegt die für CIFAR-10-Rohdaten typische strukturelle Instabilität, bei der Datenpunkte im Durchschnitt näher an benachbarten Clustern liegen als am eigenen Zentrum.

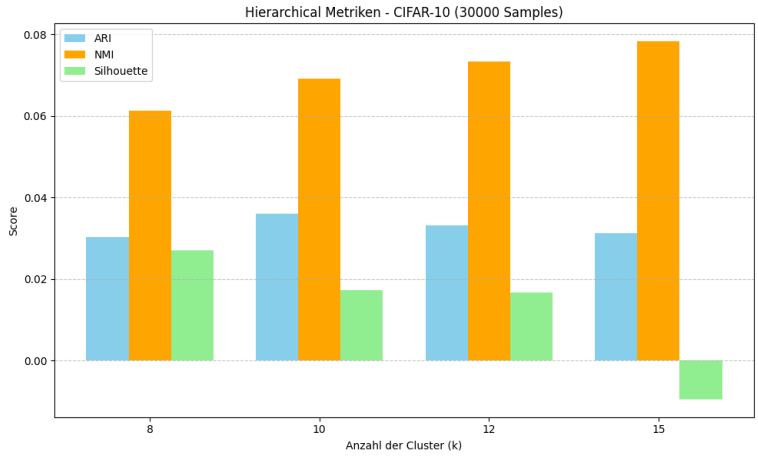

Abbildung 29 Hierarchical Metriken – CIFAR-10 (30k Samples)

Die Konfusionsmatrix für k=10 bestätigt die mangelnde Trennschärfe auf der Rohpixel-Ebene. Zwar lassen sich punktuelle Konzentrationen identifizieren, jedoch bleibt die Streuung über die restlichen Cluster hinweg sehr hoch. Die Cluster-IDs weisen somit kaum eine verlässliche Korrelation zu den tatsächlichen Objektkategorien auf. Dies belegt, dass die Varianzminimierung bei Ward im rohen Merkmalsraum nicht ausreicht, um die komplexe, nicht-lineare Struktur von Farbbildern erfolgreich aufzulösen.

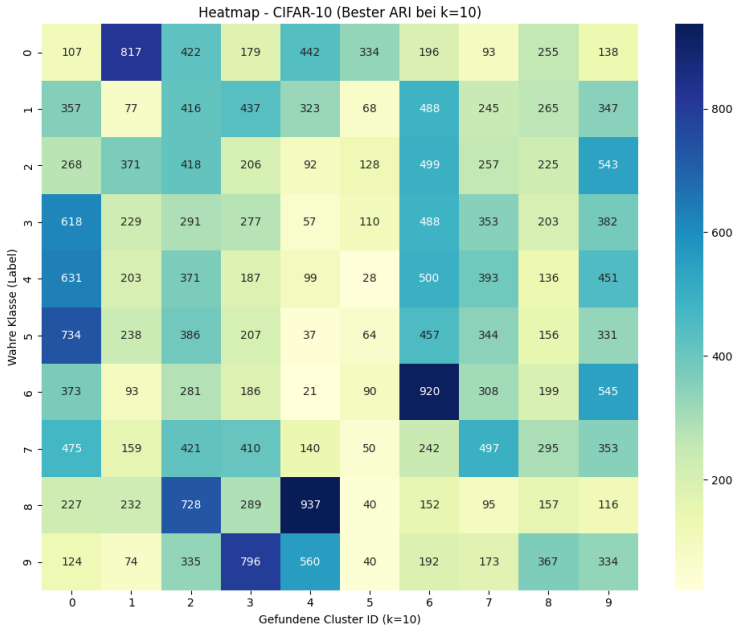

Abbildung 30 Heatmap – CIFAR-10 (Bester ARI bei k=10)

4.2 Clustering auf Autencoder-Latens

In diesem Abschnitt wird untersucht, inwieweit die durch die Autoencoder gelernten, niedrigdimensionalen Repräsentationen die Trenngüte der Clustering-Verfahren verbessern. Im Gegensatz zu den Rohdaten arbeiten die Algorithmen hier auf dem

„Bottleneck“ der Netzwerke – also auf 32 Dimensionen bei MNIST und 128 Dimensionen bei CIFAR-10.

4.2.1 K-Means mit AE-Latens auf MNIST

Die Analyse der K-Means-Ergebnisse auf Basis der Autoencoder-Merkmale zeigt eine leichte Verbesserung gegenüber der rein statistischen Baseline (PCA). Bei der Ziel-Clusteranzahl von k=10 erreicht das Modell einen ARI von 0.3179 und eine NMI von 0.4404. Interessanterweise steigt die Clustering-Güte bei einer Erhöhung der Clusteranzahl weiter an und erreicht bei k=14 mit einem ARI von 0.3547 ihren Bestwert. Dies deutet darauf hin, dass der Autoencoder im Latent Space verschiedene Schreibstile einzelner Ziffern als separate Untergruppen abbildet.

Clusteranzahl (k)	ARI	NMI	Silhouette Score
10 **(Klassenanzahl)**	0,3179	0,4404	0,0969
12	0,3358	0,4689	0,1001
14 **(Bestwert ARI)**	**0,3547**	0,4974	0,1022
16	0,3448	**0,5069**	**0,1035**

Tabelle 11 AE-Clustering Metriken – MNIST AE-KMeans

Der Silhouette-Score, der bei k=10 bei 0.0969 liegt (im Vergleich zu 0.0680 bei der Baseline), verdeutlicht, dass die Repräsentationen im kompakten 32-dimensionalen Merkmalsraum zwar etwas dichter beieinander liegen, die Clustergrenzen aber weiterhin stark überlappen.

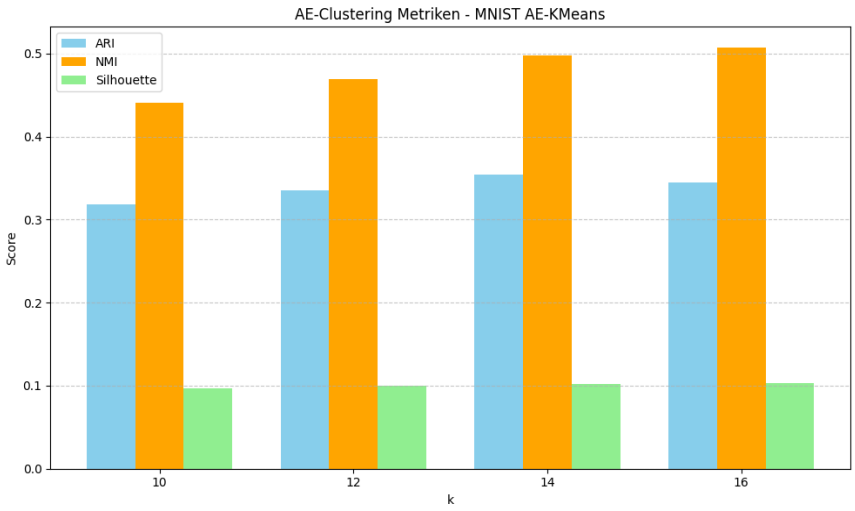

Abbildung 31 AE-Clustering Metriken – MNIST AE-KMeans

Die qualitative Analyse der Konfusionsmatrix bestätigt, dass der Autoencoder primär visuelle und topologische Ähnlichkeiten lernt. Während geometrisch eindeutige Formen (0, 3 oder 6) stabil gruppiert werden, offenbart die Heatmap bei Zeichen mit ähnlichen Grundstrukturen (4,5 oder 8) deutliche Trennungsprobleme. Diese verteilen sich diffus

über mehrere Cluster-IDs, was die Grenzen der rein rekonstruktionsbasierten Logik verdeutlicht: Da das Modell auf die Minimierung des Rekonstruktionsfehlers (MSE) und nicht auf die semantische Abgrenzung optimiert ist, führen komplexe oder ähnliche Formmerkmale im Latent Space zwangsläufig zu unscharfen Clustergrenzen.

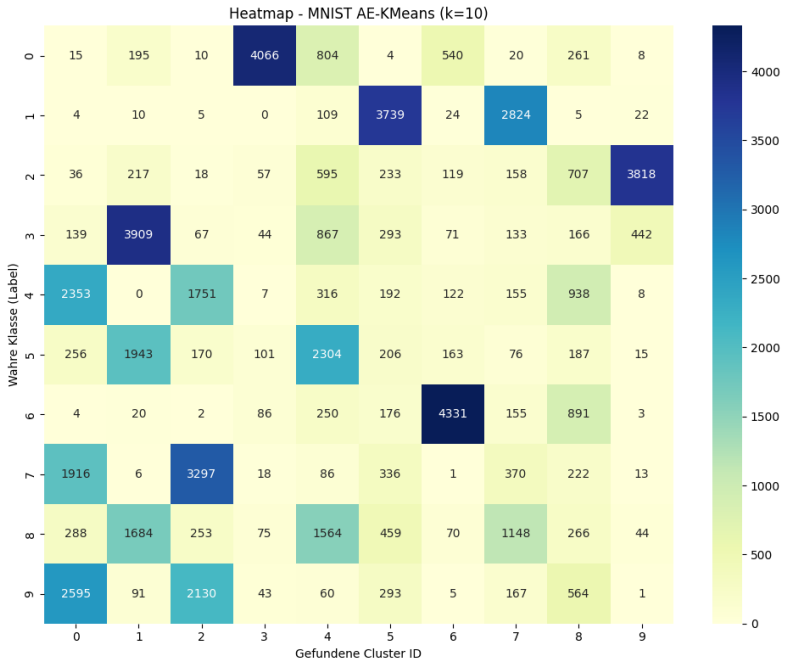

Abbildung 32 Heatmap AE-KMeans (k=10)

4.2.2 K-Means mit AE-Latens auf CIFAR_10

Die Auswertung zeigt, dass selbst die Merkmalsextraktion mittels CAE bei komplexen Bilddaten wie CIFAR-10 an ihre Grenzen stößt. Die erzielten Metriken liegen deutlich unter den Werten von MNIST. Die quantitativen Ergebnisse belegen die enorme Komplexität des CIFAR-10-Datensatzes. Mit einem ARI von 0,0476 bei k=1 zeigt das Clustering nur eine minimale Übereinstimmung mit den tatsächlichen Objektklassen. Im Gegensatz zu MNIST führt eine Erhöhung der Clusteranzahl hier zu keiner signifikanten Verbesserung des ARI, während der Silhouette-Score (0,0261) fast den Nullpunkt erreicht. Dies ist ein klares Indiz dafür, dass die Repräsentationen im Latent Space keine separaten Gruppen bilden, sondern als eine fast untrennbare Datenwolke vorliegen.

Clusteranzahl (k)	ARI	NMI	Silhouette Score
10 **(Referenz)**	**0,0476**	0,0943	**0,0261**
12	0,0456	0,0947	0,0236
14	0,0443	0,0956	0,0126
16	0,0459	**0,1039**	0,0102

Tabelle 12 AE-Clustering Metriken – CIFAR-10 AE-KMeans

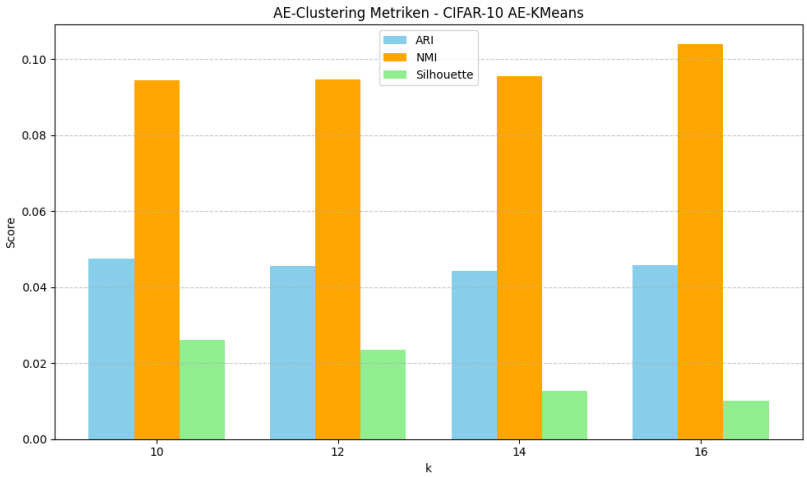

Abbildung 33 AE-Clustering Metriken – CIFAR-10 AE-KMeans

Die qualitative Analyse der Konfusionsmatrix verdeutlicht das diffuse Verhalten des Algorithmus. Während bei MNIST klare Schwerpunkte erkennbar waren, verteilt sich bei CIFAR-10 fast jede wahre Klasse über nahezu alle gefundenen Cluster-IDs. Einzig bei Klasse 9 (Cluster 5: 1.688 Instanzen) und Klasse 6 (Cluster 3: 1.486 Instanzen) zeigen sich leichte Tendenzen einer Gruppierung.

Das Hauptproblem liegt in der rekonstruktionsbasierten Logik: Der Convolutional Autoencoder optimiert auf die Wiederherstellung von Pixeln (Farben, grobe Formen). Bei natürlichen Bildern fließen daher zu viele Informationen über Hintergründe (z. B. blauer Himmel oder grüne Wiesen) in den Latent Space ein. Da diese Merkmale oft klassenübergreifend auftreten, fehlt es den Vektoren an der nötigen semantischen Schärfe, um komplexe Objekte wie „Hunde“ von „Katzen“ oder „Flugzeuge“ von „Schiffen“ rein dichte- oder schwerpunktbasiert zu trennen.

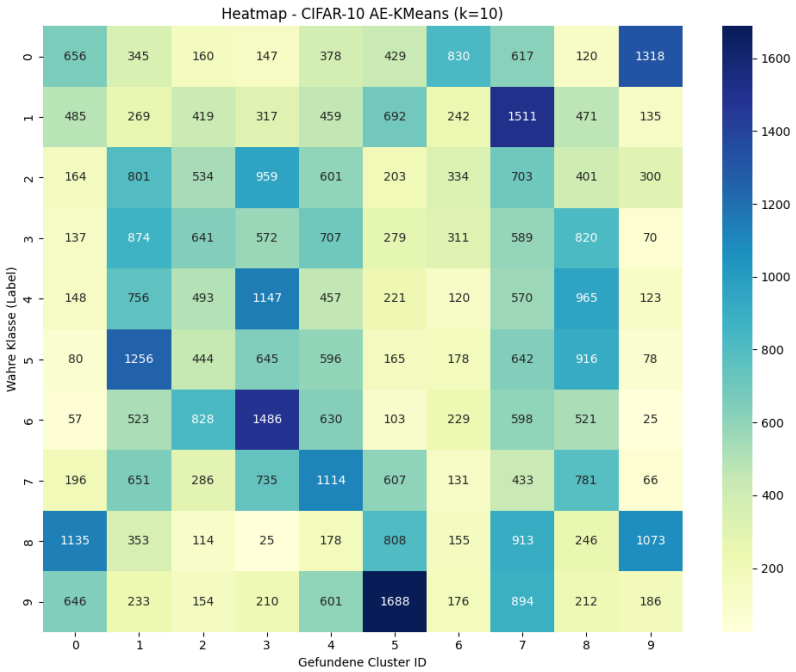

Abbildung 34 Heatmap – CIFAR-10 AE-Kmeans (k=10)

4.2.3 DBSCAN mit AE-Latens auf MNIST

Die Untersuchung des dichte-basierten Clusterings auf Basis der Autoencoder-Latents zeigt ein lokales Optimum der Trenngüte bei ε= 7,0. Im Vergleich zur Baseline lässt sich eine moderate Steigerung des ARI von 0,0583 auf 0,0760 sowie eine deutliche Zunahme der NMI auf 0,2878 verzeichnen. Dennoch verbleibt der Silhouette-Score mit -0,1404 in einem deutlich negativen Bereich. Zwar stellt dies eine geringfügige Verbesserung gegenüber dem Baseline-Wert (-0,1956) dar, belegt jedoch mathematisch, dass auch der Latent Space des Autoencoders keine hinreichend isolierten Dichtezentren ausprägt, wodurch Instanzen im Durchschnitt weiterhin näher an benachbarten Clustern liegen als an der eigenen Gruppe.

Epsilon (eps)	Clusteranzahl	ARI	NMI	Silhouette Score
5,0	3	0,0436	0,2267	-0,0954
7,0 (Bestwert)	**11**	**0,0760**	**0,2878**	-0,1404
9,0	10	0,0630	0,1524	-0,1349
11,0	1	0,0037	0,0283	0,0000

Tabelle 13 AE-DBSCAN Metriken - MNIST

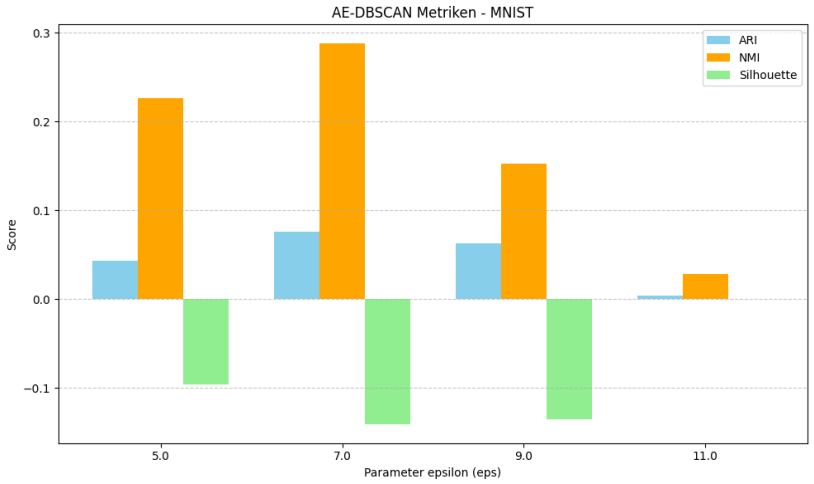

Abbildung 35 AE-DBSCAN Metriken - MNIST

Die Heatmap bei ε= 7,0 verdeutlicht, dass trotz 11 identifizierter Cluster weiterhin ein „Hauptcluster“ (ID 0) dominiert, welcher die semantische Varianz weitgehend ignoriert. Im Vergleich zur Baseline zeigen sich jedoch erste klassenspezifische Abspaltungen, etwa bei den Ziffern **„2“ (Cluster 1)** und **„7“ (Cluster 6)**. DBSCAN profitiert zwar von der rauschreduzierten Merkmalsextraktion des Autoencoders, scheitert jedoch an der zu homogenen Punktdichte des Raums. Da die Optimierung auf den Rekonstruktionsfehler fließende Übergänge erzeugt, fehlen die für DBSCAN essenziellen Dichtelücken zwischen den semantischen Gruppen.

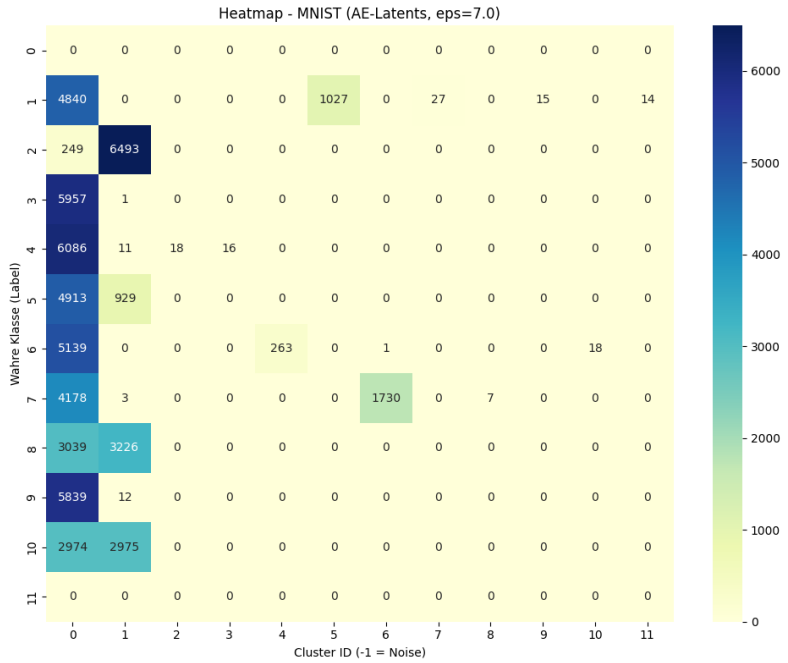

Abbildung 36 Heatmap – MNIST (AE-LAtents, eps=7.0)

4.2.4 DBSCAN mit AE-Latens auf CIFAR-10

Die Untersuchung der CIFAR-10-Latents zeigt eine nahezu vollständige Abwesenheit separierbarer Dichtestrukturen. Die quantitative Auswertung liefert bei ε=5,0 das rechnerische Optimum, wobei die Werte für den ARI (0,0257) und den NMI (0,0500) faktisch eine Ununterscheidbarkeit von einer Zufallsklassifikation belegen. Im Vergleich zur Baseline (-0,1322) ist der Silhouette-Score mit 0,0457 zwar minimal positiv, verbleibt jedoch auf einem Niveau, das eine diffuse und stark überlappende Datenstruktur ohne klare klassenspezifische Konzentrationen widerspiegelt.

Epsilon (eps)	Clusteranzahl	ARI	NMI	Silhouette Score
3,0	1	0,0014	0,0234	0,0000
4,0	2	0,0198	0,0577	-0,0567
5,0 (Bestwert)	**2**	**0,0257**	**0,0500**	**0,0457**
6,0	1	0,0025	0,0159	0,0000

Tabelle 14 AE-DBSCAN Metriken – CIFAR-10

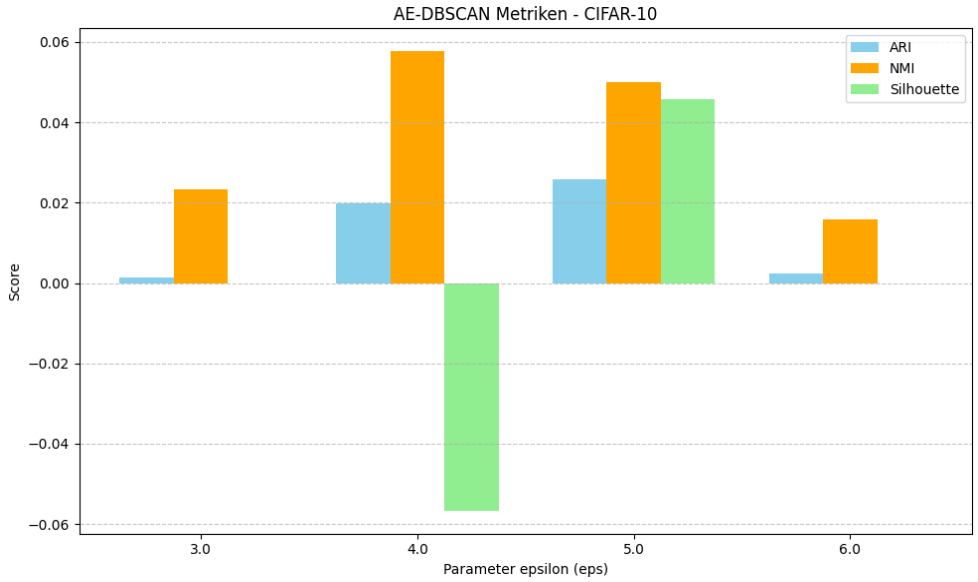

Abbildung 37 AE-DBSCAN Metriken – CIFAR-10

Die Heatmap für ε=5,0 spiegelt das strukturelle Scheitern der Baseline wider: Die Bilddaten kollabieren nahezu vollständig in zwei massive Super-Cluster (IDs 0 und 1), wodurch die semantische Trennung ausbleibt. Analog zur Baseline verteilen sich selbst markante Klassen wie „LKW“ (Zeile 10) diffus auf diese Hauptcluster. Dies belegt, dass die Varianz der Bildinhalte (Hintergründe, Posen) die euklidischen Distanzen im Latent Space so stark dominiert, dass DBSCAN trotz der Autoencoder-Kompression keine klassenspezifischen Merkmale isolieren kann.

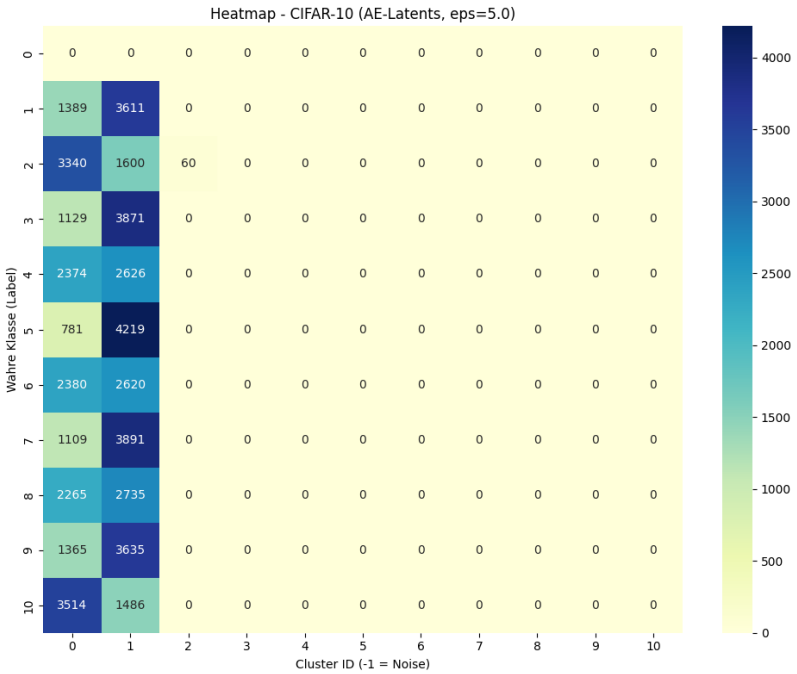

Abbildung 38 Heatmap – CIFAR-10 (AE-Latents, eps=5.0)

4.2.5 Hierarchisches mit AE-Latens auf MNIST

Die Ergebnisse belegen eine massive Steigerung der Trenngüte im Vergleich zur PCA-Baseline und zum K-Means-Verfahren auf AE-Latents. Während das hierarchische Verfahren in der Baseline (PCA) bei k=10 einen ARI von 0,3963 erzielt, steigert sich dieser Wert auf den Autoencoder-Repräsentationen auf 0,5035. Das absolute Maximum wird bereits bei k=8 mit einem ARI von 0,5167 erreicht. Dieser deutliche Sprung verdeutlicht, dass die Minimierung der Varianz innerhalb der Cluster (Ward-Linkage) die vom Autoencoder gelernten kompakten Strukturen wesentlich effektiver nutzt als rein zentroid-basierte Ansätze, die auf denselben Daten nur einen ARI von 0,3179 erzielten.

Clusteranzahl (k)	ARI	NMI	Silhouette Score
8 **(Optimum ARI)**	**0,5167**	0,6722	**0,0743**
10 **(Referenz)**	0,5035	0,6714	0,0566
12	0,5113	0,6821	0,0669
14	0,4771	0,6753	0,0667
16	0,4875	**0,6866**	0,0742

Tabelle 15 AE-Clustering Metriken – MNIST AE-Hierarchical

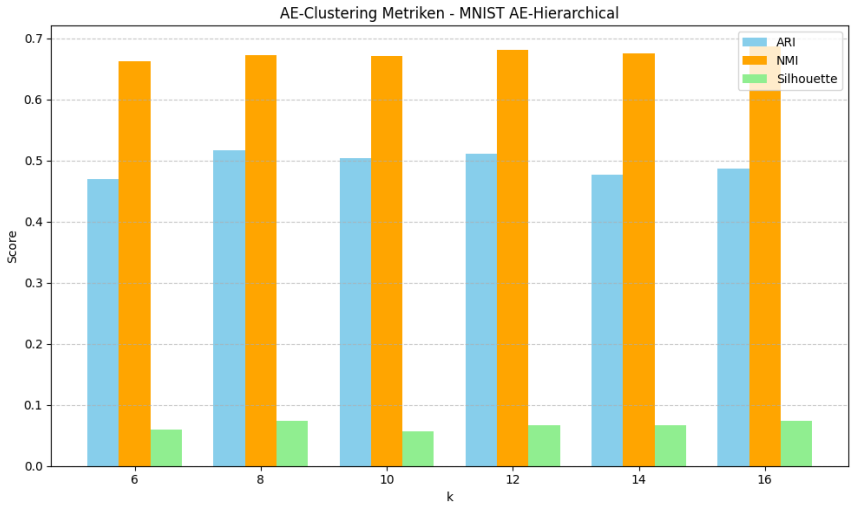

Abbildung 39 AE-Clustering Metriken – MNIST AE-Hierarchical

Die Heatmap für das Optimum bei k=8 belegt die hohe Trennschärfe der AE-Repräsentationen: Klassen wie „1“ (Cluster 0: 3.221), „6“ (Cluster 2: 2.871), „0“ (Cluster 4: 2.861) und „2“ (Cluster 7: 2.600) werden nahezu fehlerfrei isoliert. Die hohe Güte resultiert zudem aus der effektiven Bündelung topologisch ähnlicher Ziffern: Während die „3“ auf die Cluster 3 und 5 aufgeteilt wird, erfahren die Klassen „4“, „7“ und „9“ eine fokussierte Zusammenfassung in den Clustern 1 und 6. Dies beweist, dass das Ward-Linkage-Verfahren die gelernten „Form-Familien“ erfolgreich als eigenständige Gruppen identifiziert und trotz niedriger Silhouette-Scores eine semantisch präzisere Zuordnung ermöglicht als zentroid-basierte Ansätze.

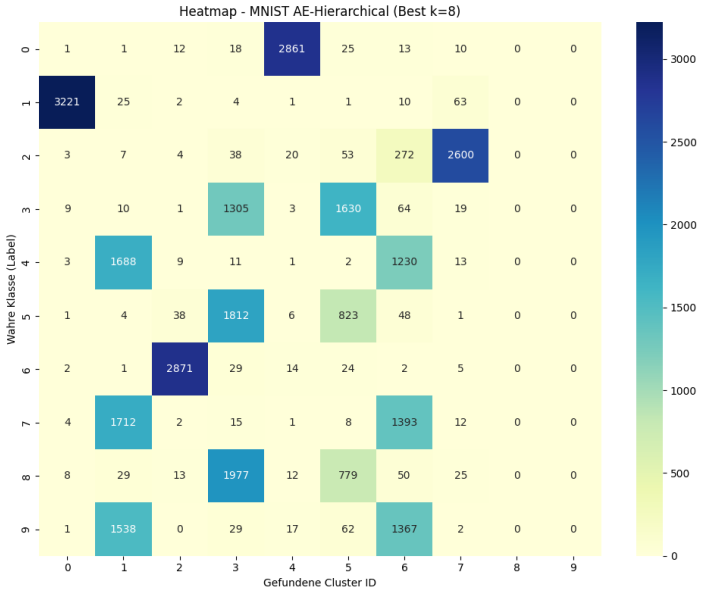

Abbildung 40 Heatmap - MNIST AE-Hierarchical (Best k=8)

Abbildung 37 XXX

4.2.6 Hierarchisches mit AE-Latens auf CIFAR-10

Die hierarchische Clustering auf den CAE-Latents belegt eine kleine Steigerung der Trenngüte gegenüber der Baseline. Während die PCA-Baseline bei k=10 einen ARI von 0,0361 erzielt, erreicht der CAE einen Wert von 0,0412. Das Maximum wird bei k=14 (ARI =0,0477) identifiziert. Trotz der leichten ARI-Steigerung bleibt die Repräsentation strukturell fragil, wie die ab k=14 negativen Silhouette-Werte belegen.

Clusteranzahl (k)	ARI	NMI	Silhouette Score
8	0,0408	0,0845	0,0105
10 **(Referenz)**	0,0412	0,0894	0,0013
12	0,0438	0,0900	0,0027
14 **(Optimum ARI)**	**0,0477**	0,0988	-0,0096
16	0,0452	**0,1035**	-0,0191

Tabelle 16 AE-Clustering Metriken - CIFAR-10 AE-Hierarchical

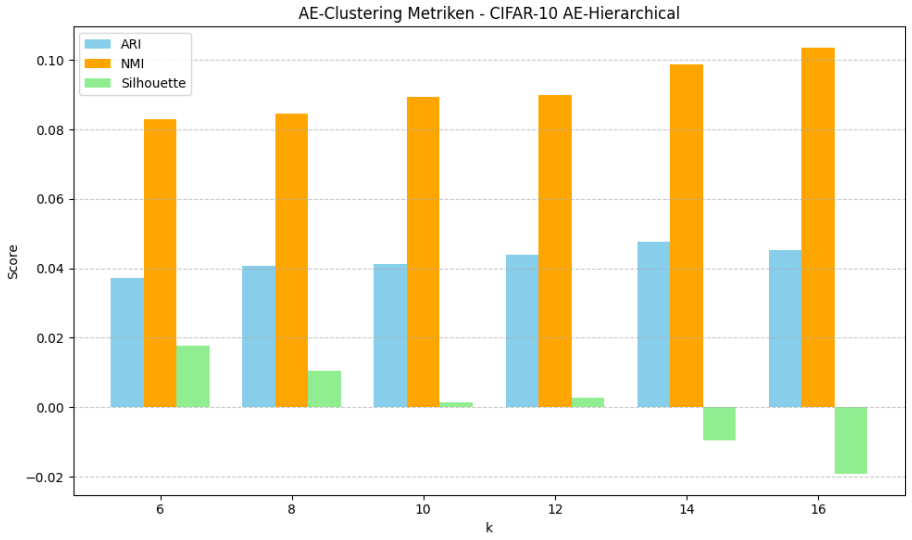

Abbildung 41 AE-Clustering Metriken - CIFAR-10 AE-Hierarchical

Die Heatmap für das Optimum bei k=14 bestätigt das bereits in der Baseline beobachtete Verhalten einer mangelnden Trennschärfe. Zwar lassen sich punktuelle Konzentrationen identifizieren – etwa bei Klasse 9 (Cluster 0: 1.278 Instanzen) –, jedoch bleibt die Streuung über die restlichen Cluster hinweg hoch.

Selbst markante Objektkategorien entwickeln keine eigenständige Identität, sondern verteilen sich ohne klare semantische Trennung. Dies belegt, dass die Varianzminimierung bei Ward-Linkage auch im CAE-Merkmalsraum nicht ausreicht, um die komplexe Struktur von Farbbildern aufzulösen. Die Merkmale des Autoencoders, die primär auf die Rekonstruktion von Pixeln und räumlichen Hierarchien optimiert sind, bieten für dieses Verfahren nicht genügend diskriminative Kraft, um die Klassen erfolgreich zu isolieren.

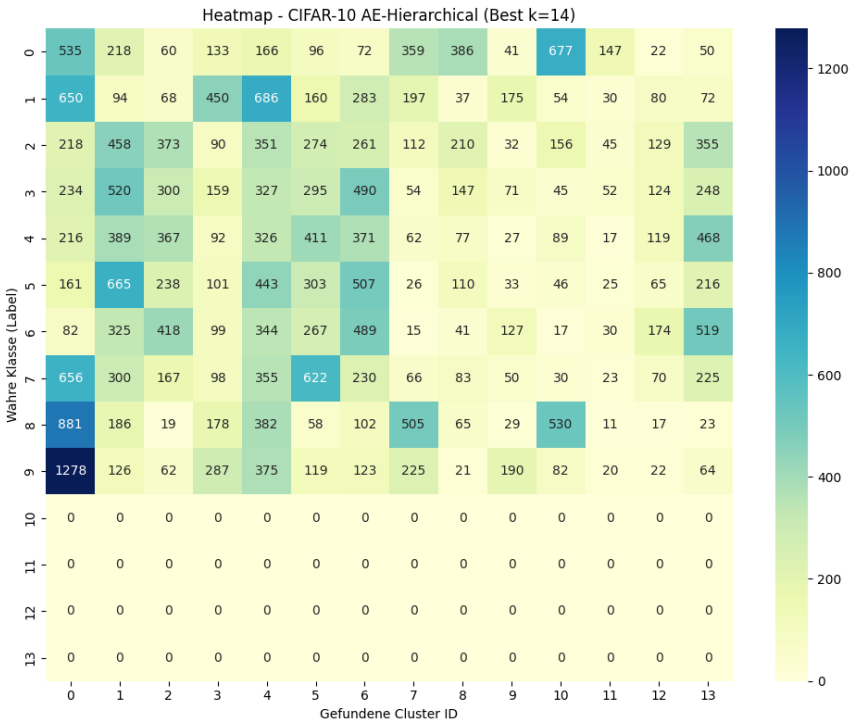

Abbildung 42 Heatmap - CIFAR-10 AE-Hierarchical (Best k=14)

4.3 Clusteranalyse im Merkmalsraum des DCGAN-Diskriminators

Nachdem die vorangegangenen Untersuchungen die Grenzen rekonstruktionsbasierter Ansätze aufgezeigt haben, widmet sich dieses Kapitel der Evaluation des DCGAN-Diskriminators als Feature-Extraktor, um die Wirksamkeit diskriminativ trainierter Repräsentationen für die Clusteranalyse zu überprüfen.

4.3.1 KMEANS mit DCGAN auf MNIST

Die Auswertung belegt, dass die diskriminativ gelernten Merkmale des DCGAN eine deutlich präzisere Trennung der Ziffernklassen ermöglichen als alle vorherigen Ansätze. Die Ergebnisse markieren den qualitativen Durchbruch deiner Untersuchung. Mit einem ARI von 0,6697 bei der Referenzgröße k=10 wird die Performance des Autoencoders (ARI 0,3179 mehr als verdoppelt. Dies beweist empirisch, dass der DCGAN-Diskriminator Merkmale extrahiert, die semantische Klassenunterschiede wesentlich schärfer abbilden als rein rekonstruktionsbasierte Vektoren.

K-Wert	ARI	NMI	Silhouette Score
8	0,6057	0,6939	0,0456
10 (Referenz)	**0,6697**	**0,7385**	**0,0481**
12	0,6267	0,7085	0,0390
15	0,5549	0,6923	0,0357

20	0,4763	0,6733	0,0487

Tabelle 17 K-Means Metrik Vergleich - MNIST

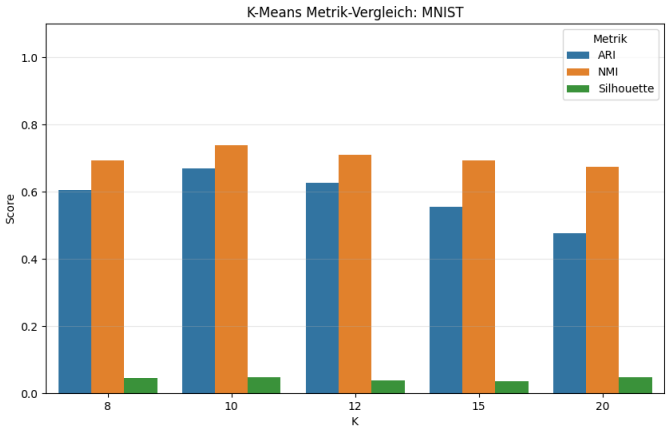

Abbildung 43 K-Means Metrik Vergleich - MNIST

Die Heatmap bei k=10 verdeutlicht diese gesteigerte Trennschärfe: Während beim Autoencoder noch viele Klassen diffus vermischt wurden, zeigen sich hier hochkonzentrierte Cluster für fast alle Ziffern. Besonders präzise werden die „1“ (1.103 Instanzen), „6“ (963 Instanzen) und „0“ (957 Instanzen) isoliert. Selbst die „Problem-Ziffern“ der Baseline (4, 7, 9) weisen nun eine deutlich stabilere Zuordnung auf, wenngleich zwischen der „4“ und „9“ systembedingt noch leichte Überlappungen verbleiben. Trotz der weiterhin niedrigen Silhouette-Scores belegt die hohe Übereinstimmung mit den wahren Labels, dass der Diskriminator die für eine erfolgreiche Gruppierung essenziellen strukturellen Merkmale erfolgreich isoliert hat.

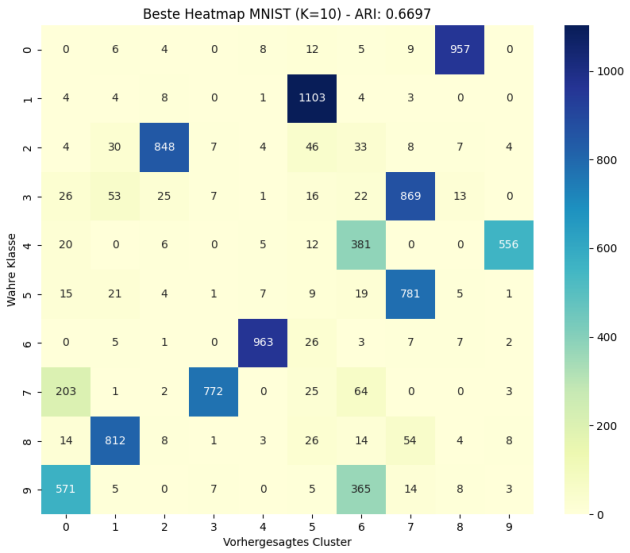

Abbildung 44 Heatmap - MNIST (k=10)

4.3.2 KMEANS mit DCGAN auf CIFAR-10

Auch hier belegt die Auswertung, dass die diskriminative Merkmalsextraktion des DCGAN-Diskriminators die semantische Struktur von CIFAR-10 signifikant besser erfasst als rein rekonstruktionsbasierte Ansätze.

Die Ergebnisse markieren einen signifikanten Fortschritt in der Merkmalsqualität für CIFAR-10. Mit einem ARI von 0,1400 bei k=12 verdreifacht das DCGAN die Performance des Autoencoders (ARI von ungefähr 0,047) nahezu. Dieser Sprung belegt, dass der Diskriminator durch das Adversarial-Training lernt, klassenspezifische Objekteigenschaften (wie Texturen und Formen) stärker zu gewichten als bloße Farbhintergründe, an denen die Baseline scheiterte.

K-Wert	ARI	NMI	Silhouette Score
8	0,1362	0,2270	-0,0248
10	0,1299	0,2218	-0,0321
12 (Optimum)	**0,1400**	**0,2356**	-0,0351
15	0,1226	0,2242	-0,0592
20	0,1225	0,2289	-0,0542

Tabelle 18 K-Means Metrik Vergleich - CIFAR-10

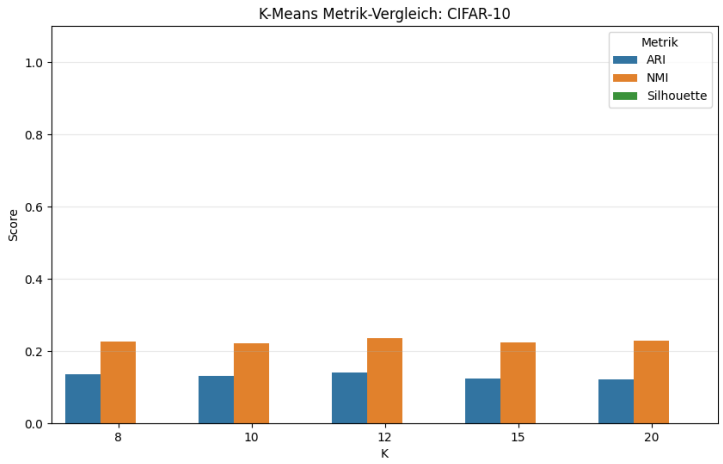

Abbildung 45 K-Means Metrik Vergleich - CIFAR-10

Die Heatmap bei k=12 belegt eine deutlich gesteigerte Trennschärfe: Klassen wie „LKW“ (474 in Cluster 3) und „Schiff“ (541 in Cluster 9) werden signifikant präziser isoliert als in den vorangegangenen AE-Experimenten. Besonders markant ist Cluster 8, der die Klassen „Frosch“ (716) und „Hirsch“ (713) massiv bündelt und damit die Bildung einer biologisch-formbasierten „Familie“ im Merkmalsraum suggeriert. Trotz der weiterhin leicht negativen Silhouette-Scores (ca. -0,03) – resultierend aus der enormen Varianz in Posen und Beleuchtung – bilden die diskriminativen Features klare semantische Schwerpunkte. Dies unterstreicht die Überlegenheit des DCGAN-Diskriminators, der eine Objektdifferenzierung ermöglicht, die mit einfachen Pixel- oder Rekonstruktionsdaten unerreicht bleibt.

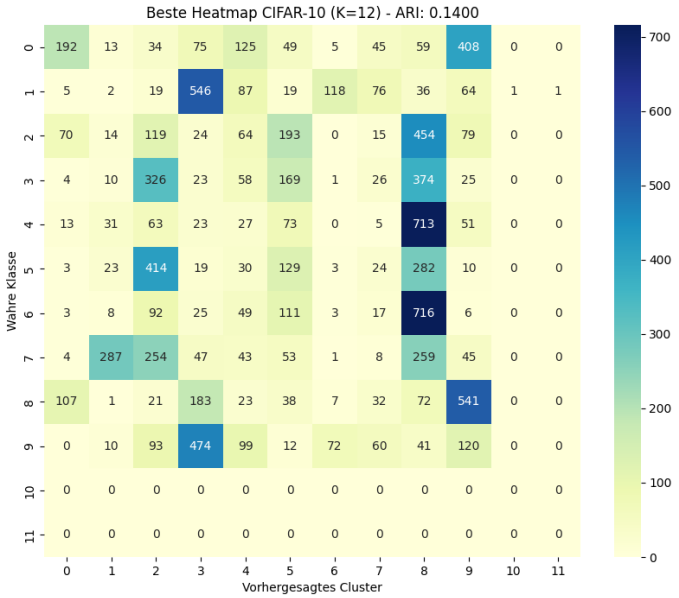

Abbildung 46 Heatmap - CIFAR-10 (k=12)

4.3.3 DBSCAN mit DCGAN auf MNIST

Die Analyse des dichte-basierten Clusterings auf den DCGAN-Repräsentationen von MNIST zeigt eine messbare Verbesserung gegenüber der Autoencoder-Baseline sowie dem AE-DBSCAN. Bei einem optimalen Parameter von ε= 30,0$ wird ein ARI von 0,1011 und eine NMI von 0,2511 erzielt. Obwohl dies eine Steigerung im Vergleich zum AE-Verfahren ARI = 0,076 darstellt, verdeutlicht der weiterhin negative Silhouette-Score von -0,0447, dass auch die diskriminativen Merkmale des DCGAN im dichte-basierten Raum noch signifikante Überlappungen aufweisen

Epsilon (eps)	Clusteranzahl	ARI	NMI	Silhouette Score
26,0	2	0,0627	0,2303	-0,0632
28,0	5	0,0874	0,2491	-0,0646
30,0 (Optimum)	**5**	**0,1011**	**0,2511**	-0,0447
32,0	2	0,0711	0,1687	0,0347
34,0	1	0,0408	0,1047	0,0000

Tabelle 19 DBSCAN Metrik Vergleich - MNIST

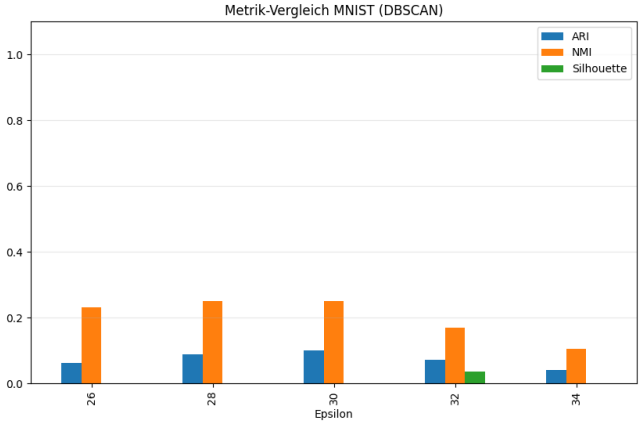

Abbildung 47 DBSCAN Metrik Vergleich - MNIST

Die qualitative Analyse bei ε= 30,0 verdeutlicht, dass das grundlegende strukturelle Problem dichte-basierter Verfahren auch bei DCGAN-Repräsentationen bestehen bleibt. Die Heatmap zeigt eine weiterhin dominante Konzentration der Daten in zwei massiven Clustern.

Zusammenfassend belegt das Experiment, dass auch das Adversarial-Training keine ausreichend isolierten „Dichte-Inseln“ für DBSCAN generiert. Während zentroid-basierte Ansätze massiv profitieren, bleibt DBSCAN aufgrund der fließenden Übergänge zwischen den semantischen Gruppen ineffektiv.

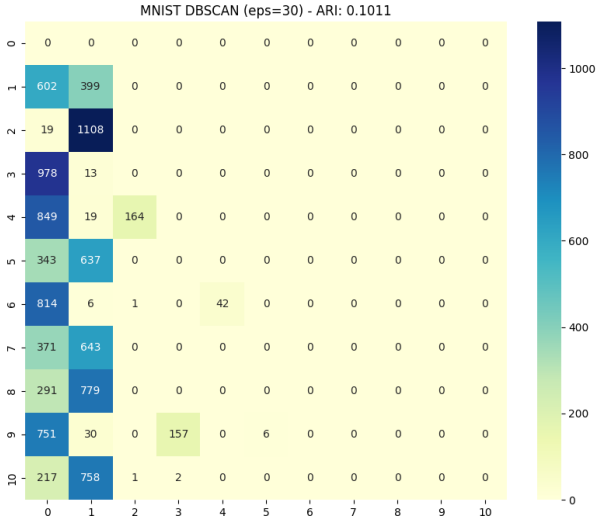

Abbildung 48 Heatmap DBSCAN - MNIST (eps = 30)

4.3.4 DBSCAN mit DCGAN auf CIFAR-10

Die DCGAN-Features mittels DBSCAN liefert bei einem optimalen Parameter von ε = 36,0 lediglich einen ARI von 0,0345 und einen NMI von 0,0641. Diese Werte belegen, dass die diskriminativen Merkmale zwar bei K-Means zu deutlichen Verbesserungen führten, für

DBSCAN jedoch weiterhin keine separierbaren Dichtestrukturen bieten. Über den gesamten untersuchten Parameterbereich von ε=32,0 bis ε=40,0 gelingt es dem Algorithmus nicht, mehr als ein einziges Cluster zu identifizieren.

Epsilon (eps)	Clusteranzahl	ARI	NMI	Silhouette Score
32,0	1	0,0233	0,0637	0,0000
34,0	1	0,0313	0,0671	0,0000
36,0 (Optimum)	**1**	**0,0345**	**0,0641**	0,0000
38,0	1	0,0321	0,0584	0,0000
40,0	1	0,0275	0,0542	0,0000

Tabelle 20 DBSCAN Metrik Vergleich - CIFAR-10

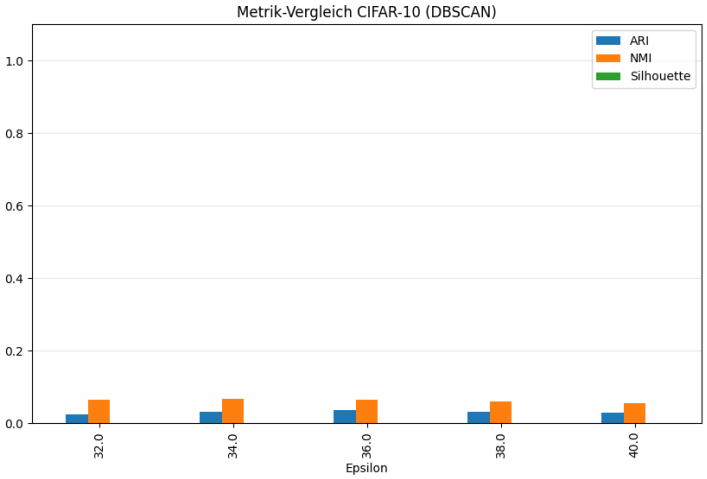

Abbildung 49 DBSCAN Metrik Vergleich - CIFAR-10

Die Heatmap bei ε= 36,0 bestätigt den erwarteten strukturellen Kollaps: Die Daten werden fast vollständig einem einzigen Cluster (ID 0) zugeordnet, während ein signifikanter Teil als Rauschen deklariert wird. Wie bereits bei der AE-Baseline entwickeln semantische Klassen keine eigenständige Identität. Dies belegt, dass der „DCGAN-Boost“ bei dichte-basierten Verfahren vollständig ausbleibt, da die enorme Bildvarianz im 128-dimensionalen Raum weiterhin zu homogenen Punktdichten führt, die eine Abgrenzung durch DBSCAN verhindern.

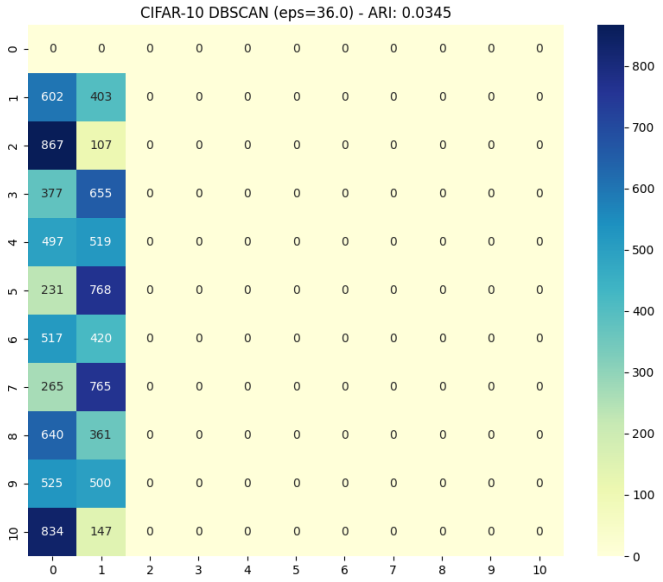

Abbildung 50 Heatmap DBSCAN - CIFAR-10 (eps = 36)

4.3.5 Hierarchisches Clustering mit DCGAN auf MNIST

Diese Ergebnisse zeigen die bisher beste Performance- der Untersuchung: Mit einem ARI von 0,7010 bei k=12 übertreffen die DCGAN-Features sowohl die PCA-Baseline als auch den besten Autoencoder-Wert (ARI ungefähr 0,51) massiv. Der beeindruckende NMI-Wert von 0,7921 belegt, dass fast 80% der in den Labels enthaltenen Informationen durch die Clusterstruktur abgedeckt werden. Dass das Optimum bei k=12 liegt, deutet darauf hin, dass der Diskriminator innerhalb der Klassen feinere, strukturell relevante Untergruppen wie verschiedene Schreibstile erfolgreich isoliert hat.

K-Wert	ARI	NMI	Silhouette Score
8	0,6530	0,7815	0,0346
10	0,6915	0,7919	0,0172
12 (Optimum)	**0,7010**	**0,7921**	0,0246
15	0,6558	0,7756	0,0297
20	0,5621	0,7502	0,0336

Tabelle 21 Hierarchisches Clustering Metrik - MNIST

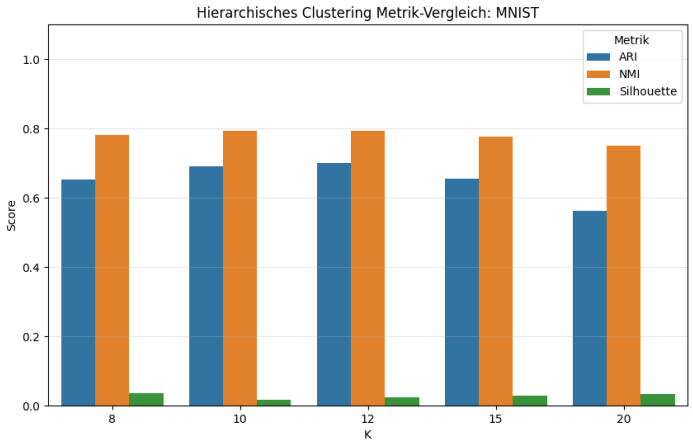

Abbildung 51 Hierarchisches Clustering Metrik - MNIST

Die Konfusionsmatrix bestätigt diese exzellente Trennschärfe: Während Ziffern wie die „0“ (982 in Cluster 4), „6“ (979 in Cluster 2) und „2“ (962 in Cluster 0) fast verlustfrei isoliert werden, zeigt die Ziffer „1“ eine semantisch sinnvolle Aufspaltung in zwei große Stil-Gruppen (Cluster 8 und 9). Selbst topologisch komplexe Problemklassen wie die „4“ werden präzise differenziert und auf die Cluster 5 und 11 verteilt, anstatt diffus zu vermischen. Zusammenfassend stellt das hierarchische Clustering auf DCGAN-Features die die beste Lösung dar, da die Kombination aus diskriminativ gelerntem Merkmalsraum und Varianzminimierung nach Ward eine semantische Gruppierung ermöglicht, welche die Kapazitäten klassischer Rekonstruktionsmodelle weit übertrifft.

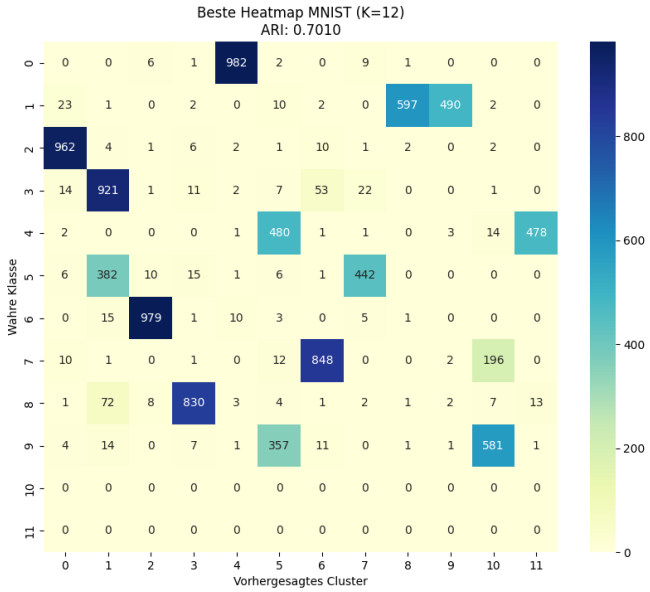

Abbildung 52 Heatmap - MNIST

4.3.6 Hierarchisches Clustering mit DCGAN auf CIFAR-10

Wie erwartet erzielen wir auch hier mit dem DCGAN-Diskriminator eine deutliche Steigerung gegenüber dem Autoencoder, wenngleich die Komplexität der Bilddaten eine perfekte Trennung weiterhin erschwert.

Die hierarchische Clusteranalyse auf den DCGAN-Latents bestätigt den „diskriminativen Vorsprung“ gegenüber dem Autoencoder: Mit einem ARI von 0,1145 bei k=10 wird die Performance der AE-Baseline ARI = 0,0412 fast verdreifacht. Die varianzminimierende Eigenschaft des Ward-Verfahrens nutzt die vom Diskriminator gelernten Objektstrukturen wesentlich effizienter als rein pixelbasierte Ansätze.

K-Wert	ARI	NMI	Silhouette Score
8	0,1079	0,1843	-0,0363
10 (Referenz)	**0,1145**	**0,1935**	-0,0388
12	0,1110	0,1976	-0,0344
15	0,1067	0,1985	-0,0784
20	0,1020	0,2021	-0,0731

Tabelle 22 Hierarchisches Clustering Metrik Vergleich - CIFAR-10

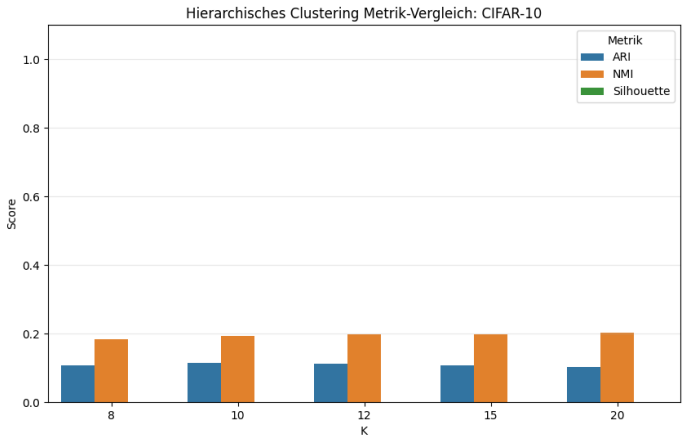

Abbildung 53 Hierarchisches Clustering Metrik Vergleich - CIFAR-10

Die Heatmap bei *k = 10* zeigt, dass das Clustering klare semantische Strukturen erlernt. Besonders auffällig ist ein dominanter Tier-Supercluster, in dem mehrere biologische Klassen wie Frösche, Hirsche und Vögel zusammengefasst werden. Dies deutet darauf hin, dass semantische Ähnlichkeiten erkannt werden, eine feingranulare Trennung innerhalb dieser Gruppe jedoch noch begrenzt ist. Technische Objekte wie Schiffe und Automobile hingegen werden deutlich präziser isoliert, was auf eine höhere Trennschärfe in diesem Bereich hinweist. Die Klasse „LKW“ verteilt sich auf mehrere Cluster, was auf unterschiedliche visuelle oder funktionale Perspektiven von Fahrzeugen schließen lässt. Insgesamt bestätigt die Analyse, dass hierarchisches Clustering auf DCGAN-Features für CIFAR-10 sinnvoll ist, da trotz hoher Bildvarianz logisch kohärente Superklassen wie Tiere und Maschinen entstehen.

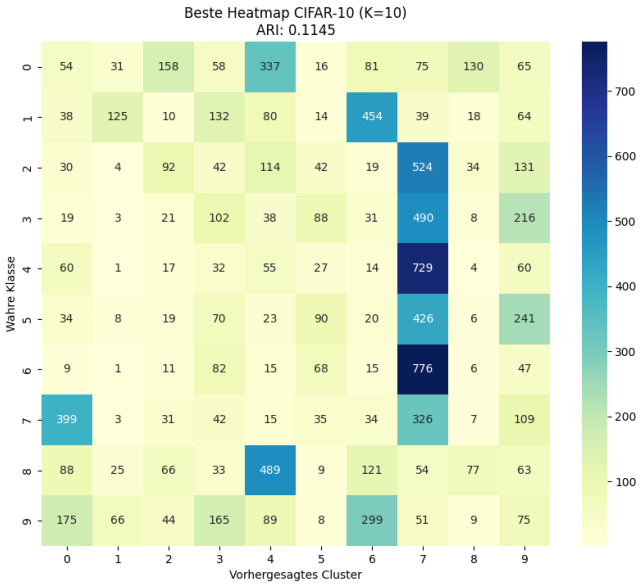

Abbildung 54 Heatmap - CIFAR-10 (k=10)

5 Zusammenfassung & Fazit

Im Rahmen dieser Projektarbeit wurde das Potenzial unüberwachter Lernverfahren zur automatisierten Strukturierung ungelabelter Bilddaten untersucht. Ziel war es, die Wirksamkeit von K-Means, hierarchischem Clustering und DBSCAN auf den Benchmark-Datensätzen MNIST und CIFAR-10 zu evaluieren.

Die Untersuchung der PCA-reduzierten Rohdaten markierte den Ausgangspunkt und zeigte unmittelbar die Grenzen klassischer Distanzmaße auf. Während MNIST aufgrund seiner hohen Kontraste noch grundlegend strukturiert werden konnte, scheiterten die Algorithmen an der Komplexität von CIFAR-10 nahezu vollständig. Besonders das dichtebasierte Clustering (DBSCAN) erwies sich im hochdimensionalen Raum als ungeeignet, da es entweder den Großteil der Daten als Rauschen deklarierte oder zu massiven Super-Clustern kollabierte.

Ein signifikanter Fortschritt wurde durch den Einsatz von Autoencodern (AE) zur Merkmalsextraktion erzielt. Die Kompression in einen kompakten Latent-Space ermöglichte es insbesondere dem hierarchischen Clustering nach Ward, stabilere semantische Gruppen zu bilden und bei MNIST einen ARI von ca. 0,51 zu erreichen. Es zeigte sich jedoch, dass die auf Rekonstruktion optimierten AE-Features eine gewisse Unschärfe aufweisen, die eine scharfe Klassentrennung bei komplexen Objekten verhindert.

Der qualitative Durchbruch der Arbeit gelang durch die Evaluation der DCGAN-basierten Repräsentationen. Durch das Adversarial-Training lernt der Diskriminator diskriminative Merkmale, die semantische Unterschiede wesentlich präziser abbilden als rein rekonstruktive Ansätze. Dies gipfelte in der besten Lösung dieser Arbeit: dem hierarchischen Clustering auf DCGAN-Features für MNIST, welches mit einem ARI von 0,7010 und einem NMI von 0,7921 den absoluten Performance-Höhepunkt markierte. Auch bei CIFAR-10 konnte die Trenngüte durch den DCGAN-Diskriminator auf einen ARI von 0,1400 verdreifacht werden.

Zusammenfassend belegt die Arbeit, dass die Wahl der Merkmalsextraktion entscheidender ist als die Wahl des Clustering-Algorithmus selbst. Während DBSCAN über alle Repräsentationsebenen hinweg aufgrund der homogenen Punktdichte ineffektiv blieb, konnten K-Means und Ward massiv von den diskriminativen DCGAN-Features profitieren. Damit Algorithmen den Inhalt komplexer Bilder verstehen können, ist eine leistungsfähige Aufbereitung mittels moderner generativer Modelle zwingend erforderlich.

Literaturverzeichnis

[1] SENSEYE/SIEMENS, „The True Cost of Downtime 2024," 2024. [Online]. Available: https://assets.new.siemens.com/siemens/assets/api/uuid:1b43afb5-2d07-47f7-9eb7-893fe7d0bc59/TCOD-2024_original.pdf. [Zugriff am 10 1 2026].

[2] Y. B. A. C. Ian Goodfellow, Deep Learning, The MIT Press, 2016, p. 338.

[3] „Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Kernel_%28image_processing%29 . [Zugriff am 12 1 2026].

[4] A. S. B. S. Everitt, The Cambridge Dictionary of Statistics, Cambridge University Pr., 2010.

[5] R. R. S. G. E. Hinton, „Reducing the Dimensionality of Data with Neural Networks," Toronto, Canada, 2006.

[6] P. D.-I. D. Schwung, „PCA/Skalierung/Training & Testing (KI)," Hochschule Düsseldorf, Düsseldorf, 2026.

[7] P. D.-I. D. Schwung, „Autoencoder & GAN," Hochschule Düsseldorf, Düsseldorf, 2026.

[8] P. D.-I. D. Schwung, „Clusteranalysen," Hochschule Düsseldorf, Düsseldorf, 2026.

[9] I. Goodfellow, „Generative Adversial Networks," Cornell University, 2014.

[10] D. Stutz, „Understanding Convolutional Neural Networks," 2014.

[11] L. M. S. C. A. Radford, „Unsupervised Representation Learning with Deep Convolutional Generative Adversial Networks," Cornell University, 2015.

[12] „skicit-learn," [Online]. Available: https://scikit-learn.org/stable/index.html. [Zugriff am 12 1 2026].

[13] **LECUN, Yann, Corinna CORTES und Christopher J.C. BURGES.** *THE MNIST DATABASE of handwritten digits* [online]. [Zugriff am: 1. Januar 2026]. Verfügbar unter: http://yann.lecun.com/exdb/mnist/

[14]Alpaydin: Maschinelles Lernen, De Gruyter 2. Ertel: Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung, Springer Vieweg 3. Lunze: Künstliche Intelligenz für Ingenieure, Oldenbourg 4. Sutton, Barto: Reinforcement Learning: An Introduction, MIT Press