UNIVERSITÄT
LEIPZIG

Institut für Informatik
Fakultät für Mathematik und Informatik
Abteilung Datenbanken

# Evaluierung von Deep-Learning-Methoden zur Generierung von Embeddings aus Genexpressionsdaten

Masterarbeit

vorgelegt von:
Leonie Preker

Matrikelnummer:
3754867

Betreuer:
Prof. Dr. Erhard Rahm
Dr. Kristin Reiche

## Zusammenfassung

Das explosionsartige Wachstum biologischer Daten durch Hochdurchsatztechnologien wie Desoxyribonukleinsäure (DNA)-Microarrays und Next Generation Sequencing hat den Bereich der Präzisionsmedizin revolutioniert. Die hohe Dimensionalität von Genexpressionsdaten bei gleichzeitig geringen Stichprobengrößen stellt jedoch aufgrund des „Fluches der Dimensionalität“ eine große Herausforderung für viele Ansätze des maschinellen Lernens dar. Zur Lösung dieses Problems muss die Anzahl molekularer Merkmale für Aufgaben des maschinellen Lernens reduziert werden. Zur Reduzierung des Attributraums einer Merkmalsmenge können sogenannte Merkmalsextraktions-Verfahren (Dimensionsreduktionsmethoden) verwendet werden. Dimensionsreduktionsmethoden führen eine lineare oder nichtlineare Transformation der ursprünglichen Merkmale durch, um die Originaldaten in einen reduzierten Merkmalsraum zu überführen. Aufgrund der signifikanten Erfolge im Bereich Deep Learning (DL) wurden verschiedene, auf DL basierende Dimensionsreduktionsmethoden zur Reduzierung des Merkmalsraums eingesetzt. Die Anzahl der Studien, welche die Qualität verschiedener DL-Verfahren bei der Reduktion von Genexpressionsdaten vergleichen, ist jedoch gering.

In dieser Arbeit wurden zwei traditionelle Dimensionsreduktionsmethoden (Selbstorganisierende Karte (SOM), Hauptkomponentenanalyse (PCA)) mit 4 verschiedenen DL-Dimensionsreduktionsmethoden (Deep Autoencoder (DAE), Stacked Denoising Autoencoder (SDAE), Variational Autoencoder (VAE), siamesisches neuronales Netz (SNN)) verglichen. Die Merkmalsextraktionsqualität der Dimensionsreduktionsmethoden wurde anhand der Vorhersage-Genauigkeit von vier Klassifikatoren (k-Nearest Neighbour (kNN), Support Vector Machine (SVM), Logistische Regression (LR), Random Forest (RF)) auf den generierten niederdimensionalen Repräsentationen evaluiert. Zur Validierung der Qualität der Dimensionsreduktionsmodelle wurden zwei öffentlich zugängliche Genexpressionsdatensätze mit unterschiedlichen Stichprobengrößen und verschiedenen prädiktiven Aufgaben (binäre Klassifizierung und Mehrklassenklassifizierung) verwendet. Neben der quantitativen Evaluierung anhand der Klassifikator-Genauigkeit wurden die Dimensionsreduktionsmethoden anhand ihrer Fähigkeit zur räumlichen Trennung von Datenklassen evaluiert. Des Weiteren wurden die verwendeten Klassifikatoren (kNN, SVM, LR, RF) anhand ihrer Vorhersagegenauigkeit bewertet.

Aus der Beobachtung der Genauigkeit geht hervor, dass die LR über die dimensionsreduzierten Datensätze die besten Ergebnisse liefert, während die SVM am schlechtesten abschnitt. Des Weiteren kann gezeigt werden, dass das SNN als einziges Modell die Vorhersagegenauigkeit über die Datensätze hinweg verbessert. Die Ergebnisse legen eine Verbesserung durch die Integration von Klasseninformationen bei der Dimensionsreduktion nahe. Insgesamt lässt sich feststellen, dass Dimensionsreduktionsverfahren die nachfolgende Klassifikation von Genexpressionsdaten nicht zwangsläufig verbessern. Ebenso lässt sich keine Verbesserung der DL-Dimensionsreduktionsmethoden gegenüber den traditionellen Dimensionsreduktionsmethoden erkennen.

Alle Skripte zur Reproduktion und zum Aufbau des in dieser Arbeit vorgestellten Evaluierungsworkflows werden unter `https://git.informatik.uni-leipzig.de/lp81vumy/master_thesis/-/tree/master` zur Verfügung gestellt.



Leonie Preker
3754867

## Danksagung

An dieser Stelle möchte ich mich bei allen, die mich beim Verfassen dieser Arbeit unterstützt haben, bedanken.
Als Erstes möchte ich mich bei Herrn Prof. Dr. Erhard Rahm bedanken für die Möglichkeit, meine Masterarbeit in der Abteilung Datenbanken am Institut für Informatik zu schreiben.
Des Weiteren möchte ich mich ganz herzlich bei Frau Dr. Kristin Reiche vom Fraunhofer IZI Leipzig für ihre stets freundliche, kompetente und engagierte Betreuung meiner Masterarbeit bedanken. In diesem Zuge möchte ich mich auch bei dem Fraunhofer IZI für die Kooperation sowie die Bereitstellung von Arbeitsmitteln bedanken.
Darüber hinaus gilt mein großer Dank Herrn Dr. Victor Christen, der mir diese Masterarbeit in Kooperation mit dem Fraunhofer IZI vermittelt und mich bei meiner Masterarbeit intensiv betreut und unterstützt hat sowie mir mit seinem informatischen Fachwissen jederzeit zur Seite gestanden ist.
Ebenfalls möchte ich mich bei dem Rechenzentrum der Universität Leipzig für die Bereitstellung von Ressourcen des Clara Clusters sowie den guten Support bedanken.
Ein besonderer Dank gilt meiner Familie, vor allem meinen Eltern, die mir das Studium ermöglicht und mich jederzeit unterstützt haben.

Leonie Preker
3754867

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| **ADAM** | Adaptive Moment Estimation |
| **AD-AE** | Adversarial Deconfounding Autoencoder |
| **AE** | Autoencoder |
| **ANN** | künstliches neuronales Netz |
| **BMU** | Best Matching Unit |
| **cDNA** | komplementäre DNA |
| **CRC** | Darmkrebs |
| **DAE** | Deep Autoencoder |
| **DL** | Deep Learning |
| **DNA** | Desoxyribonukleinsäure |
| **ELBO** | Evidence Lower Bound |
| **FN** | falsch negativ |
| **FP** | falsch positiv |
| **GEO** | Gene Expression Omnibus |
| **GDC** | Genomic Data Commons |
| **HDF** | hierarchisches Datenformat |
| **kNN** | k-Nearest Neighbour |
| **LR** | Logistische Regression |
| **miRNA** | micro-RNA |
| **ML** | Machine Learning |
| **mRNA** | messenger-RNA |
| **MSE** | mittlerer quadratischer Fehler |
| **PCA** | Hauptkomponentenanalyse |
| **PCR** | Polymerase-Kettenreaktion |
| **ReLU** | Rectified Linear Unit |
| **RF** | Random Forest |
| **RNA** | Ribonukleinsäure |
| **RNA-Seq** | RNA-Sequenzierung |
| **SDAE** | Stacked Denoising Autoencoder |
| **SGD** | Stochastic Gradient Descent |

| | |
|---|---|
| **SNE** | Stochastic Neighbor Embedding |
| **SNN** | siamesisches neuronales Netz |
| **SOM** | Selbstorganisierende Karte |
| **SRA** | Sequence Read Archive |
| **SVM** | Support Vector Machine |
| **TCGA** | The Cancer Genome Atlas |
| **TN** | wahr negativ |
| **TP** | wahr positiv |
| **tRNA** | transfer-RNA |
| **t-SNE** | t-distributed Stochastic Neighbor Embedding |
| **VAE** | Variational Autoencoder |

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1 Einleitung

## 1.1 Motivation

Ein Großteil der Medikamente wurden mit dem Ziel entwickelt, eine hohe Wirksamkeit für einen Großteil der Patienten zu erreichen. Dies hat zur Folge, dass gewisse Medikamente bei einigen Patienten erfolgreich sind, bei anderen jedoch nicht [1]. Ursache für diese interindividuelle Variation der Arzneimittelantwort ist der genetische Hintergrund sowie die Umweltexposition jedes einzelnen Patienten [2]. Daher werden neue Präventions- und Behandlungsstrategien, welche die individuelle Variabilität berücksichtigen, benötigt [3]. Bei der Präzisionsmedizin werden Patienten hinsichtlich Ihrer Anfälligkeit für eine bestimmte Krankheit oder in ihrer Reaktion auf eine bestimmte Behandlung in Untergruppen eingeteilt, um die medizinischen Behandlungen entsprechend anzupassen [4]. Die Genexpressionsanalyse ist eine häufig verwendete Methode, um Mechanismen die zur Ausprägung einer bestimmten Krankheit führen, zu identifizieren [5]. Genexpressionsdaten enthalten intrinsische biomedizinische Informationen, welche die individuelle genetische Variabilität repräsentieren und somit entscheidend für die Präzisionsmedizin sind [6]. Die angewandte Hypothese ist, dass ähnliche Genexpressionsprofile eine verwandte Funktion implizieren. Die Übertragung der aus Genexpressionsdatensätzen gewonnenen Informationen in personalisierte Therapien war bislang jedoch beschränkt [7, 8].

Der Durchbruch im Bereich der künstlichen Intelligenz hat die Erforschung der biomedizinischen Forschung vorangetrieben. Insbesondere Deep Learning (DL), einer der aufstrebenden Bereiche der künstlichen Intelligenz, hat in Bereichen wie Computer Vision [9] und Natural Language Processing [10] signifikante Erfolge erzielt. Diese Erfolge beruhen auf der Fähigkeit von DL-Modellen, Merkmale eigenständig auf ein höheres Abstraktionsniveau zu transformieren [11]. Basierend auf diesen Erfolgen wurden verschiedene DL-Methoden vorgeschlagen, um Phänotypen aus Genexpressionsprofilen zu identifizieren [12, 13]. Trotz ihrer Leistungsfähigkeit bestehen bei der Verwendung von DL-Methoden in der biomedizinischen Forschung noch große Herausforderungen. Dazu gehören der Bedarf an großen Datensätzen, fehlendes Wissen über die biologischen Systeme und der Mangel an Interpretierbarkeit von DL-Modellen [14]. Das Fehlen von großen Datensätzen führt zu einem der Hauptprobleme bei der Analyse von Genexpressionsdaten mit Machine Learning (ML)-Verfahren, dem „large p (Anzahl der Merkmale) small n (Anzahl der Proben) - Problem“. Dieses Problem ist auf zwei inhärente Eigenschaften von Genexpressionsdaten zurückzuführen. Zum einen ist die Anzahl der Merkmale in Genexpressionsdaten sehr hoch. Zum anderen ist die Anzahl der öffentlich verfügbaren Proben, die man unter einer bestimmten Bedingung erhalten kann, beschränkt. Dies hat technologische (z. B. Geräteauflösung und damit verbundene Kosten) und grundlegende biologische (z. B. Grenzen der Kohortengröße unter einer bestimmten Bedingung) Ursachen [14]. Die Kombination der zwei inhärenten Eigenschaften führt dazu, dass Genexpressionsdatensätze eine hohe Dimensionalität (Anzahl der Merkmale) bei gleichzeitig geringer Stichprobengröße aufweisen.

Die dadurch entstehenden spärlichen Daten stellen eine große Herausforderung für viele Ansätze des maschinellen Lernens dar. Die resultierenden Probleme des maschinellen Lernens hängen mit dem von Bellman entdeckten Phänomen, dem „Fluch der Dimensionalität“ zusammen [15]. Der „Fluch der Dimensionalität“ bezeichnet das mit zunehmender Dimensionalität der Daten exponentielle Wachstum der Datenmenge, welches für eine zuverlässige Analyse erforderlich ist [16]. Daher sind im Vergleich zu ML-Modellen, die auf einem niedrigdimensionalen Datensatz mit großer Stichprobenanzahl trainiert werden, Modelle, die auf einem hochdimensionalen Genexpressionsdatensatz mit geringer Stichprobenanzahl basieren, oft überangepasst und weisen eine hohe Varianz auf [17].

Eine Lösung des „large p small n - Problems“ stellt die Reduzierung der molekularen Merkmale für nachgelagerte Aufgaben des maschinellen Lernens dar [6]. Zwei gängige Ansätze sind die Merkmalsauswahl und die Merkmalsextraktion. Verfahren zur Merkmalsauswahl ermitteln aussagekräftige Merkmale aus dem Datensatz durch Entfernung von irrelevanten und redundanten Merkmale [18], wohingegen die Algorithmen zur Merkmalsextraktion darauf abzielen, die ursprünglichen Stichproben im hochdimensionalen Raum in einen niedrigdimensionalen Raum einzubetten [19]. Die Idee der Einbettung besteht darin, hochdimensionale kategoriale Variablen auf eine niedrigdimensionale gelernte Darstellung abzubilden, sodass ähnliche Entitäten im Einbettungsraum näher beieinander platziert werden [20]. Dies gewährleistet bei der Merkmalsextraktion im Vergleich zu der Merkmalsselektion einen geringeren Informationsverlust und eine höhere Trennschärfe [18]. Daher beschränken sich die in dieser Arbeit durchgeführten Analysen auf Techniken zur Merkmalsextraktion.

Es wurden bereits verschiedene DL-Methoden (Denoising Autoencoder, Stacked Denoising Autoencoder (SDAE) Variational Autoencoder (VAE), siamesisches neuronales Netz (SNN)) erfolgreich eingesetzt, um niedrigdimensionale Darstellungen aus Genexpressionsdaten zu extrahieren, die sich für Sekundäranalysen in verschiedenen Bereichen eignen [21, 22, 23, 24, 6, 25, 26, 27, 28, 29, 30]. Prädiktive Aufgaben, die auf diesen niedrigdimensionalen Darstellungen basieren, sollten Modelle übertreffen, die auf Basis der ursprünglichen Eingabedaten erstellt wurden, vorausgesetzt die extrahierten Merkmale erfassen biologisch relevante Prozesse [12].

Neben der verbesserten Qualität von Sekundäranalysen auf Datensätze mit reduzierter Dimensionalität bringt die Dimensionsreduktion weitere entscheidende Vorteile mit sich [31]:

- Reduzierung der Speicherauslastung und des Rechenaufwands.
- Entfernung redundanter, irrelevanter und verrauschter Daten, wodurch eine Verbesserung der Datenqualität erreicht werden kann.
- Unterstützung der Datenvisualisierung, sodass eine bessere Interpretation „versteckter“ Strukturen in den Daten möglich ist.

Das Ziel ist es, die in dieser Arbeit generierten niedrigdimensionalen Darstellungen in nachfolgenden wissenschaftlichen Arbeiten als Entitäten in sogenannten Wissensgraphen zu verwenden. Dies stellt bislang aufgrund der großen Dimensionalität und der damit verbundenen hohen Speicherauslastung ein Problem dar. Wissensgraphen bieten die Möglichkeit, heterogene Daten und ihre Beziehungen zueinander effektiv darzustellen. Da mehrere Ansichten auf denselben Patienten

ergänzende Informationen liefern können, hat die integrative Analyse von verschiedenen medizinischen Daten mit Ansätzen des maschinellen Lernens großes Potenzial, die molekulare Grundlage der Krankheitsursache aufzuklären [32].

Trotz der großen Menge an unterschiedlichen Dimensionsreduktionsmethoden ist bislang keine Methode bekannt, die für alle Situationen das beste Ergebnis liefert. Deshalb sollten die Dimensionsreduktionsmodelle abhängig von den zu analysierenden Daten und der gewünschten Anwendung ausgewählt werden. In dieser Arbeit wurde ein Evaluierungsframework zum Vergleich von vier häufig verwendeten, auf DL basierenden Methoden zur Dimensionsreduktion von Genexpressionsdaten entwickelt.

## 1.2 Zielstellung

Auf dieser Grundlage besteht der Hauptzweck dieser Arbeit darin, einen stabilen und vielseitig einsetzbaren Ansatz zur Dimensionsreduktion auf Grundlage von Genexpressionsdaten zu finden, um die personalisierte medizinische Entscheidungsfindung zu verbessern.

Um dies zu erreichen, wurden folgende Aufgaben angegangen:

1. Die Konzeptualisierung eines Workflows zur Evaluierung verschiedener DL-Dimensionsreduktionsmodelle: Die Konzeptualisierung umfasst die Identifikation der essentiellen Schritte des Workflows, welche (a) die Datenerfassung, (b) die Datenvorverarbeitung, (c) die Generierung der Training-, Validierungs- und Testdaten, (d) die Definition der verschiedenen Dimensionsreduktionsmethoden-Methoden sowie (e) die Evaluierung der Dimensionsreduktionsmethoden. Dabei ist zu beachten, dass der Workflow erweiterbar sein soll, sodass zukünftige Dimensionsreduktionsmethoden-Methoden effizient integriert werden können.

2. Die Implementierung des Evaluierungsworkflows: Der definierte Evaluierungsworkflows soll prototypisch mittels existierender DL-Frameworks in Python implementiert werden. Die Implementierung des Evaluierungsworkflows soll weiterhin die Möglichkeit bieten, verschiedene Konfiguration effizient, d.h. ohne hohen manuellen Konfigurationsaufwand ausführen zu können.

3. Die Evaluierung verschiedener DL-Dimensionsreduktionsmodelle basierend auf einer nachfolgenden prädiktiven Aufgabe: Die Evaluierung zeigt die Durchführbarkeit des Workflows sowie die Effektivität der einzelnen Modelle unter Berücksichtigung verschiedener Konfigurationen. Zur Validierung der Merkmalsextraktionsqualität der DL-Dimensionsreduktionsmodelle werden die einzelnen DL-Modelle untereinander, mit traditionellen Dimensionsreduktionsmethoden sowie einem Baseline-Modell, welches die Daten nicht reduziert, verglichen.

## 1.3 Aufbau der Arbeit

Die Arbeit ist wie folgt aufgebaut:

**Kapitel 2** beschreibt sowohl die Grundlagen der in dieser Arbeit verwendeten Methoden als auch verwandte Arbeiten. Zu den Grundlagen gehören unter anderem die verschiedenen Modelle, die in dieser Arbeit zur Durchführung der Dimensionsreduktion verwendet wurden.

**Kapitel 3** beschreibt das Konzept des Evaluierungsworkflows zum Vergleich verschiedener DL-basierter Methoden zur Dimensionsreduktion von Genexpressionsdaten. Des Weiteren werden die in dieser Arbeit verwendeten Datensätze sowie Einzelheiten zur Implementierung der Dimensionsreduktionsmodelle erläutert.

In **Kapitel 4** sind die anhand des Evaluierungsworkflows generierten Ergebnisse dargestellt. Zudem werden die Ergebnisse der untersuchten Dimensionsreduktionsmodelle sowie der verwendeten Datensätze mit zugehöriger prädiktiver Aufgabe detailliert verglichen und diskutiert.

**Kapitel 5** gibt einige Schlussfolgerungen und mögliche zukünftige Arbeiten zur Verbesserung und Weiterführung dieser Arbeit.

# 2 Grundlagen und verwandte Arbeiten

In diesem Kapitel werden die für das Verständnis erforderlichen elementaren Begriffe, Konzepte und Verfahren erläutert sowie verwandte Arbeiten präsentiert. Abschnitt 2.1 beginnt mit den Grundlagen der Genexpression sowie einer kurzen Einführung in die häufig verwendeten Methoden zur Messung der Genexpression. In Abschnitt 2.2 wird das Verfahren der Dimensionsreduktion im Allgemeinen beschrieben. In Abschnitt 2.3 wird eine Einführung in künstliche neuronale Netze (ANNs) gegeben. Aufbauend darauf wird in Unterabschnitt 2.3.1 und Unterabschnitt 2.3.2 auf verschiedene Arten von ANNs zur Dimensionsreduktion eingegangen. Unterabschnitt 2.3.3 fasst die wichtigsten Eigenschaften der Dimensionsreduktionsmethoden tabellarisch zusammen. Abschließend werden in Abschnitt 2.4 verwandte Arbeiten zur Dimensionsreduktion anhand verschiedener DL-Technologien auf Genexpressionsdaten vorgestellt.

## 2.1 Genexpression

Die genetische Information eines Organismus ist in dem sogenannten Genom codiert, das aus Molekülen der Desoxyribonukleinsäure (DNA) besteht. Gene sind Abschnitte auf der DNA, welche die Informationen für die Erzeugung funktioneller zellulärer Produkte tragen [33]. Die Herstellung funktioneller zellulärer Produkte wird als Genexpression bezeichnet [5]. Die Genexpression folgt einem obligatorischen Fluss, der als zentrales Dogma der Molekularbiologie bezeichnet wird [34] (Abbildung 1).

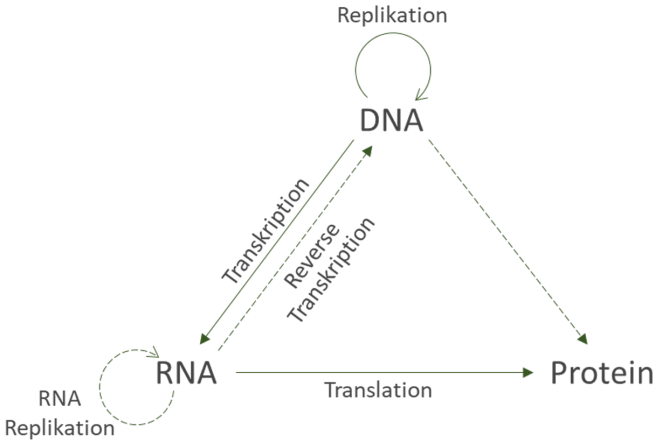



**Abbildung 1:** Zentrales Dogma der Molekularbiologie. Die durchgezogenen Pfeile stellen die übliche Übertragung der biologischen Information dar, die gepunkteten Pfeile stellen die in Ausnahmefällen durchgeführte Übertragung dar [34]. In Anlehnung an Francis Crick [34]. DNA = Desoxyribonukleinsäure; RNA = Ribonukleinsäure.

Das zentrale Dogma der Molekularbiologie beschreibt die Übertragung von Informationen ausgehend von der DNA über die RNA bis hin zu den Proteinen [34]. In der DNA/RNA werden die Informationen verewigt oder übertragen, in Proteinen werden die Informationen lediglich verewigt,

da die Übertragung von Informationen in Proteine irreversibel ist. Als Genexpression wird der Prozess bezeichnet, bei dem ein Gen in ein RNA-Molekül kopiert wird, das wiederum zur Synthese funktioneller Genprodukte verwendet werden kann [5]. Der Prozess, durch den die genetische Information der DNA in ein RNA-Molekül übertragen wird, wird als Transkription bezeichnet. Bestimmte RNA-Moleküle, sogenannte messenger-RNAs (mRNAs), werden anschließend während der Translation als Vorlage für die Bildung von Proteinen verwendet. Andere Transkripte wie transfer-RNA (tRNA) oder micro-RNA (miRNA) sind das eigentliche Endprodukt und werden demnach nicht translatiert. Obwohl alle Zellen eines Organismus denselben Satz an Genen enthalten, werden nur aktive Gene exprimiert [35]. Gene können unterschiedlich stark exprimiert sein. Der gesamte Satz der Gene, die zu einem bestimmten Zeitpunkt in einer Zelle vorliegenden, wird als Genom, der gesamte Satz der RNA-Transkripte als Transkriptom und der gesamte Satz der Proteine als Proteom bezeichnet. Die Gesamtheit dieser Wissenschaftszweige, die mit dem Suffix-Omics (in deutsch mit dem Suffix-Omik) enden und Genomik, Epigenomik, Transkriptomik, Proteomik, Metabolomik usw. umfassen, werden als „Omics“ bezeichnet [36]. In dieser Arbeit wird unter Genexpression die Expression der Gene auf Transkriptom-Ebene verstanden. Dementsprechend umfassen Genexpressionsdaten die Gesamtheit der Transkripte in einer Zelle zum Erfassungszeitpunkt. Die Genexpressionsdaten stellen eine wichtige Verbindung zwischen der DNA und dem individuellen Phänotypen wie Krankheitsstadium und Zell- oder Gewebetyp dar [37].
In den letzten Jahrzehnten wurden wichtige Fortschritte in der Entwicklung verschiedener Technologien zur Messung der Genexpression gemacht. Die zwei am häufigsten verwendeten Methoden sind die auf Hybridisierung basierende Microarray-Technologie (vereinfacht als Microarrays bezeichnet) [38] und die RNA-Sequenzierung (RNA-Seq) [39]. Beide sind in der Lage, Genexpressionsmuster parallel für Hunderte oder Tausende von Genen gleichzeitig zu messen.

### 2.1.1 Messen der Genexpression mittels Microarrays

Ein DNA-Mikroarray [38, 40] besteht aus einer Anordnung von kurzen, Polymerase-Kettenreaktion (PCR)-amplifizierten komplementären DNAs (cDNAs) oder Oligonucleotiden, den sogenannten Sonden. Jede Sonde besteht aus einem kleinen Teil einer Nukleotidsequenz, die einem einzelnen Gen zugeordnet werden kann. Die Sonden sind kovalent an feste Stellen (Spots) eines DNA-Chips gebunden. Jeder einzelne Spot auf dem Array enthält eine Vielzahl an Kopien der spezifischen Nukleotidteilsequenz, welche für ein einzelnes Gen repräsentativ ist. Die Ziel-RNA wird in der Regel aus Proben von Interesse (zum Beispiel: Tumorgewebe) extrahiert, in cDNA revers transkribiert, fluoreszenzmarkiert und zuletzt mit dem Array hybridisiert (Abbildung 2). Zwei DNA-Stränge hybridisieren, wenn sie komplementär zueinander sind. Die Quantität der in der Probe enthaltenen spezifischen RNA kann durch die optische Messung der Fluoreszenzintensität eines Spots auf dem Objektträger geschätzt werden [41].
Die Microarrays weisen trotz ihrer Leistungsfähigkeit und Preisgünstigkeit einige Einschränkungen auf [42]. Zum einen müssen die Sondensequenzen im Voraus festgelegt werden, dementsprechend sind vorherige Kenntnisse über die zu untersuchenden Sequenzen erforderlich. Zum anderen besitzen die Expressionsniveaus häufig ein hohes Hintergrundsignal, welches durch unspezifische Bindung

von cDNAs an die Sonden entsteht. Die auf Sequenzierung basierende RNA-Seq hat das Potenzial, diese Einschränkungen zu überwinden.

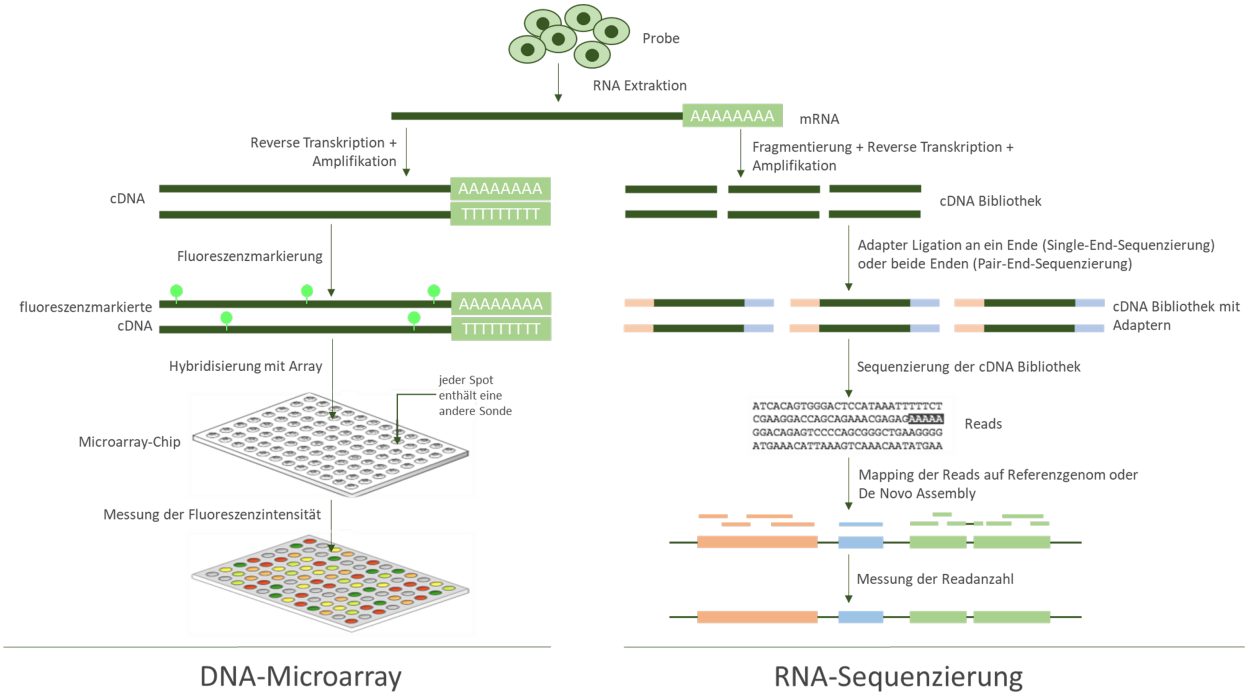



**Abbildung 2:** Workflow der Technologien zur Messung der Genexpression. **Microarray (links):** RNA wird aus den Zellen extrahiert, revers transkribiert, mit fluoreszierenden Sonden markiert und zuletzt mit dem Array hybridisiert. Die Quantität der in der Probe enthaltenen spezifischen RNA kann durch die optische Messung der Fluoreszenzintensität eines Spots gemessen werden. **RNA-Sequenzierung (rechts):** Die extrahierte RNA wird fragmentiert, revers transkribiert und mit Adaptern modifiziert, um die Sequenzierung zu erleichtern. Die cDNA wird sequenziert, und die resultierenden Reads gegen ein bekanntes Referenzgenom gemappt oder de novo assembliert, um die Expressionsniveaus verschiedener Gene in der Probe zu ermitteln. Die Expressionsniveaus der Gene werden durch Aufsummieren der am Transkriptom/Genom ausgerichteten Reads ermittelt. Die Grafik ist angelehnt an [42] und [43]. DNA = Desoxyribonukleinsäure; RNA = Ribonukleinsäure; cDNA = komplementäre DNA; mRNA = messenger-RNA.

### 2.1.2 Messen der Genexpression mittels RNA-Sequenzierung

Die Einführung von Hochdurchsatz-Sequenzierungsmethoden der „nächsten Generation“ [44] wie der RNA-Seq [39] hat die Transkriptomik revolutioniert. Anders als Microarrays bestimmt die RNA-Seq direkt die cDNA-Sequenz. Abbildung 2 zeigt eine schematische Darstellung des RNA-Seq-Workflows. Die RNA-Seq umfasst die Extraktion der RNA aus einer Zellpopulation, die Umwandlung der gesamten oder fraktionierten RNA-Population in eine Bibliothek von cDNA-Fragmenten mit an einem oder beiden Enden versehenen Adaptern und die Hochdurchsatzsequenzierung der gesamten cDNA-Bibliothek zur Generierung der Sequenzen (Reads) von einem Ende (Single-End-Sequenzierung) oder beiden Enden (Pair-End-Sequenzierung) [42]. Anschließend werden die resultierenden Reads gegen ein bekanntes Referenzgenom gemappt oder de novo assembliert. Durch

Aufsummieren der am Transkriptom/Genom ausgerichteten Reads können nun die Expressionsniveaus der Gene bestimmt werden. In einem einzigen RNA-Seq-Experiment können nicht nur die Genexpression, sondern auch alternatives Spleißen [45], die Expression neuartiger Transkripte [46], allelspezifische Expression [47], Genfusionsereignisse [48] und genetische Variationen untersucht werden.
Wie in Unterabschnitt 2.1.1 bereits erwähnt, bietet die RNA-Seq gegenüber Microarray-Technologien mehrere entscheidende Vorteile [42]. Zum einen ist RNA-Seq im Gegensatz zu Microarrays nicht auf die Untersuchung von vorab bekannten Sequenzen beschränkt. Diese Tatsache macht die RNA-Seq für die Untersuchung von Organismen mit noch nicht erfassten genomischen Sequenzen besonders wertvoll. Zum anderen besitzt die RNA-Seq aufgrund ihrer Fähigkeit, DNA-Sequenzen eindeutig auf einzigartige Regionen des Genoms abzubilden, ein sehr geringes bis gar kein Hintergrundsignal.

## 2.2 Dimensionsreduktion

Die Dimensionsreduktion bezieht sich auf eine Gruppe von Techniken, die darauf abzielen, die Dimensionalität $d$ eines hochdimensionalen Datensatzes in eine aussagekräftige Darstellung mit reduzierter Dimensionalität $q$ umzuwandeln [49]. Durch den explosionsartigen Anstieg hochdimensionaler Daten sind Verfahren zur Dimensionsreduktion in vielen Anwendungsbereichen attraktiv geworden, unter anderem im Bereich der Präzisionsmedizin [6, 50]. Die Dimensionsreduktion hochdimensionaler Daten trägt dazu bei, das Problem des „Fluchs der Dimensionalität“ [15] zu lösen. Des Weiteren lässt sich durch Verfahren zur Dimensionsreduktion eine Speicherreduktion bei ähnlichem Informationsgehalt erreichen. Die Vorteile der Dimensionsreduktion sind die vereinfachte Analyse, Verarbeitung sowie Visualisierung der Daten [18]. Wie schon in Kapitel 1 erwähnt, existieren zwei häufig verwendete Techniken zum Reduzieren des Attributraums einer Merkmalsmenge:

- Verfahren zur Merkmalsauswahl und
- Verfahren zur Merkmalsextraktion.

Verfahren zur **Merkmalsauswahl** ermitteln aussagekräftige Merkmale aus dem Datensatz durch Entfernung von irrelevanten und redundanten Merkmale [18]. **Merkmalsextraktionsverfahren** führen eine Transformation der ursprünglichen Merkmale durch, um die Originaldaten in einen reduzierten Merkmalsraum zu überführen [51]. Sowohl Merkmalsauswahlverfahren als auch Merkmalsextraktionsverfahren führen zwangsläufig zu einem gewissen Informationsverlust der Originaldaten, allerdings gewährleistet die Transformation der Daten bei der Merkmalsextraktion einen geringeren Informationsverlust und eine höhere Trennschärfe im Vergleich zu der Merkmalsselektion [18]. Daher beschränken sich die in dieser Arbeit durchgeführten Analysen auf Techniken zur Merkmalsextraktion.
Merkmalsextraktionsmodelle beruhen auf der Mannigfaltigkeitshypothese, die besagt, dass reale, hochdimensionale Daten entlang einer niedrigdimensionalen Mannigfaltigkeit liegen, die in einen hochdimensionalen Raum eingebettet ist [52]. Demnach entspricht die optimale Dimensionalität

des reduzierten Merkmalsraums der intrinsischen Dimensionalität (Dimensionalität der Mannigfaltigkeit) der Daten [49]. Die intrinsische Dimensionalität eines Datensatzes entspricht der minimalen Anzahl von Merkmalen, die erforderlich sind, um die beobachteten Eigenschaften des Datensatzes zu beschreiben [53]. Intuitiv zielt die Dimensionsreduktion also darauf ab, das Wesentliche der Daten zu erfassen [54]. Bei der Merkmalsextraktion werden die niedrigdimensionalen Äquivalente der Stichproben des ursprünglichen Datensatzes als latente Punkte oder Embedding bezeichnet und der Raum, in dem das Embedding liegt, wird als latenter Raum $\mathbb{R}^q$ bezeichnet. Entsprechend dazu wird der Raum, in dem die ursprünglichen Daten liegen, als $\mathbb{R}^d$ bezeichnet, wobei $q < d$. Die Ziele der Merkmalsextraktionsverfahren bestehen darin [31]:

- eine Überanpassung der Daten zu vermeiden und die Modellleistung für nachgeschaltete Klassifikations- oder Clusteranalysen zu verbessern,
- schnellere und kosteneffektivere Modelle zu erzeugen und
- die Datenvisualisierung zu unterstützen, sodass eine bessere Interpretation „versteckter“ Strukturen in den Daten möglich ist.

Merkmalsextraktionsalgorithmen lassen sich in lineare und nichtlineare Verfahren unterteilen [49]. Bei der linearen Merkmalsextraktion wird davon ausgegangen, dass die Daten auf einem niedrigdimensionalen linearen Unterraum liegen. Demnach haben linearen Verfahren Probleme bei der Verarbeitung komplexer nichtlinearer Daten. Im Gegensatz zu den linearen Verfahren sind die nichtlinearen Verfahren in der Lage, mit komplexen nichtlinearen Daten umzugehen. Daher werden bei der Verarbeitung realer Daten vermehrt nichtlineare Dimensionsreduktionsmethoden eingesetzt.
Die populärste Methode zur linearen Dimensionsreduktion ist die von Pearson 1901 eingeführte **Hauptkomponentenanalyse (PCA)** [55]. Das Ziel der PCA besteht darin, einen Datensatz mit einer großen Anzahl korrelierter Variablen in einen niedrigdimensionalen Datensatz mit unkorrelierten Variablen zu transformieren, die einen maximalen Teil der Varianzen in dem Datensatz erhalten [49]. Dieser Prozess erfolgt durch orthogonale Transformation des ursprünglichen Datensatzes in einen kleineren Satz von Attributen, die als Hauptkomponenten bezeichnet werden. Die resultierenden Hauptkomponenten sind unkorreliert, wobei der größte Eigenwert (die maximale Varianz) als erste Hauptkomponente und der kleinste Eigenwert als letzte Hauptkomponente eingestuft wird [56]. Die PCA ist eine nichtparametrische Dimensionsreduktionsmethode, und die einzige Annahme bei der PCA ist, dass alle Daten eine lineare Kombination der Hauptkomponenten sind.
Neben der PCA können auch ANNs wie die Selbstorganisierende Karte (SOM) [57] oder der Autoencoder (AE) zur Dimensionsreduktion verwendet werde. Im Folgenden werden zunächst die grundlegenden Konzepte von ANNs erläutert, bevor die verschiedenen auf ANNs basierenden Dimensionsreduktionsmethoden vorgestellt werden.

## 2.3 Künstliche neuronale Netze

In diesem Abschnitt wird ein Überblick über die Grundlagen von neuronalen Netzen und DL-Verfahren gegeben. ANNs sind eine Methode des maschinellen Lernens im Bereich der künstlichen Intelligenz. Das Konzept der ANNs wurde von biologischen neuronalen Netzen inspiriert. Die

ANNs lassen sich auf das von Warren McCulloch und Walter Pitts 1943 vorgeschlagene mathematische Modell der neuronalen Aktivität im Gehirn zurückführen [58]. Sie bewiesen, dass dieses Neuronenmodell in der Lage war, jede berechenbare Funktion mit einer endlichen Anzahl künstlicher Neuronen und einstellbaren synaptischen Gewichten auszuführen. ANN bestehen aus einer Ein- und Ausgabeschicht sowie einer bis mehreren zwischengelagerten verdeckten Schichten. Jede einzelne Schicht besteht aus einem bis mehreren künstlichen Neuronen, die miteinander über Gewichte verbunden sind. Das Trainieren der Gewichte ist der wichtigste Mechanismus, mit dem ein neuronales Netz in der Lage ist, Vorhersagefunktionen zu modellieren. Vor der Betrachtung des vollständigen neuronalen Netzes wird das Verhalten eines einzelnen künstlichen Neurons präsentiert. In Abbildung 3 ist eine schematische Darstellung des mathematischen Modells eines künstlichen Neurons dargestellt.

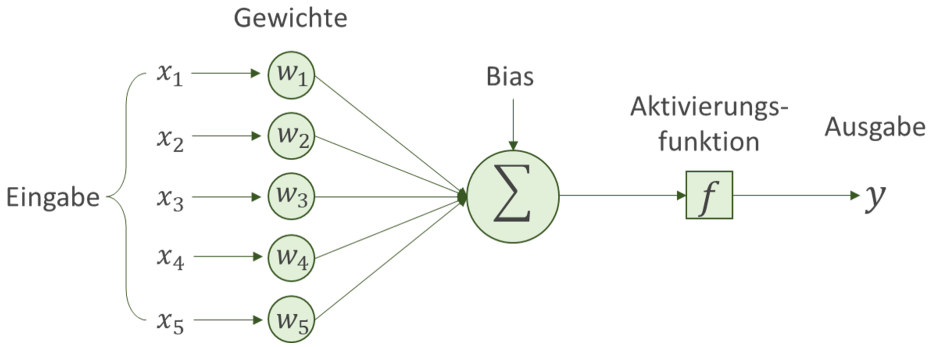



**Abbildung 3:** Schematische Darstellung des mathematischen Modells eines künstlichen Neurons. Die Ausgaben $x$ der vorgeschalteten Neuronen werden mit den Gewichten $W$ multipliziert und in dem nachgeschalteten Neuron aufsummiert. Zu der gewichteten Summe wird ein $Bias$ addiert und das Resultat wird anhand einer Aktivierungsfunktion $f$ verarbeitet, wodurch die Ausgabe $y$ des Neurons erhalten wird.

Jedes Neuron besteht aus einer Aktivierungsfunktion $f$, welche die gewichtete Summe der Ausgaben der vorgeschalteten Neuronen sowie einen Bias als Eingabe erhält [59]. Die Ausgabe ($Out$) jedes Neurons wird demnach anhand folgender Formel [59] bestimmt:

$$Out(m_j) = f(\sum_{i=1}^{k} Out(n_i) \times W_{n_i,m_j} + Bias(m_j)), \tag{1}$$

wobei $n_1, \ldots, n_k$ die Neuronen der Schicht $i$ und $m_1, \ldots, m_l$ die Neuronen der Schicht $i+1$ sind. $w_{n_i m_j}$ ist das Gewicht der Verbindung zwischen Neuron $n_i$ und Neuron $m_j$, $Bias(m_j)$ ist der Bias des Neurons $m_j$ und $f$ beschreibt die Aktivierungsfunktion [59]. Es existieren lineare und nichtlineare Aktivierungsfunktionen. Nichtlineare Aktivierungsfunktionen wie die Rectified Linear Unit (ReLU)-Funktion oder die Sigmoid-Funktion werden benötigt, um nichtlinear separierbare Probleme zu lösen. Die Parameter eines neuronalen Netzes bestehen aus der Gesamtheit der Gewichte und des Bias jedes einzelnen künstlichen Neurons.

Bei einem Feedforward-Netz, der einfachsten Form eines neuronalen Netzes, findet die Übertragung der Information ausschließlich von der Eingabeschicht zu der Ausgabeschicht statt. Die Neuronen in der Eingabeschicht führen keine Berechnungen durch, sondern empfangen die Daten und übertragen sie über die gewichteten Verbindungen an die Neuronen in der ersten verdeckten Schicht

[60]. In den verdeckten Schichten werden die Originaldaten von den einzelnen Neuronen, wie in Gleichung 1 beschrieben, mathematisch verarbeitet und die Ausgabe an die Neuronen in der nächsten Schicht weitergeleitet. Die Ausgabe des gesamten Netzes wird von den Neuronen in der letzten Schicht geliefert. Ziel des neuronalen Netzes ist es, eine gewünschte Ausgabe für eine Reihe von Eingabedaten zu erzeugen. In den meisten Fällen stimmt bei der anfänglich zufälligen Verteilung der Parameter die berechnete Ausgabe nicht mit dem erwarteten Ergebnis überein. Durch Anpassung der Parameter des neuronalen Netzes ist es möglich, die berechnete Ausgabe an die gewünschte Ausgabe anzunähern. Dieser Optimierungsprozess der Parameter des neuronalen Netzes wird als Lernen oder Training bezeichnet.
Das Lernen in neuronalen Netzen erfolgt durch iterative Änderung der Parameter, sodass das Netz schließlich die gewünschte Ausgabe mit einem minimalen Fehler erzeugt. Die Summe der Fehler aller Trainingsbeispiele wird als Gesamtfehler des Netzes bezeichnet. Ein üblicher Weg, einen geeigneten Satz von Parametern zu finden, der den Gesamtfehler minimiert, ist die Durchführung eines Gradientenabstiegsverfahrens, d.h. die Änderung der Parameter, sodass die Änderungen umgekehrt proportional zum Gradienten des Fehlers sind [61]. Im Allgemeinen sucht das Gradientenabstiegsverfahren durch Folgen des Gradienten nach dem Minimum der Fehlerfunktion. Das Training umfasst die Eingabe von Stichproben als Eingangsvektoren, die Berechnung des Fehlers zwischen der angestrebten und der erreichten Ausgabe der Ausgabeschicht und die anschließende Anpassung der Parameter des Netzes zur Minimierung des Fehlers. Die Fehlerfunktion ist häufig durch den mittleren quadratischen Fehler (MSE) zwischen gewünschter Ausgabe $y$ und berechneter Ausgabe $z$ definiert:

$$MSE = \frac{1}{2}||y - z||^2. \tag{2}$$

Nach der Berechnung des Fehlers können die Parameter der einzelnen Neuronen entsprechend ihrer Gradienten aktualisiert werden. Die Anpassung der Parameter jedes Neurons erfolgt mithilfe des Backpropagation-Algorithmus, wobei der Fehler von der Ausgabeschicht bis hin zu der Eingabeschicht durch die Schichten zurückpropagiert wird. Der Backpropagation-Algorithmus ist der am häufigsten verwendete Optimierungsalgorithmus für das Training von Feedforward-Netzen. Weitere Einzelheiten über die Funktionsweise des Backpropagation-Algorithmus sind in [61] gegeben.
Ein Durchlauf des Trainingssatzes zum Trainieren des neuronalen Netzes wird als Trainingsepoche bezeichnet und kann zur Verbesserung des Gesamtfehlers beliebig oft wiederholt werden. Da die Bewertung des Fehlers unter Verwendung des gesamten Trainingssatzes eine hohe Rechenleistung erfordert, wird der Verlust für jeden Trainingsschritt häufig unter Verwendung einer kleinen Anzahl von zufällig ausgewählten Trainingsbeispielen, auch Trainingsbatch genannt, berechnet. Die Menge der Trainingsbeispiele in einem Trainingsbatch wird als Batchgröße bezeichnet.
Das Lernen kann grob in verschiedene Lernverfahren unterteilt werden, welche je nach Anwendungszweck und zu untersuchenden Daten gewählt werden. Im Folgenden werden vier verschiedenen Lernverfahren (überwacht, unüberwacht, selbst-überwacht und teilüberwacht) näher erläutert, um eine bessere Eingrenzung der nachfolgenden Algorithmen zu vermitteln.
Die Aufgabe der Algorithmen beim **überwachtem Lernen** besteht darin, anhand von Eingabe-Ausgabe-Paaren eine Funktion, welche die Eingabe auf die Ausgabe abbildet, zu erlernen. Durch die Verfügbarkeit der jeweiligen Ausgaben lässt sich prüfen, ob der Algorithmus die Ausgabe korrekt

vorhergesagt hat. Die bekanntesten Algorithmen des überwachten maschinellen Lernens sind die Klassifizierung und die Regression [62].
**Unüberwachtes Lernen** wird oft als überwachtes Lernen mit unbekannter Ausgabe bezeichnet [63]. Daher besteht das Ziel der Algorithmen des unüberwachten Lernens darin, Muster und Zusammenhänge in den Eingaben eigenständig (ohne Ausgabe) zu erkennen. Zu den Algorithmen des unüberwachten Lernens gehören beispielsweise die Dimensionsreduktion sowie die Clusteranalyse.
Eine Teilmenge des unüberwachten Lernens ist das **selbstüberwachte Lernen**. Bei dem selbstüberwachten Lernen werden die Algorithmen explizit mit Überwachungssignalen, die aus den Eingabedaten selbst generiert werden, trainiert.
**Teilüberwachtes Lernen** kann als eine Kombination aus überwachtem und unüberwachtem Lernen betrachtet werden. Der Datensatz besteht beim teilüberwachten Lernen sowohl aus Eingabe-Ausgabe-Paaren als auch aus Eingaben mit unbekannter Ausgabe [64].
Ein neuronales Feedforward-Netzwerk mit mehr als einer verborgenen Schicht nennt man tiefes neuronales Netz [65]. Das tiefe neuronale Netz ist eine komplexere Variante des flachen ANN [66] und gehört zu den DL-Methoden. In den letzten Jahren haben ML- und insbesondere DL-Verfahren erhebliche Fortschritte gemacht. DL-Algorithmen können im Gegensatz zu herkömmlichen ML-Modellen abstrakte Repräsentationen aus Daten durch die erhöhte Anzahl verborgener Schichten extrahieren, wobei jede Schicht eine zusätzliche Abstraktionsebene darstellt [64].

### 2.3.1 Selbstorganisierende Karte

Das Konzept der SOM wurde 1982 von Kohonen eingeführt [67]. Die SOM ist ein ANN, das eine niedrigdimensionale Darstellung der Eingabedaten auf einer zugrunde liegenden Mannigfaltigkeit unüberwacht lernt. Kohonens SOM bildet die Eingabedaten auf ein (häufig zweidimensionales) regelmäßiges Gitter aus Neuronen ab, wodurch komplexe, nichtlineare Beziehungen zwischen den hochdimensionalen Eingabedaten topologieerhaltend in einfache Beziehungen auf einem niedrigdimensionalen Gitter transformiert werden. Jedem Neuron im Gitter wird ein Gewichtsvektor $\vec{W} = (W_1, W_2, W_3, \ldots, W_d)$, mit der Dimensionalität $d$ der Eingabedaten $\vec{x} = (x_1, x_2, x_3, \ldots, x_d)$ zugeordnet. Zu Beginn werden die Gewichte zufällig oder durch gleichmäßiges Abtasten des Unterraums initialisiert. Aus dem Eingaberaum werden anschließend zufällig Stichproben in das Netzwerk eingespeist und die Distanz (üblicherweise die euklidische Distanz) zu den Gewichtsvektoren $\vec{W}$ berechnet. Das Neuron mit der kleinsten euklidischen Distanz wird als Best Matching Unit (BMU) bezeichnet. Die Gewichte der BMU $c$ sowie die Gewichtsvektoren, die im Neuronengitter innerhalb eines Nachbarschaftsradius $N_c(t)$ um $c$ liegen, werden unter Verwendung der folgenden Formel aktualisiert [67]:

$$W_i(t-1) = W_i(t) + \alpha(t)(x(t) - W_i(t)), \text{ wenn } i \in N_c(t). \tag{3}$$

$t$ ist der Index des Regressionsschritts und $x(t)$ ist demnach die aktuell präsentierte Probe. Die Lernrate $\alpha$ nimmt über die Iterationen $t$ ab. Alle Gewichtsvektoren, die außerhalb des Nachbarschaftsradius liegen, bleiben unverändert.
Jeder hochdimensionale Datenpunkt wird somit in ein einzelnes Neuron eingebettet, das seine

Struktur am genauesten reproduziert. Der wesentliche Punkt bei der Anwendbarkeit von SOM ist der topologische Charakter des Embeddings: Ähnliche Muster werden auf nahe gelegene Neuronen der Karte abgebildet. Dementsprechend lässt sich die SOM auch als Clusteranalyse-Verfahren verwenden.

### 2.3.2 Autoencoder

Die Idee des AEs wurde 1987 von LeCun als unüberwachtes oder auch selbstüberwachtes lineares Merkmalsextraktionsverfahren eingeführt [68]. Der traditionelle AE ist ein künstliches neuronales Feedforward-Netz mit symmetrischer Struktur, welches aus zwei Teilen, einem Encoder und einem Decoder besteht, wie in Abbildung 4 dargestellt ist. Spezifisch für die Architektur des AEs ist die Dimension der Ausgabeschicht, welche der Dimension der Eingabeschicht entspricht.

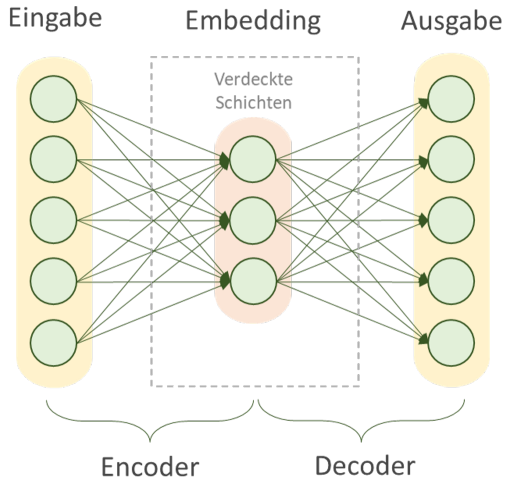



**Abbildung 4:** Veranschaulichung des minimalen Autoencoder-Modells. Die Eingabe wird durch den Encoder in eine niedrigdimensionale Darstellung (Embedding) abgebildet und anschließend durch den Decoder in die Ausgabe decodiert. Ziel des Decoders ist es, die Roheingabe aus dem Embedding zu rekonstruieren. Der zu minimierende Rekonstruktionsverlust ist die Differenz zwischen Ausgabe und Eingabe des Netzes.

Die kleinste Konfiguration des AEs beinhaltet eine Eingabe- und Ausgabeschicht sowie eine zwischengelagerte verdeckte Schicht, die in dieser Arbeit als Embedding-Schicht bezeichnet wird. Der Encoder bildet die hochdimensionalen Eingabedaten $\vec{x} = (x_1, x_2, x_3, \ldots, x_d)$ anhand eines deterministischen Mappings $\vec{y} = f(\vec{x}) = s(W\vec{x} + \vec{b})$ in eine niedrigdimensionale Repräsentation $\vec{y} = (y_1, y_2, y_3, \ldots, y_q)$ ab. Dabei bezeichnet $W$ die $q \times d$ - Gewichtsmatrix und $\vec{b}$ den Bias-Vektor mit der Dimensionalität $q$ [69]. Abschließend bildet der Decoder $\vec{z} = g(\vec{y}) = s(W'\vec{y} + \vec{b'})$ die latente Repräsentation $\vec{y}$ auf einen $d$-dimensionalen Vektor $\vec{z} = (z_1, z_2, z_3, \ldots, z_d)$ im Eingaberaum ab [69]. Der AE lässt sich somit im Ganzen durch die Rekonstruktionsfunktion $g(f(\vec{x})) = \vec{z}$ beschreiben, wobei $\vec{z}$ an die ursprüngliche Eingabe $\vec{x}$ angenähert wird. Um die Eingabedaten aus der Ausgangsschicht zu rekonstruieren, wird der Parametersatz $(W, \vec{b}, W', \vec{b'})$ durch Minimierung des Rekonstruktionsverlustes $L(\vec{x}, \vec{z})$ optimiert. Der Rekonstruktionsverlust $L$ bestraft die Differenz

zwischen dem Eingangsvektor $\vec{x}$ und dem Ausgangsvektor $\vec{z}$. Im traditionellen AE-Modell ist der Rekonstruktionsverlust durch den MSE (Gleichung 2) zwischen dem Eingangsvektor $\vec{x}$ und dem Ausgabevektor $\vec{z}$ definiert. Ein ideales neuronales AE-Netz zielt darauf ab, die Eingabedaten perfekt zu rekonstruieren. Die triviale, aber aussichtslose Lösung besteht darin, die Identitätsfunktion $f(\vec{x}) = \vec{x}$ zwischen der Eingabe- und der Ausgabeschicht zu erlernen. Damit der AE nicht die Identitätsfunktion lernt, ist eine zusätzliche Regularisierung erforderlich. Die gebräuchlichste Option besteht darin, die Mächtigkeit der Neuronen in der Embedding-Schicht zu reduzieren, sodass $q < d$. Auf diese Weise wird ein sogenannter Flaschenhals eingeführt. Dies dient auch direkt dem Ziel, eine niedrigdimensionale Darstellung der Daten zu erhalten.
Baldi und Hornik haben 1989 die Beziehung zwischen einem einschichtigen AE und der PCA untersucht [70]. Wenn der Decoder linear ist und die Verlustfunktion durch den MSE gegeben ist, lernt ein nicht vollständiger AE, denselben Unterraum wie die PCA in der Embedding-Schicht aufzuspannen. Allerdings ist die Rechenkomplexität des Trainings eines AEs viel höher als die der PCA, wodurch die Anwendung von linearen AEs eingeschränkt ist. Mit nicht linear auferlegten Schichten kann der AE eine leistungsfähigere, nicht lineare Verallgemeinerung der PCA lernen [17]. Daher liegt die Vermutung nahe, dass AEs aufgrund ihrer nicht linearen Fähigkeiten in der Lage sind, komplexere Beziehungen der Eingabedaten zu lernen.

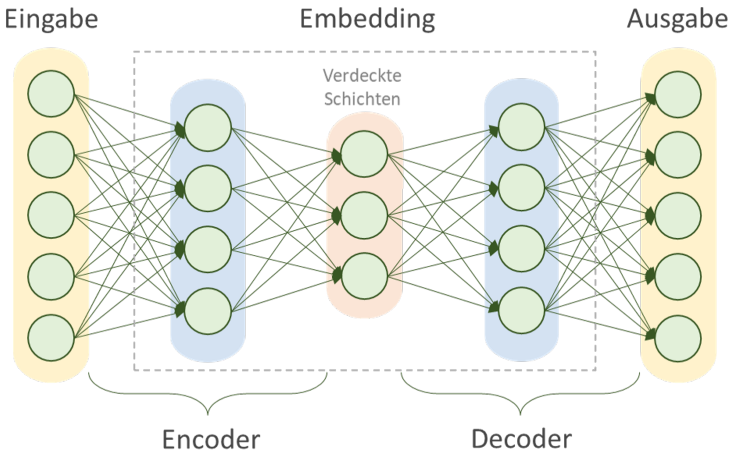



**Abbildung 5:** Schematische Darstellung eines Deep Autoencoder-Modells. Die Eingabe wird durch die verdeckten Schichten des Encoders in eine niedrigdimensionale Darstellung (Embedding) abgebildet und anschließend durch die verdeckten Schichten des Decoders in die Ausgabe decodiert. Ziel des Decoders ist es, die Roheingabe aus dem Embedding zu rekonstruieren. Der zu minimierende Rekonstruktionsverlust ist die Differenz zwischen Ausgabe und Eingabe des Netzes.

Zur Extraktion geeigneter latenter Repräsentationen wurden bereits eine Reihe verschiedener Arten von AEs vorgeschlagen. Eines der AE-Modelle ist der Deep Autoencoder (DAE), welcher sich durch mehr als eine verdeckte Schicht auszeichnet. In Abbildung 5 ist der allgemeine Aufbau eines DAEs veranschaulicht. Aufgrund der symmetrischen Struktur der AE-Modelle werden verdeckte Schichten paarweise (Encoder und Decoder) zu den AE-Modellen hinzugefügt. Durch die vielen Schichten des DAE können nicht lineare Zusammenhänge über die Eingabedaten effektiv gelernt werden. Die

in dieser Arbeit verwendeten AE-Modelle besitzen alle eine tiefe Struktur. Im Folgenden werden die neben dem DAE verwendeten AE-Modelle näher erläutert.

#### 2.3.2.1 Stacked Denoising Autoencoder

Der SDAE [71] ist eine Kombination aus dem Denoising AE [68] und dem Stacked AE [72, 73, 74]. Der **Denoising AE** ist eine spezielle Form des klassischen AEs, jedoch verwendet der Denoising AE, anders als der herkömmliche AE, eine durch stochastisches Rauschen korrumpierte Version $\tilde{x}$ der Eingabedaten $x$. In Abbildung 6 ist eine schematische Darstellung des Denoising AEs dargestellt.

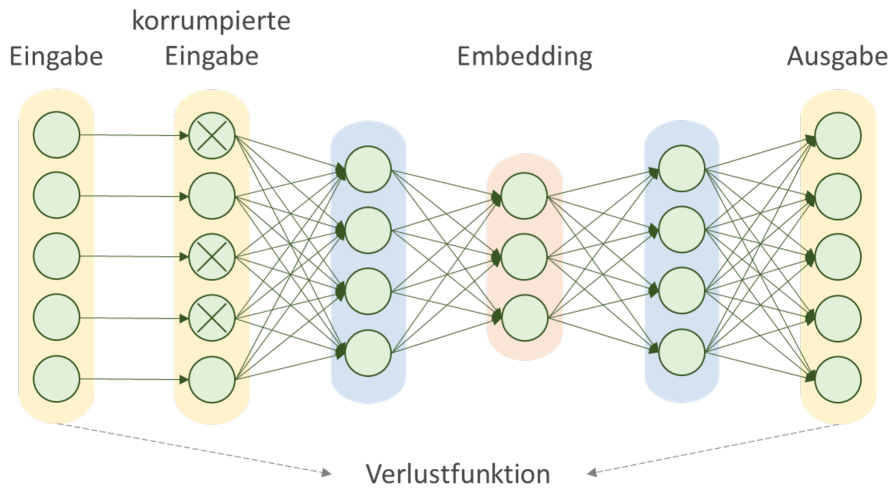



**Abbildung 6:** Schematische Darstellung eines Denoising Autoencoder-Modells. Die rohe Eingabe wird durch eine Form von Rauschen korrumpiert (korrumpierte Eingabe). Die schwarzen Kreuze verdeutlichen die Korrumpierung der Eingabedaten. Aus der korrumpierten Eingabe wird durch den Encoder ein Embedding generiert. Anschließend versucht der Decoder, die Roheingabe aus dem Embedding zu rekonstruieren. Zur Berechnung des Rekonstruktionsverlustes werden die ursprünglichen, nicht korrumpierten Eingabedaten mit der Ausgabe des Decoders verglichen.

Die Regularisierung durch die Korrumpierung der Daten verhindert das Lernen der Identitätsfunktion, weshalb der Denoising AE auch ohne Einführung eines Flaschenhalses zur Rauschunterdrückung von Daten verwendet werden kann [75]. Bei Verwendung des Denoising-Kriteriums als zusätzliche Regularisierung neben der Verwendung eines Flaschenhalses sind die von einem DAE gelernten Merkmale robust gegenüber verrauschten Eingabedaten [69]. Der Denoising AE wird darauf trainiert, die ursprünglichen Eingabedaten aus den korrumpierten Daten zu rekonstruieren. Dazu wird zunächst die ursprüngliche Eingabe $x$ durch eine Form von Rauschen in $\tilde{x}$ umgewandelt. Die verfälschte Eingabe $\tilde{x}$ wird dann wie beim klassischen AE auf eine verborgene Darstellung $y = f(\tilde{x}) = s(W\tilde{x} + b)$ abgebildet, aus der $z = g(y)$ rekonstruiert wird [71]. Um die ursprünglichen Eingabedaten aus den beschädigten Daten zu rekonstruieren, wird der Rekonstruktionsverlust zwischen der Ausgabe des Netzes $z$ und der ursprünglichen, unkorrumpierten Eingabe $x$ minimiert.
Zur Leistungssteigerung des klassischen AEs kann der sogenannte **Stacked AE** verwendet werden.

Dieser besitzt eine erhöhte Komplexität der Konfiguration sowie des Trainingsalgorithmus im Vergleich zu dem klassischen AE [64]. Hinton et al. zeigten, dass das Stapeln von restricted Boltzmann-Maschinen eine gute Strategie für die Initialisierung der Gewichte eines tiefen ANNs darstellt [76]. Dieser Ansatz kann auf AE ausgeweitet werden, wie von Bengio et al. [72] gezeigt wurde. Im Allgemeinen umfasst der Trainingsprozess von **Stacked AEs** eine unüberwachte Pretraining-Phase und eine überwachte Finetuning-Phase, wie in Abbildung 7 gezeigt. In Abbildung 7 ist der SDAEs schematisch dargestellt.

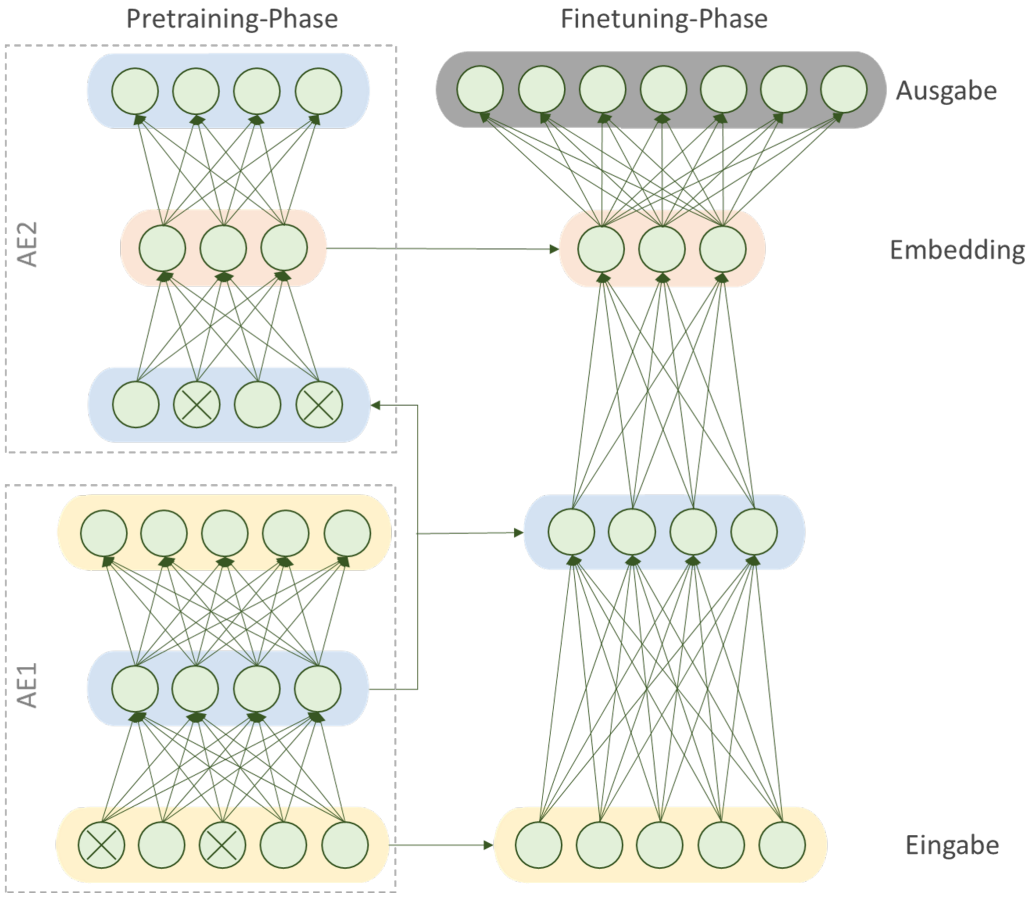



**Abbildung 7:** Schematische Darstellung eines Stacked Denoising Autoencoder-Modells. Im Allgemeinen besteht der Trainingsprozess aus einer unüberwachten Pretraining-Phase und einer überwachten Finetuning-Phase. In der Pretraining-Phase (links) wird jede verdeckte Schicht als separater Denoising AE (AE1, AE2, ...) trainiert, wobei die von dem vorherigen Denoising AE (AE1) aus einer nicht korrumpierten Eingabe generierte Repräsentation als Eingabe für das nächste Denoising AE-Modell (AE2) verwendet wird. In der Finetuning Phase werden die Parameter und die Struktur der Encoder der einzelnen AE-Modelle als Initialisierung für ein neuronales Netz mit einer zusätzlichen Ausgabeschicht verwendet, das im Hinblick auf ein überwachtes Trainingskriterium optimiert wird. AE = Autoencoder.

Das unbeaufsichtigte Pretraining wird Schicht für Schicht durchgeführt, wobei die generierte Repräsentation des vorherigen AEs als Eingabe des folgenden AEs verwendet wird. Dieser Ansatz führt nachweislich zu einer besseren Generalisierung [69]. Beim SDAE wird jede verdeckte Schicht als separater Denoising AE trainiert, indem der Rekonstruktionsfehler zwischen der ursprünglichen Eingabe und der Rekonstruktion minimiert wird. Dabei wird der Korruptionsprozess des Denoising AEs nur während des Trainings verwendet, jedoch nicht für die Weitergabe der Repräsentationen

an nachgeschaltete AE-Modelle [71]. Sobald die Pretraining-Phase abgeschlossen ist, durchläuft das Netzwerk ein überwachtes Finetuning. Dazu werden sowohl die Parameter als auch die Struktur der Encoder-Schichten der einzelnen AE-Modelle als Initialisierung für ein überwachtes neuronales Netz verwendet, das anhand eines Klassifikationsproblems optimiert wird (Abbildung 7). Es hat sich gezeigt, dass die Initialisierung der Gewichte mithilfe dieses greedy, schichtweisen Trainingsverfahrens, den Optimierungsprozess im Vergleich mit der zufälligen Initialisierung der Gewichte verbessert [72].

#### 2.3.2.2 Variational Autoencoder

Der VAE ist zurückzuführen auf Kingma und Welling [77] und kann als generative Variante des traditionellen AEs betrachtet werden. Während der traditionelle AE die Daten im Rekonstruktionsprozess komprimiert, zielt der VAE darauf ab, die unbekannte, zugrunde liegende gemeinsame Wahrscheinlichkeitsverteilung $p^*(y|x)$ der latenten Variablen $y$ in Abhängigkeit von den beobachteten Variablen $x$ zu modellieren [28]. Die zugrunde liegende bedingte Wahrscheinlichkeitsverteilung $p^*(y|x)$ wird durch ein Modell $p_\theta(y|x)$ mit den Parametern $\theta$ approximiert. Ein Beispiel für ein VAE-Modell ist in Abbildung 8 dargestellt.

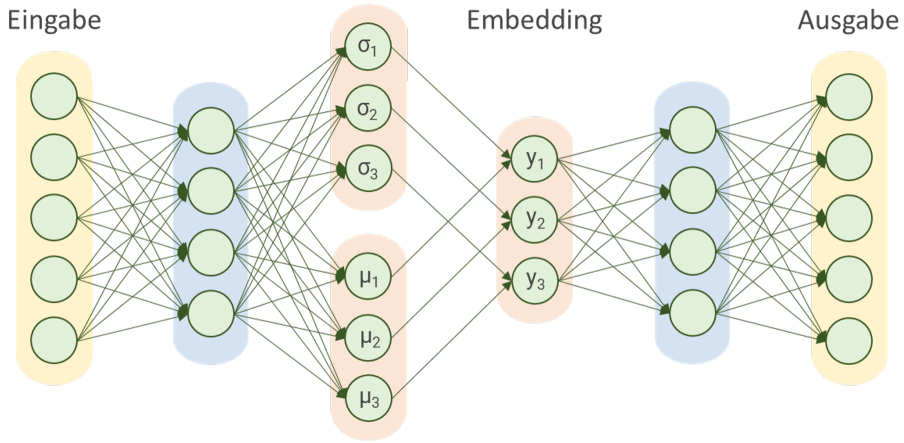



**Abbildung 8:** Schematische Darstellung eines Variational Autoencoder-Modells mit latenter Gauß-Verteilung. Der Encoder approximiert die Posteriorverteilung mit der Gaußschen Normalverteilung. Der latente Vektor ist dementsprechend eine Wahrscheinlichkeitsverteilung, die aus dem Mittelwert $\mu$ und der Standardabweichung $\sigma$ besteht. Aus dieser Verteilung wird ein latenter Vektor $\vec{y}$ abgetastet und durch den Decoder in eine Rekonstruktion der Eingabedaten decodiert.

Der VAE besteht wie die zuvor beschriebenen AEs aus einem Encoder- und einem Decoder-Netzwerk [78]. Der Encoder $q_\phi(y|x)$ ist ein parametrisches Inferenzmodell, das die wahre, schwer zu ermittelnde bedingte posterior Verteilung $p_\theta(y|x)$ der latenten Variablen $y$ bei gegebenen Eingangsdaten $x$ des generativen Modells mit einer Verteilung (häufig der Gauß-Verteilung) approximiert. Das zu optimierende Problem besteht demnach in der Annäherung von $q_\phi(y|x)$ an die posterior Verteilung $p_\theta(y|x)$ durch Anpassung der Modellparameter $\phi$, sodass für beliebige x und y gilt: $q_\phi(y|x) \approx p_\theta(y|x)$. Aufgrund des Inferenz-Encoder-Modells ist die Ausgabe des Encoder-Netzes, anders als bei den vorherigen AEs, kein deterministischer latenter Vektor $\vec{y}$, sondern eine Wahr-

scheinlichkeitsverteilung, die im Falle der Gaußschen Normalverteilung aus dem Mittelwert $\mu$ und der Standardabweichung $\sigma$ besteht, wie in Abbildung 8 dargestellt ist. Aus dieser Verteilung wird ein latenter Vektor $\vec{y}$ abgetastet und durch den generativen Decoder $p_\theta(x|y)$ in eine Rekonstruktion der Eingabedaten $\vec{x}$ decodiert. Daraus ergibt sich die Möglichkeit zur Optimierung des Parameters $\Theta$ durch Maximierung der bedingten Wahrscheinlichkeit $p_\theta(x|y)$ zwischen den beobachteten Variablen $x$ und den generierten Variablen $y$. Die Parameter $\phi$ und $\theta$ entsprechen der Gewichtsmatrix des Encoder- und Decoder-Netzes. Der Trainingsprozess des VAE umfasst die Optimierung der Parameter $\phi$ und $\theta$ mithilfe des Reparametrisierungs Tricks zur Maximierung der Evidence Lower Bound (ELBO) $L_{\theta,\phi}$. Die ELBO besteht aus zwei Termen und ist wie folgt definiert [78]:

$$\mathcal{L}_{\theta,\phi}(x) = E_{q_\phi(y|x)}[\log p_\theta(x|y)] - D_{KL}(q_\phi(y|x) \parallel p_\theta(y)). \tag{4}$$

Der erste Term der ELBO repräsentiert den klassischen Rekonstruktionsfehler und der zweite Term die Kullback-Leibler-Divergenz, welche die Differenz zwischen der zugrunde liegenden bedingten Wahrscheinlichkeitsverteilung $p_\theta(y|x)$ und der tatsächlichen Verteilung $q_\phi(y|x)$ durch den Encoder bestraft [6].

#### 2.3.2.3 Siamesisches neuronales Netz

Das SNN wurde erstmals in den frühen 1990er-Jahren von Bromley et al. zur Signaturverifizierung eingeführt [79]. Neben der Signaturverifizierung wurden SNN für die One-Shot-Bildklassifizierung [80], die Gesichtsverifizierung, bei der die Kategorien nicht im Voraus bekannt sind [81] sowie für die Dimensionsreduktion [19] verwendet. Das SNN führt, anders als die die oben genannten AE-Modelle, eine Dimensionsreduktion auf überwachte Weise durch. In Abbildung 9 ist die typische Struktur eines SNN zu sehen.
Ein SNN ist eine Art neuronales Netz, das zwei Komponenten besitzt: ein Zwillings-Netz und eine Ähnlichkeits- oder Distanzfunktion [80]. Das **Zwillings-Netz** besteht aus mehreren Eingabeschichten, typischerweise zwei oder drei, jeweils gefolgt von einem Encoder-Netz zum Extrahieren von Merkmalen aus den Eingabedaten. Die einzelnen Encoder-Netze des Zwillings-Netzes sind hinsichtlich ihrer Struktur, Gewichte und Parameter identisch [82]. Aufgrund dieser Übereinstimmungen sind die Encoder-Netze gezwungen, dieselbe Transformation zu erlernen, wodurch garantiert wird, dass zwei ähnliche Eingaben im latenten Raum nahe beieinander abgebildet werden [80]. Die durch die Encoder-Netze erlernten Merkmalsvektoren werden anhand einer **Ähnlichkeits oder Distanz-Funktion** (Euklidische Distanz, Kosinus-Ähnlichkeit) verglichen. Das Ziel des Trainings ist es, den Abstand zwischen ähnlichen Paaren (positive Eingabepaare) im latenten Raum zu minimieren und den Abstand zwischen unähnlichen Paaren (negative Eingabepaare) zu maximieren (Abbildung 10), um die Generalisierungsfähigkeit zu verbessern [82]. Diese Lernmethode, eine Metrik aus Datenpaaren (oder Triplets) zu spezifizieren, wird auch Metric Learning genannt [83, 84, 85, 86].
Für das Training eines SNN sind verschiedene Verlustfunktionen vorgeschlagen worden. Die logische Wahl ist die binäre Kreuzentropieverlustfunktion, aufgrund der Tatsache, dass SNNs eine binäre Klassifizierung durch Unterteilung in ähnliche und unähnliche Eingabepaare vornehmen [87]. Bei Verwendung des **binären Kreuzentropieverlustes** besteht das SNN aus zwei Eingabeschichten,

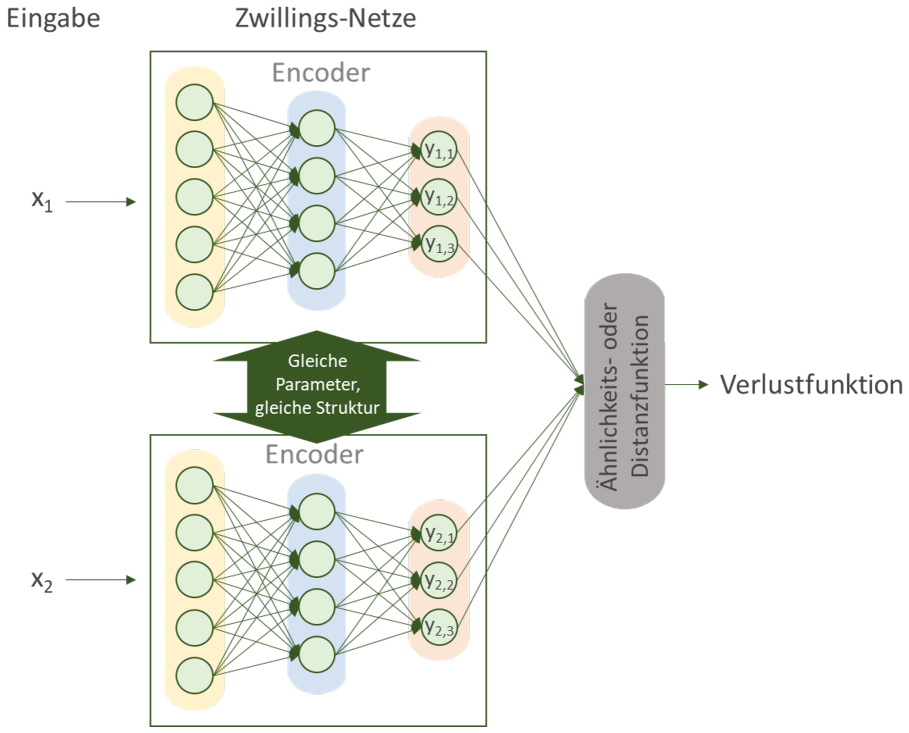



**Abbildung 9:** Schematische Darstellung eines siamesischen neuronalen Netzes. Das hier dargestellte SNN besteht aus zwei Komponenten: dem Zwillings-Netz und einer Ähnlichkeits- oder Distanzfunktion. Das Zwillings-Netz des hier dargestellten SNN besteht aus zwei identischen Encoder-Netzen, die zwei Vektoren $\vec{x_1}$ und $\vec{x_2}$ als Eingabe erhalten und zwei Embeddingvektoren $\vec{y_1}$ und $\vec{y_2}$ ausgeben. Die Embeddingvektoren $\vec{y_1}$ und $\vec{y_2}$ werden mittels einer Ähnlichkeits- oder Distanzfunktion verglichen. Die Netzwerkparameter werden von den Encoder-Netzen gemeinsam genutzt und durch Minimierung der Distanz zwischen den Embeddingvektoren trainiert.

jeweils gefolgt von einem Encoder-Netz, wie in Abbildung 9 gezeigt. Die von den Encoder-Netzen generierten Zwillings-Merkmalsvektoren werden mittels einer Distanzfunktion in Kombination mit einer sigmoidalen Aktivierung, die auf das Intervall [0,1] abbildet, verbunden. Die Trainingsmenge für ein solches Netz besteht aus positiven und negativen Eingabepaaren in Kombination mit einem Label $l \in \{0, 1\}$. Ein positives Eingabepaar bezeichnet ein Paar von Eingaben, welche derselben Klasse angehören ($l = 1$), wohingegen ein negatives Eingabepaar ein Paar von Eingaben, welche unterschiedlichen Klassen angehören ($l = 0$), definiert. In einem Zwei-Klassen-Problem kann die binäre Kreuzentropie für ein Eingabepaar ($x_1$, $x_2$) wie folgt berechnet werden [80]:

$$\mathcal{L}(x_1, x_2) = l_i * \log(z_i) + (1 - l_i) * \log(1 - z_i), \tag{5}$$

wobei $l_i$ das zu dem jeweiligen Eingabepaar ($x_1$, $x_2$) gehörende binäre Label ist. Für ein positives Eingabepaar ist $l_i$=1 und für ein negatives Eingabepaar ist $l_i$=0. Das vorhergesagte Label $z_i$ entspricht der Ausgabe des SNN.

Neben dem binären Kreuzentropieverlust wurden zwei weitere Verlustfunktionen vorgeschlagen: der Tripletverlust [82] und der kontrastive Verlust [19]. Im Folgenden werden die erwähnten Verlustfunktionen näher erläutert.
Ein SNN, welches den **kontrastiven Verlust** minimiert, ähnelt dem SNN mit binärem Kreuzentro-

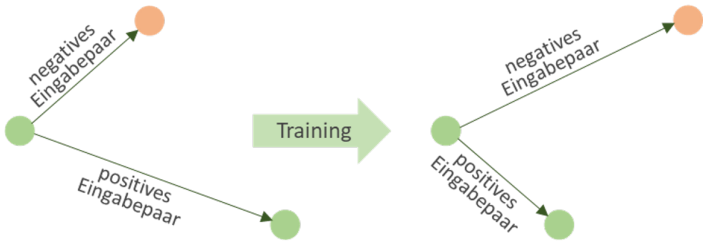



**Abbildung 10:** Training des SNN. Die Verlustfunktion des SNN minimiert den Abstand zwischen zwei Stichproben derselben Klasse (positives Eingabepaar) und maximiert den Abstand zwischen zwei Stichproben unterschiedlicher Klassen (negatives Eingabepaar). In Anlehnung an Schroff et al. [82].

pieverlust. Die Trainingsmenge entspricht der oben beschriebenen Trainingsmenge. Der Unterschied des grundsätzlichen Aufbaus der Netze liegt in der Verwendung der sigmoidalen Aktivierung. Diese wird in Zusammenhang mit dem kontrastiven Verlust nicht verwendet. Die kontrastive Verlustfunktion für ein Eingabepaar wird wie folgt berechnet:

$$\mathcal{L}(x_1, x_2, l_i) = l\frac{1}{2}D^2 + (1 - l_i)\frac{1}{2}\{max(0, m - D\}^2, \tag{6}$$

wobei $l_i$ das einem Eingabepaar $x_1$, $x_2$ zugewiesene binäre Label ist. $l_i = 1$, wenn $x_1$ und $x_2$ der gleichen Klasse angehören (positives Paar) und $l_i = 0$, wenn $x_1$ und $x_2$ unterschiedlichen Klassen angehören (negatives Paar). D ist die Distanz zwischen den latenten Vektoren $y_1$ und $y_2$ des Eingabepaars $x_1$, $x_2$. Die Margin $m > 0$ definiert den Abstand, den ein negatives Paar haben muss, um nicht mehr zur Verlustfunktion beizutragen.
Unter Verwendung des **Tripletverlustes** besteht ein SNN aus drei Eingabeschichten (im Gegensatz zu zwei Eingabeschichten wie bei dem binären Kreuzentropieverlust und dem kontrastiven Verlust), jeweils gefolgt von einem Encoder-Netz. Die Trainingsmenge besteht dementsprechend aus drei Stichproben (Triplet): dem Anker, einer positiven und einer negativen Stichprobe, wobei der Anker und die positive Stichprobe derselben Klasse angehören und der Anker und die negative Stichprobe unterschiedlichen Klassen angehören. Ein Label wird nicht benötigt. Der Verlust für ein Triplet $(x_a, x_p, x_n)$ wird wie folgt berechnet:

$$\mathcal{L}(x_a, x_p, x_n) = max(0, D_{a,p} - D_{a,n} + m), \tag{7}$$

wobei $D_{a,p}$ die Distanz zwischen $x_a$ (Anker) und $x_p$ (Positiv) ist und dementsprechend ist $D_{a,n}$ die Distanz zwischen $x_a$ (Anker) und $x_n$ (Negativ). Die Margin $m > 0$ ist die Distanz, welche zwischen Anker und negativer Stichprobe erzwungen wird [82].

### 2.3.3 Allgemeine Eigenschaften der Dimensionsreduktionsmethoden

Dimensionsreduktionsmethoden können im Hinblick auf eine Vielzahl von Eigenschaften verglichen werden. Die als zentral erachteten Eigenschaften für die zuvor vorgestellten Dimensionsreduktionsmethoden sind die Folgenden:

- **linear versus nichtlinear**,
- **unüberwacht versus überwacht**,
- **traditionell versus DL**.

Zur besseren Differenzierung der präsentierten Dimensionsreduktionsmethoden werden die zentralen Eigenschaften der Verfahren in Tabelle 1 zusammengefasst.

**Tabelle 1:** Allgemeine Eigenschaften der Dimensionsreduktionsmethoden. DL = Deep Learning; PCA = Hauptkomponentenanalyse; SOM = Selbstorganisierende Karte; DAE = Deep Autoencoder; SDAE = Stacked Denoising Autoencoder; VAE = Variational Autoencoder; SNN = siamesisches neuronales Netz.

| **Verfahren** | **nichtlinear?** | **überwacht?** | **DL?** |
|---|---|---|---|
| PCA | Nein | Nein | Nein |
| SOM | Ja | Nein | Nein |
| DAE | Ja | Nein | Ja |
| SDAE | Ja | Nein | Ja |
| VAE | Ja | Nein | Ja |
| SNN | Ja | Ja | Ja |

Die PCA ist als einziges in dieser Arbeit verwendetes Modell nicht in der Lage, nichtlineare Daten zu verarbeiten. Das SNN verwendet im Gegensatz zu den restlichen Verfahren zusätzlich Klasseninformationen, um im latenten Raum den Abstand zwischen ähnlichen Paaren zu minimieren und den Abstand zwischen unähnlichen Paaren zu maximieren [82]. Die PCA sowie die SOM gehören nicht zu den DL-Verfahren. Auf Grundlage dieser Eigenschaften sollte das SNN die weiteren in dieser Arbeit untersuchten Dimensionsreduktionsverfahren übertreffen.

## 2.4 Verwandte Arbeiten

Tiefe neuronale Netze konnten in den Bereichen Computer Vision und Natural Language Processing bereits signifikante Erfolge erzielen [9, 10]. Motiviert durch diese Erfolge wurden in den letzten Jahren verschiedene DL-Ansätze auf hochdimensionale Genexpressionsdatensätze angewendet. Einzelne Verfahren wurden bereits in einer Vielzahl von Publikationen zur Dimensionsreduktion von Genexpressionsdaten verwendet.
Yeung et al. untersuchten die Auswirkungen von Hauptkomponenten zur Erfassung von Clusterstrukturen aus synthetischen und realen Genexpressionsdaten [88]. Die Untersuchungen zeigten eine Verschlechterung der Ergebnisse unter Verwendung der Hauptkomponenten. Fakoor et al. schlugen eine Methode zum Erlernen einer spärlichen Repräsentation von Genexpressionsdaten vor, bei welcher die Dimensionalität des Merkmalsraums durch eine Kombination aus PCA und Sparse-AE reduziert wurde [89]. Die erlernte Repräsentation konnte eine Verbesserung der Genauigkeit bei Krebsklassifizierungsproblemen erreichen. Tan et al. verwendeten einen Denoising AE zur Identifizierung und Extrahierung komplexer Muster aus Genexpressionsdaten [21]. Die resultierenden Embeddings konnten zur Generierung von Schlüsselmerkmalen verwendet werden. Gupta et al.

schlugen eine tiefe Architektur des Denoising-AEs zur Extraktion von Merkmalen aus Genexpressionsdaten vor [22]. Die Fähigkeit von tiefen neuronalen Netzen, die Eingabeverteilung der Genexpressionsdaten zu lernen, wurde mittels Gen-Clustering gemessen. Die Ergebnisse des Denoising-AEs wurden mit der Roheingabe und der traditionellen PCA verglichen. Es konnte gezeigt werden, dass selbst ein flacher Denoising-AE sowohl die Rohdaten als auch die PCA bei der Aufgabe des Gen-Clusterings übertreffen kann. Teixeira et al. verwendeten einen SDAE zur Merkmalsextraktion aus Genexpressionsdaten [23]. Anhand der Gewichtsmatrix wurden die relevantesten Gene bestimmt. Auf Grundlage der Liste der relevantesten Genen wurde eine funktionelle Annotations-Clustering-Analyse durchgeführt. Die Ergebnisse zeigten gute Resultate bei der Extraktion der Merkmale, jedoch konnte die Wissensextraktion keine vielversprechenden Resultate liefern. Danaee et al. stellten einen DL-Ansatz zur Krebserkennung und zur Identifizierung von Genen auf Genexpressionsdaten vor [24]. Ein SDAE-Modell wurde zur Merkmalsextraktion von hochdimensionalen Genexpressionsdatensätzen verwendet und die Leistung der ausgewählten Merkmale mithilfe eines Klassifizierungsmodells bewertet. Mit Hilfe der SDAE-Konnektivitätsmatrizen konnten mehrere hochgradig interaktive Gene, die als Krebs-Biomarker behandelt werden können, identifiziert werden. Ai et al. wendeten einen VAE auf Microarray-Daten mit bereits im Vorfeld selektierten Genen an [26]. Die aus dem VAE generierten Embeddings wurden in Kombination mit 10 Hub-Genen zum Trainieren eines Support Vector Machine (SVM)-Klassifikators verwendet. Die Autoren konnten eine Accuracy von 96,92 % bei der Krebsvorhersage erzielen. Zhang et al. schlugen ein End-to-End-DL-Modell (OmiVAE) vor, welches eine überwachte Merkmalsextraktion und Mehrklassenklassifizierung von verschiedenen Tumortypen durchführt [90]. OmiVAE ist eine Kombination aus der Grundstruktur eines VAE und einem Klassifikator. OmiVAE erreicht eine durchschnittliche Genauigkeit von 97,49 % bei 33 Tumortypen und der Kontrolle. Zhang et al. erweiterten diese Modell zu einem Multitask-Deep-Learning-Framework für Multi-Omics-Daten (OmiEmbed) [6]. OmiEmbed besteht aus einem VAE zur Generierung eines Embeddings sowie nachgeschalteten Downstream-Task-Netzwerken zur Erfassung molekularer Phänotypen aus hochdimensionalen Omics-Daten. OmiEmbed konnte mit der Multi-Task-Strategie eine bessere Leistung im Vergleich zum Einzeltraining erzielen. Withnell et al. konstruierten ein, auf einem VAE basierendes interpretierbares Deep-Learning-Modell zur Krebsklassifizierung unter Verwendung hochdimensionaler Omics-Daten [25]. Die Autoren waren in der Lage, die überwachte Aufgabe des Netzwerks zu erklären sowie die wichtigsten Gene und latenten Dimensionen für eine Vorhersage aufzudecken. Die Erklärungen der überwachte Aufgaben wurden anhand biologischer Annotationen sowie Literatur validiert. Way und Greene schlugen einen VAE-Modell (Tybalt) vor, um biologisch relevante latente Merkmale aus Krebsexpressionsdaten zu extrahieren [28]. Der VAE konnte verschiedene Merkmale wie das Geschlecht des Patienten sowie Krebs-Subtypen identifizieren. Titus et al. erweiterten das von Way und Greene vorgeschlagenen VAE-Modell Tybalt und wandten es auf DNA-Methylierungsdaten an [91]. Peng et al. schlugen die Methode DeepCDNet zur Ähnlichkeitsmessung von Genexpressionsprofilen vor, die auf einem SNN basiert [29]. Die Ergebnisse zeigten, dass DeepCDNet eine bessere Leistung als die Gene Set Enrichment Analyse bei der Messung der Ähnlichkeit zwischen Genexpressionsprofilen erzielt. Jeon et al. entwickelten ReSimNet, bei dem es sich um ein SNN handelt, das die Ähnlichkeit der Transkriptionsreaktion von Medikamenten vorhersagt [30]. Dincer et al. stellten einen Ansatz zum Erlernen von robusten Embeddings aus Genexpressionsdaten vor [92]. Der Adversarial Deconfounding Au-

toencoder (AD-AE) besteht aus einem AE zum Generieren des Embeddings und einem Adversary Netzwerk, das die Variationsquellen des Embeddings vorhersagt. Der AD-AE versucht, die Variationsquellen von wahren Signalen zu trennen, um biologisch informative Einbettungen zu generieren. Es konnte gezeigt werden, dass der AD-AE Einbettungen generieren kann, die im ursprünglichen Raum vorhandene biologische Signale konservieren und keine Informationen aus Variationsquellen codieren.

Aufgrund der Vielzahl an existierenden Dimensionsreduktionsmethoden gestaltet sich die Auswahl einer geeigneten Methode zur Dimensionsreduktion oft als schwierig. Jede der hier untersuchten Dimensionsreduktionsmethoden wurde bereits erfolgreich auf Genexpressionsdatensätze angewandt. Allerdings gibt es verhältnismäßig wenig Studien, welche die Qualität verschiedener Verfahren bei der Reduktion von Genexpressionsdaten vergleichen. Zudem haben die existierenden vergleichenden Studien wenige bis gar keine DL-Methoden zur Dimensionsreduktion mit einbezogen.
Smith et al. führten in einer umfassenden Analyse 24 binäre und Mehrklassenklassifizierungsaufgaben sowie 26 Überlebensanalyseaufgaben unter Verwendung verschiedener Genuntergruppen, Normalisierungsmethoden, Dimensionsreduktionsverfahren und überwachter ML-Algorithmen durch [12]. Die untersuchten Dimensionsreduktionsmethoden waren die PCA, der SDAE und der VAE. Die Dimensionsreduktionsmethoden wurden untereinander anhand einer prädiktiven Aufgabe verglichen. Die Verwendung der Repräsentationslerntechniken hat zu keiner konsistenten Verbesserungen der Out-of-Sample-Leistung über die Datensätze hinweg geführt. Bartenhagen et al. führten eine vergleichende Untersuchung von Dimensionsreduktionsverfahren in Bezug auf die Visualisierung von Microarray-Daten durch, um die Leistung von PCA, Kernel PCA, Locally Linear Embedding, Isomap, Diffusion Maps, Laplaceian Eigenmaps und Maximum Variance Unfolding [93] zu bewerten. Eine Vergleichsanalyse von PCA und Partial Least Square zur Merkmalsextraktion von Microarray-Daten wurde von Arowolo et al. durchgeführt [94]. Die Merkmalsextraktionsleistung wurde anhand eines SVM-Klassifikators evaluiert. Es konnte gezeigt werden, dass der Partial Least Square-Algorithmus im Vergleich zu dem PCA-Algorithmus eine bessere Qualität von $\approx$ 95,2 % Genauigkeit bietet. Franco et al. verglichen als einzige mehr als zwei AE-Typen (klassischer AE, Denoising-AE, Sparse-AE und VAE) [50]. Sie verwendeten die AE-Architekturen, um bereits selektierte Merkmale von verschiedenen Omics-Daten zu fusionieren und in einen gemeinsamen niedrigdimensionalen Unterraum zu transformieren. Die reduzierten Merkmale wurden verwendet, um Patienten mit ähnlichen Merkmalen und einer ähnlichen Überlebensprognose zu stratifizieren. Die Ergebnisse zeigten, dass der klassische AE und der VAE die anderen Dimensionsreduktionsverfahren in ihrer Leistung übertrafen, jedoch wurden starke Leistungsunterschiede bei den verschiedenen Datensätzen festgestellt. Dies unterstützt die Wichtigkeit dieser und weiterer Arbeiten zum Vergleich verschiedener (DL-basierter) Dimensionsreduktionsmethoden. Weitere verwandte Arbeiten können in [95] gefunden werden. Die Autoren diskutieren in diesem Review die Anwendung verschiedener AE-Modelle in der biomedizinischen Umgebung und die damit verbundenen erzielten Verbesserungen bei der Diagnose und dem Überleben von Patienten.

# 3 Implementierung und Validierung der Dimensionsreduktionsmodelle

In diesem Kapitel wird der Aufbau sowie die Implementierung des Evaluierungsworkflows präsentiert.

## 3.1 Evaluierungsworkflow

Der hier vorgestellte Evaluierungsworkflow ist ein mehrstufiges Verfahren, welches die Evaluierung von DL-Methoden zur Extraktion einer biologisch relevanten Repräsentation aus Genexpressionsdaten beschreibt. Eine schematische Abbildung des Evaluierungsworkflows ist in Abbildung 11 dargestellt.

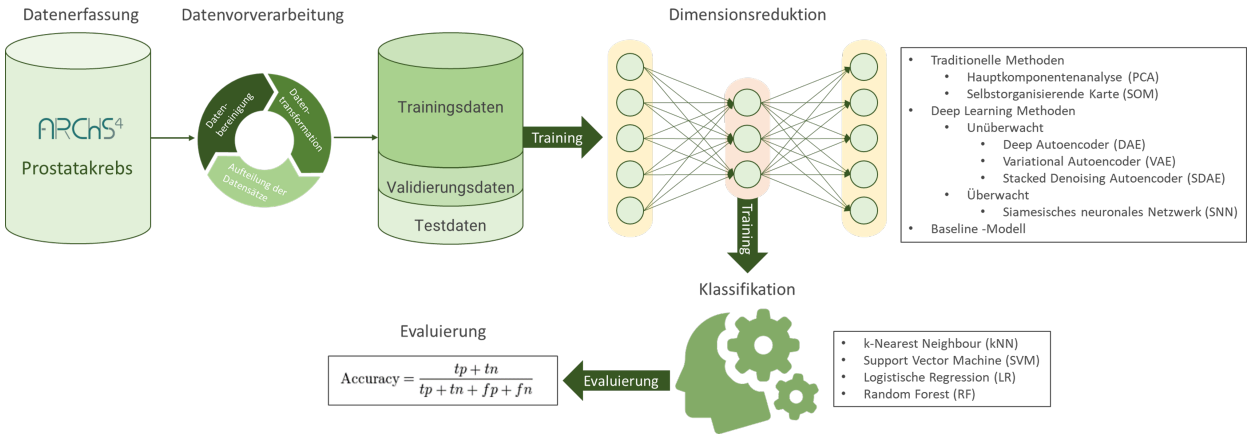



**Abbildung 11:** Schematische Darstellung des Evaluierungsworkflows. Die erfassten Daten wurden vorverarbeitet und in Trainings-, Validierungs- und Testdaten unterteilt. Die Trainingsdaten wurden verwendet, um verschiedene Dimensionsreduktionsmodelle (PCA, SOM, DAE, SDAE, VAE, SNN) zu trainieren, die in der Lage sind, eine Repräsentation zu erlernen (ein Baseline-Modell, das die Daten nicht reduziert, ist ebenfalls enthalten). Zur Evaluierung der erlernten Repräsentationen wurden verschiedene überwachte Modelle (kNN, SVM, LR, RF) trainiert. Die Klassifikationsleistung wurde mittels der Accuracy evaluiert.

Der Workflow kann in fünf Phasen unterteilt werden: **Datenerfassung**, **Datenvorverarbeitung**, **Dimensionsreduktion** zum Erlernen der Repräsentation, **Klassifikation** auf Grundlage der erlernten Repräsentation zur Evaluierung der Merkmalsextraktionsqualität der Dimensionsreduktionsmodelle und **Evaluierung** der Klassifikationsleistung.

### 3.1.1 Hardware Setup

Aufgrund der großen Dimensionalität der in dieser Arbeit verwendeten Datensätze wurden die in diesem Kapiteln beschriebenen Experimente auf dem Computing Cluster Clara der Universität Leipzig durchgeführt. Das Clara Cluster besteht aus 31 Knoten mit insgesamt 992 CPU-Kernen und

15,5 Terabyte nutzbarem Speicher. Davon sind 8 Knoten mit je 4 NVIDIA ® Tesla ® V100 32GB Grafikkarten und 23 Knoten mit je 8 NVIDIA ® GeForce ® RTX2080Ti-Grafikkarten ausgestattet. Alle Experimente mit Ausnahme der DL-Methoden wurden auf den CPU-Kernen durchgeführt. Für die DL-Methoden wurden die mit NVIDIA ® Tesla ® V100 32GB ausgestatteten Knoten verwendet. Im Folgenden wird die Implementierung der fünf Phasen des Workflows näher erläutert.

## 3.2 Datenerfassung

Zur Validierung der Dimensionsreduktionsmodelle wurden zwei öffentlich zugängliche Genexpressionsdatensätze verwendet: der ARCHS4-Datensatz [96] und der Prostatakrebs-Datensatz (GSE134051) [97].

### ARCHS4-Datensatz

ARCHS4 ist eine Web-Ressource, die bereits veröffentlichte RNA-Seq Rohdatenvon Mensch und Maus aus dem Gene Expression Omnibus (GEO) [98] und dem Sequence Read Archive (SRA) [99] mit Hilfe einer Verarbeitungspipeline neu aufbereitet hat, um einen einheitlichen Datensatz zu erzeugen [96]. ARCHS4 wird automatisch aktualisiert, sobald neu veröffentlichte Proben auf GEO/SRA verfügbar sind [96]. Die Genexpressionsdaten werden von ARCHS4 im H5-Format für den programmatischen Zugriff zur Verfügung gestellt [96]. Eine H5-Datei ist eine Datendatei, die im hierarchischen Datenformat (HDF) gespeichert ist [100]. HDF ist seit 1988 verfügbar und hat sich zu einem Standard für die wissenschaftliche Datenarchivierung und den Datenaustausch entwickelt [100]. Zum Zeitpunkt des Downloads waren 348.184 Proben mit je 35.238 Genen vom Menschen über ARCHS4 zugänglich. Zur weiteren Analyse mittels maschineller Lernmethoden wurden die Genexpressionsdaten mithilfe von R in eine kommagetrennte Textdatei konvertiert.

### Prostatakrebs-Datensatz

Der Prostatakrebs-Datensatz (GSE134051) [97] ist als TAB separierte Textdatei über die GEO-Datenbank zugänglich. Der Datensatz enthält Microarray-Daten von 164 Prostatakrebs-Patienten aus Tumorgewebe (Prostatekrebs-t) sowie tumorfreiem Gewebe (Prostatekrebs-f), die sich einer radikalen Prostatektomie unterzogen hatten und von 39 Patienten mit benigner Prostatahyperplasie (Kontrolle) [97]. Die benigne Prostatahyperplasie beschreibt eine gutartige Vergrößerung der Prostata. Über GEO werden sowohl die Roh-Genexpressionsdaten als auch die vorverarbeiteten Genexpressionsdaten des Prostatakrebs-Datensatzes veröffentlicht. In dieser Arbeit wurde mit dem bereits vorverarbeiteten Prostatakrebs-Datensatz gearbeitet. Der Prostatakrebs-Datensatz (GSE134051) umfasst 255 Proben von Patienten mit jeweils 147.044 Merkmalen. Die 255 Stichproben des Prostatakrebs-Datensatzes lassen sich in 164 kranke (Prostatakrebs-t) und 91 gesunde Proben (Kontrolle, Prostatakrebs-f) unterteilen.

## 3.3 Datenvorverarbeitung und -beschreibung

In diesem Abschnitt wird die Vorverarbeitung der erfassten Genexpressionsrohdaten beschrieben. Die Vorverarbeitung lässt sich in 3 Abschnitte unterteilen. Zuerst wurden die Daten bereinigt, anschließend transformiert und zuletzt für die weitere Analyse in Trainings-, Validierungs- und Testdatensatz aufgeteilt. Die Datentransformation wurde für den bereits vorverarbeiteten Prostatakrebs-Datensatz nicht berücksichtigt. Die verschiedenen Abschnitte der Datenvorverarbeitung wurden mit den Programmiersprachen Python und R implementiert.

### 3.3.1 Datenbereinigung

#### ARCHS4-Datensatz

Die Datenbereinigung des ARCHS4-Datensatzes wurde mit der Programmiersprache Python implementiert. Der Datenbereinigungsworkflow des ARCHS4-Datensatzes ist in Abbildung 12 dargestellt.

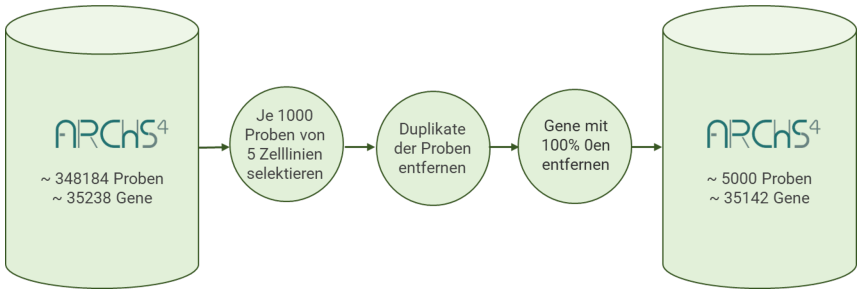



**Abbildung 12:** Datenbereinigungsworkflow des ARCHS4-Datensatzes. Die erfassten Daten wurden zunächst mithilfe eines von ARCHS4 bereitgestellten Tools zur GEO-Metadaten Textsuche auf 1000 Proben pro Zelllinie reduziert. Es wurden 5 Zelllinien für die spätere Analyse ausgewählt. Anschließend wurden doppelte Stichproben und Gene, die in allen verbleibenden Proben eine Expression von null zeigten, aus dem Datensatz entfernt. Dies führte zu den Expressionsprofilen von 35.143 Genen und 5000 Proben.

Die Qualität der Merkmalsextraktion durch Dimensionsreduktionsverfahren wurde auf Basis des ARCHS4-Datensatzes anhand der Klassifizierung von Zelllinien bewertet. Neben den Genexpressionsdaten enthält die H5-Datei der ARCHS4-Datenbank Metadaten, welche von GEO abgerufen werden [96]. Die Beschreibung der Zelllinien in den Metadaten der ARCHS4-Datenbank ist stark heterogen. ARCHS4 stellt ein Tool zur Verfügung, mit dessen Hilfe eine Textsuche der GEO-Metadaten durchgeführt werden kann, um Proben mit übereinstimmenden Metadaten in einem automatisch generierten R-Skript bereitzustellen [96]. Dieses Tool wurde verwendet, um je 1000 verschiedene Proben von 5 unterschiedlichen Zelllinien aus der ursprünglichen Expressionsmatrix mit 348184 Proben zu selektieren. Anschließend wurden alle Duplikate von Proben und alle Gene, die über keine der verbleibenden Proben eine Expression größer null zeigten, aus dem Datensatz entfernt. Da die verbleibenden Expressionsdaten keine fehlenden Werte enthielten und jeder Eintrag eine positive ganze Zahl darstellte, war die Datenbereinigung damit abgeschlossen. Dies führte zu den Expressionsprofilen von 35.143 Genen und 5000 Proben.

#### Prostatakrebs-Datensatz

Die Datenbereinigung des Prostatakrebs-Datensatzes wurde mit der Programmiersprache Python implementiert. Der Datenbereinigungsworkflow des Prostatakrebs-Datensatzes ist in Abbildung 13 dargestellt.

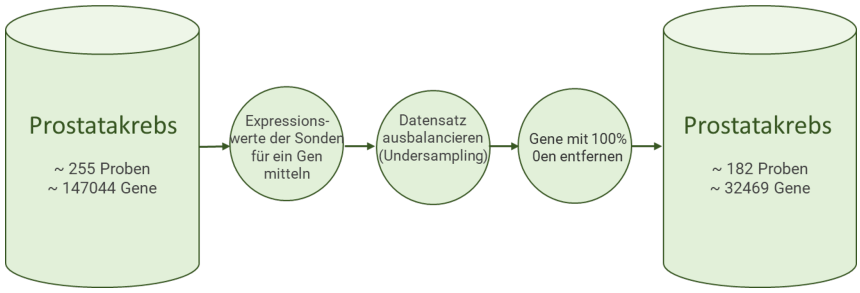



**Abbildung 13:** Datenbereinigungsworkflow des Prostatakrebs-Datensatzes. Die Expressionswerte der Sonden für ein Gen wurden gemittelt und die daraus resultierenden Expressionsprofile wurden mittels Undersampling ausbalanciert. Anschließend wurden Gene, die in allen verbleibenden Proben eine Expression von null zeigten, aus dem Datensatz entfernt. Dies führte zu den Expressionsprofilen von 32.469 Genen und 182 Proben.

Der Prostatakrebs-Datensatz (GSE134051) [97] wurde im Vergleich zu dem ARCHS4-Datensatz mit Hilfe von Microarrays generiert. Deshalb unterscheidet sich die Dimensionalität des Prostatakrebs-Datensatzes stark von der des ARCHS4-Datensatzes. Zur Vergleichbarkeit beider Datensätze wurden die Expressionswerte der Sonden für ein Gen des Prostatakrebs-Datensatz gemittelt. Da das Trainieren von Klassifikatoren mit einem unausgewogenen Trainingsdatensatz zu einem in Richtung der häufigeren Klasse voreingenommen Klassifikator führen kann [101], wurden die daraus resultierenden Expressionsprofile mittels Undersampling ausgewogen. Undersampling löscht zufällig Stichproben aus der Mehrheitsklasse, sodass die Menge an Stichproben der Mehrheitsklasse der Menge an Stichproben der Minderheitsklasse entspricht. Anschließend wurden alle Gene, die über keine der verbleibenden Proben eine Expression größer null zeigten, aus dem Datensatz entfernt. Da die verbleibenden Expressionsdaten keine fehlenden Werte enthielten und jeder Eintrag eine positive Zahl darstellte, war die Datenbereinigung damit abgeschlossen. Dies führte zu den Expressionsprofilen von 32.469 Genen und 182 Proben.

### 3.3.2 Datentransformation

#### ARCHS4-Datensatz

Die Datentransformation wurde mit der Programmiersprache R implementiert. Der Datentransformationsworkflow ist in Abbildung 14 dargestellt.
Zunächst wurde das Expressionsprofil des ARCHS4-Datensatzes $Log_2$-transformiert, um die Varianzen zwischen niedrig und hoch-exprimierten Genen anzupassen. Da Genexpressionsdaten aufgrund logistischer oder praktischer Beschränkungen im Normalfall in Versuchsreihen erstellt werden, wurde anschließend eine Quantil-Normalisierung pro Zelllinie durchgeführt, um die Vergleichbarkeit der

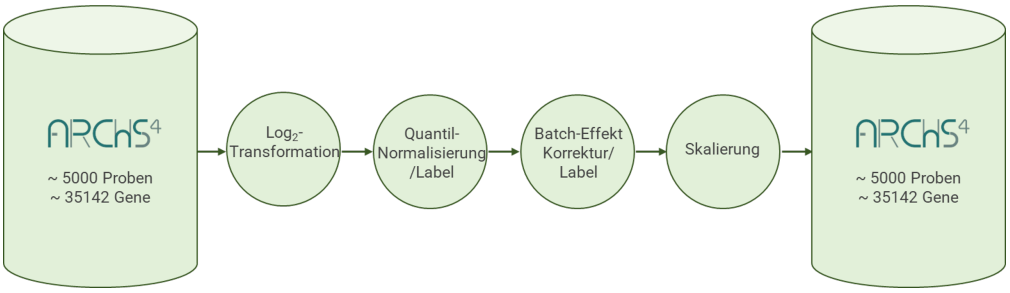



**Abbildung 14:** Datentransformationsworkflow des ARCHS4-Datensatzes. Zuerst wurden die bereinigten Genexpressionsdaten $Log_2$-transformiert, anschließend wurde eine Quantil-Normalisierung sowie eine Batch-Effekt-Korrektur pro Zelllinie durchgeführt. Zum Schluss wurden die Daten für die anschließende Weiterverarbeitung mittels maschineller Lernmethoden mit dem StandardScaler von scikit-learn skaliert.

Proben zwischen den Versuchsreihen zu erreichen [102]. Des Weiteren wurde eine Batch-Effekt-Korrektur durchgeführt. Batch-Effekte sind technische Artefakte, die nicht mit der zu untersuchenden Probe zusammenhängen, sondern mit unabhängigen Faktoren wie Laborbedingungen, Versuchszeit, Reagenzienchargen und/oder Unterschieden im Laborpersonal [103]. Die Batch-Effekt-Korrektur wurde mit der Funktion ComBat [104] aus dem R/Bioconductor-Paket sva [105] durchgeführt. Zuletzt wurden die Daten für die anschließende Weiterverarbeitung mittels maschineller Lernmethoden mit dem StandardScaler von scikit-learn [106] skaliert.

### 3.3.3 Aufteilung der Datensätze

Die Aufteilung der Datensätze in Trainings-, Validierungs- und Testdatensatz erfolgte zur Validierung der Ergebnisse mittels verschachtelter 5-facher Kreuzvalidierung. Diese dient dazu, eine robuste Schätzung der erwarteten Modellqualität für einen unabhängigen Datensatz zu erzeugen sowie die optimalen Hyperparameter für die untersuchten Modelle zu identifizieren. Die Hyperparameteroptimierung wird in Unterabschnitt 3.4.1 näher erläutert. Eine schematische Darstellung der verschachtelten Kreuzvalidierung, eingebettet in den Evaluierungsworkflow, ist in Abbildung 15 dargestellt.
Die verschachtelte Kreuzvalidierung besteht aus einer äußeren und einer inneren Schleife, um die Daten zu durchlaufen. In der inneren Schleife wird ein optimaler Satz an Hyperparameterwerten ausgewählt und in der äußeren Schleife wird der optimale Satz an Hyperparameterwerten verwendet, um die Qualität des Modells für einen unabhängigen Datensatz zu schätzen. Aufgrund der großen Dimensionalität der hier verwendeten Datensätze wurde die Hyperparameteroptimierung sowie die äußere Schleife der verschachtelten Kreuzvalidierung im Slurm Workload Manager implementiert. Um dafür Sorgen zu tragen, dass bei der Aufteilung der Datensätze jede der 5 Teilmengen eine annähernd gleiche Verteilung besitzt, wurde die stratifizierte Kreuzvalidierung des scikit-learn-Moduls [106] verwendet.
In der äußeren Schleife wurden die Daten gleichmäßig in 5 Teilmengen unterteilt. 5-1 Teilmengen wurden zum Trainieren (Trainingsdatensatz) und die verbleibende Teilmenge zum Testen (Testdatensatz) der Modelle verwendet. Dieser Schritt wurde 5-mal wiederholt, sodass jede der 5 Teilmen-

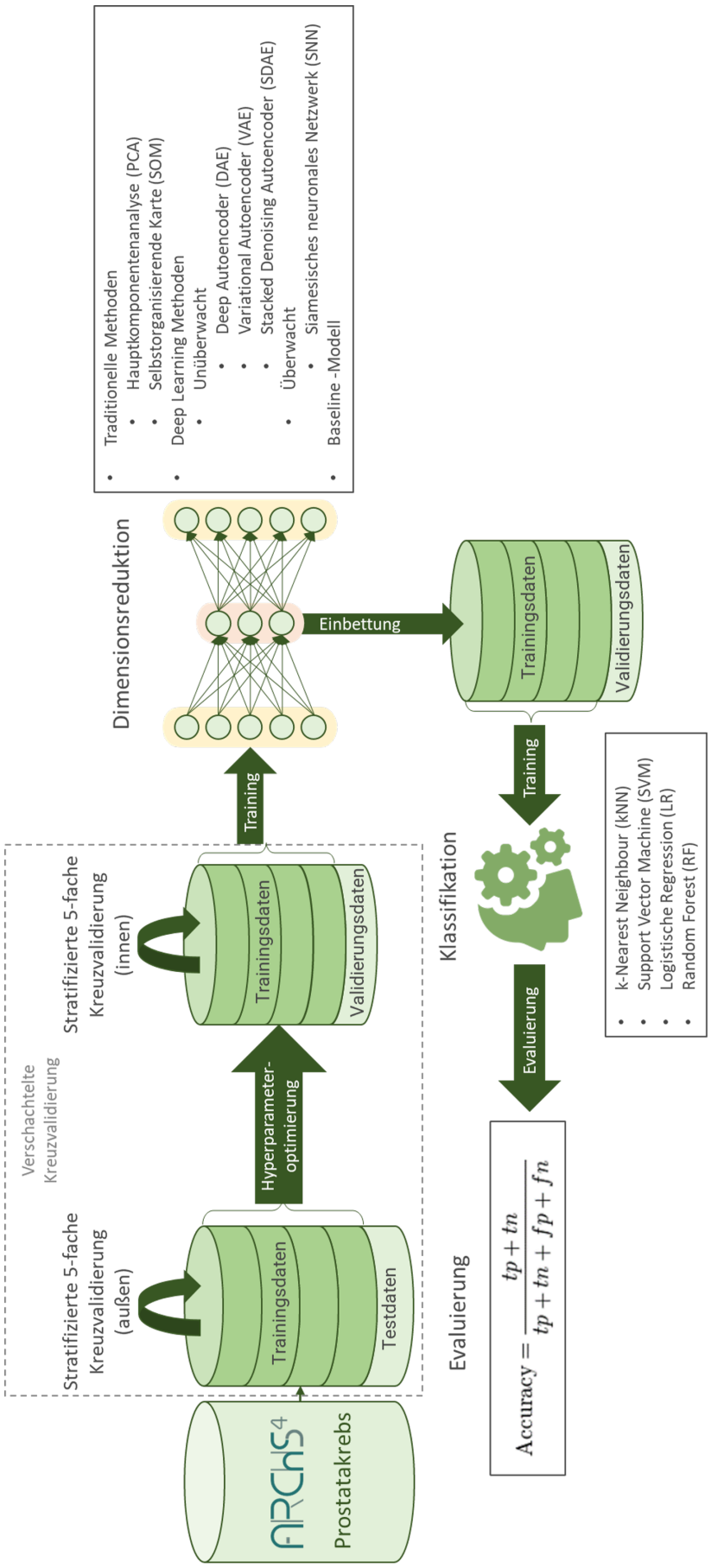



**Abbildung 15:** Aufteilung der Datensätze in Trainings-, Validierungs- und Testdatensatz. Die erfassten und transformierten Daten wurden in der äußeren Schleife der verschachtelten Kreuzvalidierung 4:1 in Trainings- und Testdaten unterteilt. Für jeden Hyperparametersatz wurden die Trainingsdaten in der inneren Schleife erneut 4:1 in Trainings und Validierungsdaten aufgeteilt. Die (inneren) Trainingsdaten wurden verwendet, um Dimensionsreduktionsmodelle (PCA, SOM, DAE, SDAE, VAE, SNN) zu trainieren (ein Baseline-Modell, das die Daten nicht reduziert, ist ebenfalls enthalten). Auf Basis der trainierten Dimensionsreduktionsmodelle wurden Embeddings der (inneren) Trainings- und Validierungsdaten generiert. Zur Evaluierung der Merkmalsextraktionsqualität wurden die generierten Embeddings der (inneren) Trainingsdaten verwendet, um verschiedene überwachte Modelle (kNN, SVM, LR, RF) zu trainieren. Die Klassifikationsleistung der überwachten Modelle wurde anhand der erlernten Repräsentationen der (inneren) Validierungsdaten anhand der Accuracy evaluiert. Das Modell mit der höchsten mittleren Vorhersagegenauigkeit über die vier Klassifikatoren wurde mit den (äußeren) Trainingsdaten erneut trainiert. Die Modellqualität wurde anhand der (äußeren) Testdaten evaluiert. Die sich daraus ergebenden Accuracy-Werte wurden über alle Wiederholungen der äußeren Schleife gemittelt, woraus sich die endgültige geschätzte Modellqualität ergab.

gen einmal zum Testen der Modelle verwendet wurde. In der inneren Schleife wurde der Trainingsdatensatz der äußeren Schleife erneut in 5 Teilmengen unterteilt. Die 5-1 Teilmengen der inneren Schleife wurden zum Trainieren (Trainingsdatensatz) der Modelle und die verbleibende Teilmenge wurde zum Validieren (Validierungsdatensatz) der Modellqualität für jeden Hyperparametersatz verwendet. Dieses Verfahren wurde 5-mal wiederholt und die endgültige Modellqualität gemittelt. Der Hyperparametersatz mit der besten Qualität wurde, wie in Unterabschnitt 3.4.1 beschrieben, ausgewählt und das Modell mit dem optimalen Hyperparametersatz wurde mit dem kompletten, in der inneren Schleife verwendeten Datensatz (Trainings- und Validierungsdatensatz) erneut trainiert. Die Modellqualität wurde anhand der Testdaten der äußeren Schleife evaluiert. Die sich daraus ergebenden Accuracy-Werte wurden über alle Wiederholungen der äußeren Schleife gemittelt, und die resultierende gemittelte Accuracy wurde zum Vergleich der Modelle herangezogen.
Die verschachtelte Kreuzvalidierung wurde gewählt, da sie resistent gegenüber dem Hyperparameter-Overfitting ist [12]. Dies liegt in der Tatsache begründet, dass ein Teil der zur Bewertung der Modellqualität verwendeten Daten bei der Identifizierung des optimalen Hyperparametersets ausgeschlossen wird [12]. Ein Nachteil der verschachtelten Kreuzvalidierung ist eine erhöhte Laufzeit, welche durch die Anzahl an zu trainierenden individuellen Modellen bedingt ist. Unter Verwendung der verschachtelten 5-fachen Kreuzvalidierung müssen bei der Untersuchung von $n$ Hyperparametersätzen $5(5n + 1)$ individuelle Modelle trainiert werden.

### 3.3.4 Datenbeschreibung

Die Expressionsmatrizen beider Datensätze sind wie folgt aufgebaut: Die erste Spalte der Expressionsmatrix entspricht der Proben-ID und die erste Zeile entspricht der Gen-ID. Jede weitere Spalte der Expressionsmatrix steht für ein Gen und jede Zeile für eine Probe. Dementsprechend steht jede weitere Zelle der Matrix für das Expressionsniveau des entsprechenden Gens in der entsprechenden Probe [107]. Diese Werte stellen die Merkmale dar, die es den ML-Algorithmen ermöglichen, die Klassifizierung zu erlernen. Die Datensätze unterscheiden sich in drei wesentlichen Aspekten. Zum einen ist die Menge der vorhandenen Stichproben in den Datensätzen stark unterschiedlich. Zum anderen handelt es sich bei dem ARCHS4-Datensatz um Microarray-Daten, während der Prostatakrebs-Datensatz aus RNA-Seq-Experimenten stammt. Abschließend unterscheiden sich die Datensätze in der gegebenen Klassenbeschreibung, was zur Folge hat, dass sich die Datensätze auch in der nachfolgenden prädiktiven Aufgaben unterscheiden.

#### ARCHS4-Datensatz

Im ARCHS4-Datensatz lässt sich jede Probe einer der 5 spezifischen Zelllinien zuordnen, wobei jede Zelllinie einem anderen Gewebe angehört. Die Beschreibung der Zelllinien mit den zugehörigen Geweben sind in Tabelle 2 aufgeführt.
Für jede Zelllinie wurden 1000 Proben selektiert. Dadurch wurden die Repräsentanten jeder Zelllinie ausgewogen.

**Tabelle 2:** Beschreibung der Label des ARCHS4-Datensatzes.

| Label | Gewebe | Zelllinie | Beschreibung der Zelllinie |
|---|---|---|---|
| **Breast_MCF7** | Brustgewebe | MCF7 | Brustkrebs-Zelllinie |
| **Cervix_HELA** | Gebärmutterhalsgewebe | HELA | Gebärmutterhalskrebs-Zelllinie |
| **Colon_HCT116** | Dickdarmgewebe | HCT116 | Dickdarmkrebs-Zelllinie |
| **Kidney_HEK293** | Nierengewebe | HEK293 | embryonale Nieren-Zelllinie |
| **Lung_A549** | Lungengewebe | A549 | Lungenkrebs-Zelllinie |

#### Prostatakrebs-Datensatz

Die Proben des Prostatakrebs-Datensatzes lassen sich in krank und gesund (Kontrolle) stratifizieren, wobei die Proben der Prostatakrebs-Patienten sowohl aus Tumorgewebe als auch aus tumorfreiem Gewebe extrahiert wurden. Die Beschreibung der Proben mit den zugehörigen Labels sind in Tabelle 3 aufgeführt. Die Repräsentanten jeder Klasse sind ausgewogen.

**Tabelle 3:** Beschreibung der Label des Prostatakrebs-Datensatzes.

| Label | Proben |
|---|---|
| **Krank** | Proben von Prostatakrebs-Patienten aus Tumorgewebe (Prostatakrebs-t) |
| **Gesund** | Proben von Prostatakrebs-Patienten aus tumorfreiem Gewebe und (Prostatakrebs-f) |
| **Gesund** | Proben von Patienten mit benigner Prostatahyperplasie (Kontrolle) |

## 3.4 Dimensionsreduktionsmodelle zum Erlernen der Repräsentation

Die vorverarbeiteten Genexpressionsdatensätze wurden zum Trainieren von sechs verschiedenen Dimensionsreduktionsmodellen verwendet. Ziel ist es, eine geeignete Repräsentation aus Genexpressionsprofilen zur Verbesserung der Vorhersagequalität der nachfolgenden prädiktiven Aufgabe zu generieren. Die Modelle lassen sich in DL-Methoden und traditionelle Dimensionsreduktionsmethoden unterteilen. Die DL-Methoden wurden mit den traditionellen Methoden quantitativ verglichen, um zu untersuchen, inwieweit DL-Methoden eine Verbesserung gegenüber den traditionellen Methoden erreichen können. Bevor die verschiedenen Arten der Dimensionsreduktionsmethoden vorgestellt werden, wird die Hyperparameteroptimierung der Modelle präsentiert. Die Algorithmen wurden mit der Programmiersprache Python implementiert.

### 3.4.1 Hyperparameteroptimierung

Die Hyperparameter werden vor dem eigentlichen Training festgelegt. Dementsprechend lassen sich diese von den im Verlauf des Trainings optimierten Parametern, den Gewichten, unterscheiden. Die festgelegten Hyperparametern beeinflussen sowohl die Laufzeit als auch die Qualität des Algorithmus. Aufgrund dessen ist es empfehlenswert, den Satz an Hyperparametern, welcher die bestmöglichen Ergebnisse liefert, zu identifizieren. In dieser Arbeit wurde die Hyperparameteroptimierung

mittels Grid-Search durchgeführt. Der optimale Hyperparametersatz wurde anhand der gemittelten Klassifikationsqualität über die vier Klassifikatoren auf Basis der erlernten Repräsentation der Dimensionsreduktionsmodelle gewählt.

Häufig verwendete Hyperparameter für ANNs sind die Anzahl der Trainingsepochen, die Größe der Trainingsbatches, der verwendete Optimierungsalgorithmus, die Lernrate, die Initialisierungen der Gewichte, die verwendeten Aktivierungsfunktionen, die Drop-out-Rate, die Anzahl der verdeckten Schichten und die Anzahl der Neuronen pro Schicht.

Die untersuchten Ausprägungen des Hyperparameterraums für die Hyperparameteroptimierung der **DL-Modelle** sind in Tabelle 4 aufgeführt. Modelle, die für bestimmte Parameter nicht optimiert wurden, sind entsprechend in der Spalte „Ausnahmen“ aufgeführt.

**Tabelle 4:** Hyperparameter-Suchraum für die Deep Learning Dimensionsreduktionsmodelle. SDAE = Stacked Denoising Autoencoder; SNN = siamesisches neuronales Netz.

| Hyperparameter | Wertebereich / Auswahlmöglichkeiten | Ausnahmen |
|---|---|---|
| Anzahl der versteckten Schichten für den Encoder/Decoder | 1; 2; 3 | |
| Lernrate | 0,05; 0,01; 0,005 | |
| Drop-out-Rate | 0,0; 0,2 | nicht für SDAE und SNN |
| Anzahl der Epochen | 50; 100 | |
| Dimensionalität der Embeddings | 500; 1000; 5000 | |

Für jede Encoder-Schicht ($hidden_i$) wurde die Anzahl an Neuronen wie folgt definiert:

$$\lceil hidden_i \rceil = d - ((d - q)/(h + 1)) \cdot i, \tag{8}$$

wobei $h$ die in Tabelle 4 aufgeführte „Anzahl der verdeckten Schichten für den Encoder“ ist. Wie in Kapitel 2 eingeführt, steht $d$ für die Dimensionalität des Datensatzes und $q$ für die Dimensionalität des latenten Raumes. $i$ stellt den Index der Schicht dar. Folglich steht $hidden_i$ für die Neuronen-Anzahl der $i$-ten Encoder-Schicht. Insgesamt wurden nach Tabelle 4 108 Hyperparametersätze pro DL-Modell evaluiert (54 Hyperparametersätze für den SDAE und das SNN). Zusammen mit der in Unterabschnitt 3.3.3 beschriebenen verschachtelten Kreuzvalidierung zur Validierung der Modelle wurden pro DL-Modell 2700 individuelle Modelle (1350 individuelle Modelle für den SDAE und das SNN) trainiert.

Der Suchraum für die Hyperparameter der **traditionellen Dimensionsreduktionsmodelle** besteht lediglich aus der in Tabelle 4 aufgeführten „Dimensionalität der Repräsentation“.

### 3.4.2 Deep Learning-Methoden

Für die Implementierung der DL-Methoden wurde die Keras-Bibliothek [1] mit TensorFlow [2] background verwendet. Keras ist eine gut dokumentierte, hoch modulare Open Source DL-Bibliothek mit

[1] https://keras.io/
[2] https://www.tensorflow.org/

der scheinbar größten Auswahl an Algorithmen (Optimierungsalgorithmus, Normalisierungsmethoden und Aktivierungsfunktionen) [108]. Es wurden symmetrische AE-Architekturen in „Sanduhr"-Form mit 3, 5 oder 7 verdeckten Schichten verwendet, wobei der Decoder die gespiegelte Konfiguration des Encoder-Netzwerkes, mit Ausnahme der Aktivierungsfunktion der letzten Schicht des Decoders (siehe Unterunterabschnitt 3.4.2.1), darstellt. Die Anzahl der verdeckten Schichten sowie die Anzahl an Neuronen pro Schicht ist in Unterabschnitt 3.4.1 beschrieben. Die Größe der Repräsentationen beträgt 500, 1000 oder 5000. Jede Schicht der DL-Modelle ist vollständig mit der nächsten Schicht verbunden. Die Modelle wurden aufgrund der großen Dimensionalität 50 oder 100 Epochen mit dem Mini-Batch-Gradientenabstieg, einer Batchgröße von 10 und einer Lernrate von 0,05, 0,01 oder 0,005 trainiert. Um Overfitting zu vermeiden und die Trainingszeit zu verkürzen, wurde ein „Early Stopping"-Ansatz mit der tf.keras.EarlyStopping-Implementierung verwendet. Das Kriterium für den Trainingsstopp war keine Verbesserung in der entsprechenden Metrik der Validierungsdaten in 5 aufeinanderfolgenden Epochen. Die Implementierung der verschiedenen DL-Methoden wird in den folgenden Abschnitten näher erläutert.

#### 3.4.2.1 Deep Autoencoder

Die Implementierung des DAE basiert auf dem von Kramer vorgeschlagenen AE-Modell [109]. Ein Problem beim Trainieren von tiefen neuronalen Netzen ist das Overfitting. Um dieses Problem anzugehen, wird in DL-Modellen häufig Drop-out verwendet [110]. In einem DL-Modell bedeutet die Drop-out-Strategie, dass Neuronen während des Trainings zufällig aus dem neuronalen Netzwerk ausgeschlossen werden, wodurch verhindert wird, dass sich die Neuronen zu stark anpassen [110]. Deshalb wurde als erste verdeckte Schicht des Encoders eine Drop-out-Schicht mit einer Drop-out-Rate von 0 oder 0,2 eingebaut. In der Eingabeschicht und den verdeckten Schichten wurde die ReLU Aktivierungsfunktion und in der Ausgabeschicht wurde die Sigmoid Funktion angewendet. Die Gewichte wurden mithilfe von Zufallszahlen einer Gleichverteilung initialisiert, und der Verlust zwischen der Eingabe und der Ausgabe wurde anhand des MSE berechnet. Weitere Details zum DAE sowie ein schematisches Layout des Netzes sind in Unterabschnitt 2.3.2 gegeben. Weitere Hyperparameter wurden wie in Unterabschnitt 3.4.1 beschrieben gewählt und optimiert.

#### 3.4.2.2 Variational Autoencoder

Die Implementierung des VAE basiert auf dem von Kingma und Welling [77] vorgeschlagenen VAE-Modell. Der VAE besteht wie der DAE aus einem Encoder- und einem Decoder-Netzwerk (Abbildung 8). Der Encoder ist ein parametrisches Inferenzmodell, das die wahre posterior Verteilung der latenten Variablen mit einer Verteilung hier der Gauß-Verteilung approximiert. Im Falle der gaußschen Normalverteilung besteht die Ausgabe des Encoder-Netzes aus dem Mittelwert $\mu$ und der Standardabweichung $\sigma$. Aus dieser Verteilung wird ein latenter Vektor abgetastet und durch den generativen Decoder in eine Rekonstruktion der Eingabedaten decodiert (Reparameterisierung) [78]. Um den VAE zu trainieren, werden zwei Verluste benötigt. Der eine ist der Kullback-Leibler-Divergenzverlust, der darauf abzielt eine Verteilung des latenten Raums zu erzwingen. Der andere ist der Rekonstruktionsverlust, der analog zum DAE anhand des MSE berechnet wurde. Weitere

Details zur Implementierung des VAEs wurden aus [78] übernommen. Anders als beim DAE wurden die Gewichte aufgrund von anfänglichen Schwierigkeiten mit NaN Verlustwerten durch Nullen initialisiert. Der restliche Aufbau des VAE entspricht dem des DAE. Weitere Details zum VAE sowie ein schematisches Layout des Netzes sind in Unterunterabschnitt 2.3.2.2 gegeben.

#### 3.4.2.3 Stacked Denoising Autoencoder

Die Implementierung des SDAE basiert auf der ursprünglichen Definition nach Vincent et al. [69]. Anders als beim DAE wurde anstelle des Drop-outs die Eingabe vor dem Eintritt in den Encoder verfälscht, um die Robustheit der Merkmalsdarstellung zu verbessern (Abbildung 7). Die Verfälschung der Eingabedaten wurde durch additives gaußsches Rauschen mit einem Rauschfaktor von 0,5 erreicht. Die Verfälschung der Eingabedaten wurde nur während des Trainings vorgenommen. Neben der Beschädigung der Eingangsdaten unterscheidet sich der SDAE noch in der Art des Trainings von dem DAE. Im Originalpaper von Vincent et al. wird der SDAE anhand von Pretraining mit anschließendem Finetuning trainiert [69]. Das Finetuning wurde in dieser Arbeit nicht verwendet, da zur Evaluierung der Merkmalsextraktionsleistung eigenständige überwachte Lernalgorithmen verwendet wurden. Das Embedding, welches für nachfolgende prädiktive Aufgaben verwendet wurde, entspricht der höchsten Repräsentation des Encoders der Pretraining-Phase. Weitere Details zum SDAE sowie ein schematisches Layout des Netzes sind in Unterunterabschnitt 2.3.2.1 gegeben. Der Rest des Netzwerks ist entsprechend dem DAE aufgebaut.

#### 3.4.2.4 Siamesisches neuronales Netz

Die Implementierung des SNN basiert auf dem von Koch et al. vorgeschlagenen SNN zur One-Shot Bildklassifizierung [80]. Das SNN besteht aus zwei Eingabeschichten, jeweils gefolgt von Encoder-Teilnetzen mit gleicher Architektur und gemeinsamen Gewichten. Anfängliche Experimente zeigten bei dem SNN eine starke Tendenz zum Overfitting, weshalb nach jeder Schicht der Encoder-Netze eine Drop-out-Schicht mit einer Drop-out-Rate von 0,2 eingebaut wurde. Des Weiteren wurden beim Trainieren des SNN große Standardabweichungen innerhalb der 5 Durchgänge der Kreuzvalidierung festgestellt. Um die Variation in den Ergebnissen aufgrund der zufälligen Initialisierung der Gewichte, des Bias und des Drop-outs zu verhindern, wurde ein globaler Seed (42) für die Zufallszahlengeneratoren von Tensorflow gesetzt. Der Seed sorgt dafür, dass die von Tensorflow generierten Pseudozufallszahlen innerhalb derselben Version von Tensorflow deterministisch sind. Unter Verwendung des Prostatakrebs-Datensatzes wurde die Lernrate des SNNs auf 0.1 und 0.5 aufgrund anfänglicher geringer Änderungen des Trainingsverlustes angepasst. Die mittels der Encoder-Netze generierten Repräsentationen der Eingabepaare wurden anhand einer Abstandsfunktion verglichen. Im Gegensatz zu Koch et al. [80], die in ihrer Arbeit eine gewichtete L1-Distanz verwendeten, wurde in dieser Arbeit die Kosinusähnlichkeit als Distanzmaß verwendet, da gezeigt wurde, dass die Kosinus-Ähnlichkeit im Vergleich zu anderen Metriken für die Bestimmung der

Distanz von Genexpressionsdaten besser geeignet ist [29]. Die Kosinus-Ähnlichkeit zwischen zwei Repräsentationen $y_1$ und $y_2$ wurde nach folgender Formel [29] bestimmt:

$$D(y_1, y_2) = \frac{y_1 * y_2}{||y_1|| * ||y_2||}. \tag{9}$$

Die darauf folgende Ausgabeschicht, bestehend aus einem einzelnen Sigmoid-Neuron, bestimmt, basierend auf der Kosinus-Ähnlichkeit, die Wahrscheinlichkeit für die gleiche Klassenzugehörigkeit der einzelnen Instanzen des Eingabepaars. Die gewünschte Ausgabe für ein positives Eingabepaar ist ein Wert nahe 1 und ein Wert nahe 0 für ein negatives Eingabepaar. Daraus ergibt sich ein binäres Klassifikationsproblem, weshalb als Verlustfunktion die binäre Kreuzentropieverlust gewählt wurde. Zur Generierung der Eingabepaare wurde jeder Stichprobe des Datensatzes zufällig eine Stichprobe der gleichen Klasse (positives Eingabepaar) sowie eine Stichprobe einer anderen Klasse (negatives Eingabepaar) zugeordnet. Das Embedding, welches für nachfolgende prädiktive Aufgaben verwendet wurde, wird aus den identischen Encoder Netzen generiert. Der weitere Aufbau des SNN entspricht dem des DAE. Weitere Details zum SNN sowie ein schematisches Layout des Netzes sind in Unterunterabschnitt 2.3.2.3 gegeben.

### 3.4.3 Traditionelle Dimensionsreduktionsmethoden

Die Ergebnisse der Dimensionsreduktion der DL-Methoden wurden mit traditionellen Dimensionsreduktionsmethoden wie der PCA und der SOM verglichen. PCA ermöglicht eine lineare Dimensionsreduktion des ursprünglichen Datensatzes, um die Daten in einen kleineren Satz von Hauptkomponenten zu transformieren. Die Anzahl der Hauptkomponenten entspricht der Dimensionalität der Embeddings. Die PCA-Transformation wurde mit dem Decomposition-Modul der scikit-learn-Bibliothek [106] durchgeführt. Die SOM ist ein unbeaufsichtigtes neuronales Netz, das hochdimensionale Eingabedaten in topologischer Reihenfolge auf einem niedrigdimensionalen Gitter abbildet [57]. Die SOM wurde mit dem sklearn_som-Modul der scikit-learn-Bibliothek [106] implementiert. Weitere Informationen zu der Funktionsweise der traditionellen Dimensionsreduktionsmodelle sind in Abschnitt 2.2, Abschnitt 2.2 gegeben.

### 3.4.4 Das Baseline-Modell

Zusätzlich zu den Dimensionsreduktionsmodellen wurde ein Baseline-Modell implementiert, welches keine Dimensionsreduktion der Genexpressionsdaten durchführt. Die Daten für die nachfolgende Klassifikationsaufgabe sind im Falle des Baseline-Modells die nach Abschluss der in Abschnitt 3.3 beschriebenen Datenvorverarbeitung erhaltenen Genexpressionsprofile.

## 3.5 Evaluierung der Dimensionsreduktionsmethoden anhand der Klassifikation

Die Merkmalsextraktionsqualität der Dimensionsreduktionsmodelle wurde quantitativ durch vier Standard-Klassifikatoren (kNN, SVM, LR, RF), die auf Grundlage der erlernten Repräsentationen trainiert wurden, bewertet. Für jede Dimensionsreduktionsmethode und jeden Datensatz mit zugehöriger prädiktiver Aufgabe wurden alle vier Klassifikatoren trainiert und evaluiert. Zur Evaluierung der Merkmalsextraktionsqualität der einzelnen Dimensionsreduktionsmodelle wurden die Ergebnisse der vier Klassifikatoren gemittelt. Die daraus resultierende gemittelte Genauigkeit der Klassifikation gibt einen Hinweis darauf, inwieweit die erlernte Repräsentation für die prädiktive Aufgabe nützliche Merkmale aus den ursprünglichen Daten extrahiert hat. Alle nachfolgenden Klassifikationsalgorithmen wurden mit den entsprechenden Modulen der scikit-learn-Bibliothek [106] implementiert. Da in dieser Arbeit der Fokus auf der Generierung einer geeigneten Repräsentation für Genexpressionsdaten liegt, wurden die Hyperparameter der Klassifikatoren nicht optimiert. Für jeden Klassifikator wurden die Standardparameter der jeweiligen Implementierung verwendet.

### 3.5.1 K-Nearest Neighbour

Der kNN-Algorithmus [111] ist eine Methode zur Klassifizierung von Instanzen auf Grundlage der k nächstgelegenen Trainingsinstanz im Merkmalsraum [112]. KNN ist ein Algorithmus, der keine Annahmen über die Struktur der Daten und die Verteilung macht, was bedeutet, dass es sich um einen nichtparametrischen Algorithmus handelt. Dies ist von Vorteil, da Daten in der Realität kaum theoretischen Regeln gehorchen.

### 3.5.2 Support Vector Machine

Der von Cortes und Vapnik [113] entwickelte SVM-Algorithmus ist Kernel-basiert und ebenfalls nichtparametrisch. SVM sucht eine Hyperebene, welche den größten Abstand (Margin) zwischen den Klassen besitzt [114]. SVM-Klassifikatoren werden in einem zweistufigen Verfahren erzeugt: Zunächst werden die Eingabedaten auf einen hochdimensionalen Merkmalsraum projiziert, der durch eine Kernelfunktion beschrieben wird [115]. Anschließend sucht der Algorithmus in diesem Raum eine Hyperebene mit der größten Margin, welche die Datenklassen trennt.

### 3.5.3 Logistische Regression

Die LR [116] ist eine parametrische Methode, die für die Analyse dichotomer Antwortvariablen entwickelt wurde. Die LR sagt die Wahrscheinlichkeiten der möglichen Ausprägungen einer abhängigen Antwortvariable anhand einer Reihe von Merkmalen vorher [112]. Die multinomiale logistische Regression ist eine Erweiterung der logistischen Regression auf Mehrklassenprobleme [112].

### 3.5.4 Random Forest

Der von Leo Breiman [117] eingeführte RF-Algorithmus ist ein baumbasierter Ensemble-Machine-Learning-Ansatz [118]. Der RF-Algorithmus basiert auf einer Sammlung oder einem Ensemble von Entscheidungsbäumen. Die einzelnen Entscheidungsbäume werden unter Verwendung einer Bootstrap-Stichprobe der Eingabedaten erstellt [119]. Die Integration einer Sammlung von Entscheidungsbäumen zur Lösung desselben Problems (Ensemble-Machine-Learning) kann die Genauigkeit des Algorithmus verbessern [120]. Um neue Eingabedaten zu klassifizieren wird ein Abstimmungsszenario entwickelt. Ein häufig verwendetes Abstimmungsszenario ist die sogenannte Mehrheitsabstimmung. Dabei wird jeder nicht gekennzeichneten Stichprobe die Klasse, welche die maximale Anzahl von Stimmen aus Ensembles von Entscheidungsbäumen besitzt, zugewiesen [120].

## 3.6 Evaluierung der Klassifikationsleistung

Zur Evaluierung der Klassifikationsleistung wurde die Accuracy verwendet. Die Accuracy wird in der wissenschaftlichen Praxis häufig zur Bewertung der Generalisierungsfähigkeit von Klassifikatoren verwendet [121].

Ausgehend von der Zwei-Klassen-Konfusionsmatrix

| | | vorhergesagter Wert | | |
|---|---|---|---|---|
| | | Positiv | Negativ | Gesamt |
| wahrer Wert | Positiv | wahr positiv (TP) | falsch negativ (FN) | P |
| | Negativ | falsch positiv (FP) | wahr negativ (TN) | N |
| | Gesamt | p' | n' | |

ist die Accuracy definiert als [121]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{10}$$

Die Accuracy gibt an, wie viele Stichproben das Modell für den gesamten Datensatz korrekt vorhersagt hat.

## 3.7 Visualisierung der Embeddings mittels t-distributed Stochastic Neighbor Embedding

Die Ergebnisse der vorangegangenen Analysen wurden durch Visualisierung der Embedding weiter analysiert, um die Fähigkeit von Dimensionsreduktionsmethoden zur Klassentrennung zu bewerten. Die Embeddings wurden mit Hilfe von t-distributed Stochastic Neighbor Embedding (t-SNE) [122]

auf zwei Dimensionen reduziert und anschließend in einem Streudiagramm visualisiert. T-SNE basiert auf Stochastic Neighbor Embedding (SNE) [123], jedoch wird anstelle einer Gauß-Verteilung eine Student-t-Verteilung verwendet, um die Tendenz von SNE, Punkte in der Mitte der Karte zusammenzudrängen, zu reduzieren [122]. T-SNE erzeugt eine zweidimensionale Visualisierung der Embeddings, die es ermöglicht, Ähnlichkeiten zwischen Proben durch die relative Lage der abgebildeten Punkte zu identifizieren. T-SNE wurde mit einer Perplexität von 30 und 1000 Iterationen unter Verwendung der scikit-learn-Bibliothek [106] implementiert.

# 4 Evaluierung der Dimensionsreduktionsmodelle

In diesem Kapitel werden die Ergebnisse des in Abschnitt 3.1 beschriebenen Evaluierungsworkflows (siehe Abbildung 11) präsentiert und diskutiert. Um den durch hochdimensionale Daten verursachten „Fluch der Dimensionalität“ so weit wie möglich zu überwinden und die Störung durch irrelevante Merkmale zu beseitigen, wurden Dimensionsreduktionsmethoden verwendet. Die Dimensionsreduktion sorgt durch Vermeidung von Overfitting für die Verbesserung der nachfolgenden Klassifizierung. In letzter Zeit wurden DL-Methoden vermehrt zur Identifikation von nützlichen Gen- oder Transkriptkombinationen verwendet [21, 22, 23, 24, 6, 25, 26, 27, 28, 29, 30]. In dieser Arbeit wurden Genexpressionsdatensätze verwendet, um die Qualität von vier verschiedenen DL-Dimensionsreduktions-Modellen (DAE, SDAE, VAE, SNN) mit den überwachten Klassifizierungsalgorithmen kNN, SVM, LR und RF zu bewerten. Darauf aufbauend wird ein Vergleich zwischen der Merkmalsextraktionsqualität der DL-Dimensionsreduktions-Modelle untereinander, mit traditionellen Dimensionsreduktionsmethoden-Modellen (PCA, SOM) sowie mit einem Baseline-Modell, welches die Daten nicht reduziert, durchgeführt. Zur Validierung der Qualität der Dimensionsreduktionsmodelle wurden zwei öffentlich zugängliche Genexpressionsdatensätze der ARCHS4- und der Prostatakrebs-Datensatz verwendet. Der ARCHS4-Datensatz umfasst 5000 Proben und 35.143 Genen und wurde aus RNA-Seq-Experimenten generiert. Der Prostatakrebs-Datensatz hingegen besteht aus lediglich 182 Proben und 32.469 Genen und wurde durch Microarrays erzeugt. Die in dieser Arbeit durchgeführte maximale Dimensionsreduktion des ARCHS4- sowie des Prostatakrebs-Datensatzes beträgt über 98 %. Daraus ergibt sich eine Speicherreduktion von über 99 % im Vergleich zu den Originaldaten. Im Folgenden werden die Ergebnisse des ARCHS4-Datensatzes (Unterabschnitt 4.1.1) und die Ergebnisse des Prostatakrebs-Datensatzes (Unterabschnitt 4.2.1) präsentiert, jeweils separat analysiert und abschließend gemeinsam ausgewertet (Abschnitt 4.3). Um die Evaluation überschaubar zu halten, werden die einzelnen Resultate der Datensätze zunächst getrennt verglichen. Für jeden Datensatz wird im Folgenden eine Hyperparameteranalyse, eine quantitative Evaluierung der Dimensionsreduktionsmethoden anhand der Klassifikator-Genauigkeit, eine semi-quantitative Bewertung der Fähigkeit von Dimensionsreduktionsmethoden zur Klassentrennung mit t-SNE und abschließend eine quantitative Evaluierung der Klassifikatoren durchgeführt. In den einzelnen Unterabschnitten werden folgende Fragen beantwortet:

- **Hyperparameteranalyse:** Was sind die optimalen Hyperparameter für die verschiedenen DL-Dimensionsreduktionsmodelle? Welche Hyperparameter haben großen Einfluss auf die Qualität der Dimensionsreduktionsmodelle? Lässt sich die intrinsische Dimensionalität der Datensätze identifizieren?
- **Quantitative Evaluierung der Dimensionsreduktionsmethoden anhand der Klassifikator-Genauigkeit:** Sind die Dimensionsreduktionsmodelle in der Lage, die Vorhersagegenauigkeit der nachfolgenden Klassifikations-Aufgabe zu verbessern, indem sie eine geeignete latente Repräsentation lernen? Inwieweit unterscheiden sich die Ergebnisse der linearen/nichtlinearen Dimensionsreduktionsmodelle und der traditionellen/DL-Dimensionsreduktionsmodelle? Gibt es ein optimales Dimensionsreduktionsmodell, dessen Leistung sich über beide untersuchten Datensätze von der Leistung der anderen Dimensionsreduktionsmodelle abhebt?

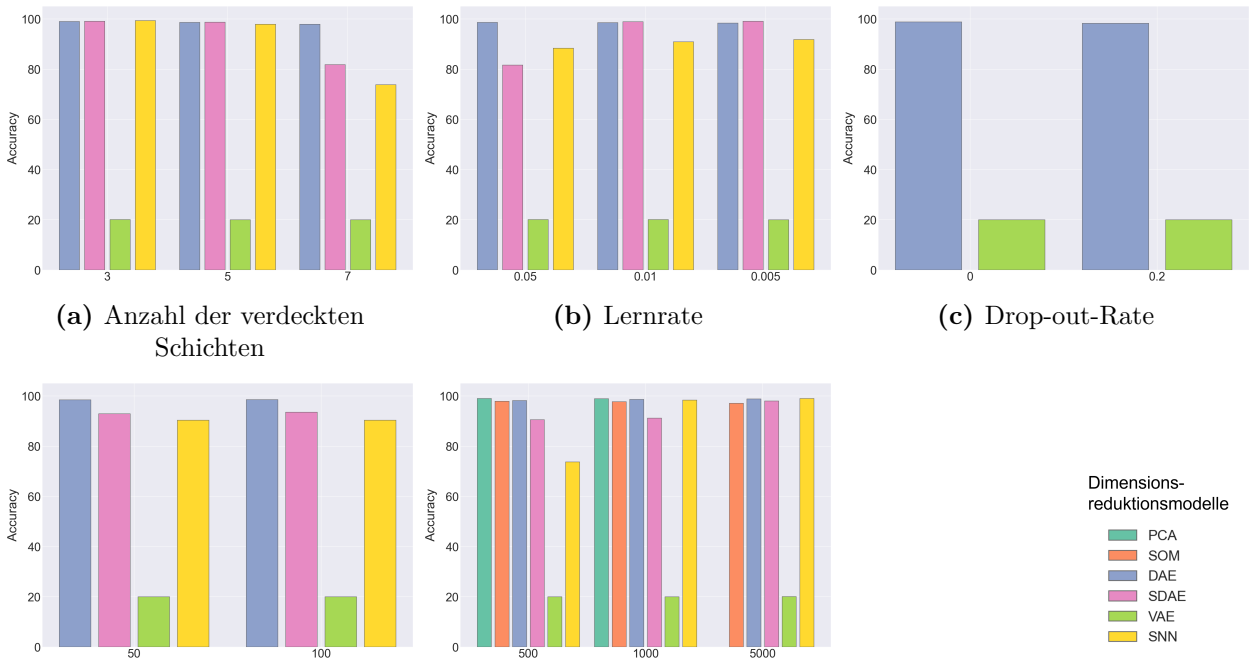
- **Semi-quantitative Bewertung der Fähigkeit von Dimensionsreduktionsmethoden zur Klassentrennung mit t-SNE:** Ist der t-SNE-Algorithmus in der Lage, die von den Dimensionsreduktionsmodellen generierten Merkmale in eine aussagekräftige zweidimensionale-Darstellung zu komprimieren? Sind die Dimensionsreduktionsmodelle fähig, die manuelle Klassentrennung durch t-SNE zu verbessern, indem sie eine geeignete latente Repräsentation lernen?
- **Quantitative Evaluierung der Klassifikatoren:** Gibt es Unterschiede in der Qualität der Klassifikatoren (kNN, SVM, RF und LR) bei den prädiktiven Aufgaben? Lässt sich ein optimaler Klassifikator identifizieren, der über die Vorverarbeitung durch die Dimensionsreduktionsmodelle sowie die zwei untersuchten Datensätze mit verschiedenen Stichprobengrößen und unterschiedlicher prädiktiver Aufgabe die beste mediane Accuracy erreicht?

## 4.1 ARCHS4-Datensatz

In diesem Abschnitt werden die Ergebnisse der auf dem ARCHS4-Datensatz beruhenden Experimente vorgestellt und diskutiert.

### 4.1.1 Hyperparameteranalyse

Die Ergebnisse der Hyperparameteroptimierung wurden für alle Dimensionsreduktionsmethoden in separaten Dateien gespeichert. Abbildung 16 fasst die Leistung der Dimensionsreduktionsmodelle über alle Hyperparameter-Permutationen hinweg zusammen. Die Accuracy für die untersuchten Werte der einzelnen Hyperparameter setzt sich aus der mittleren Accuracy über alle Klassifikatoren und über alle Hyperparametersets mit dem spezifischen, zu untersuchenden Hyperparameter-Wert zusammen. Zu beachten ist, dass nicht jedes Modell in Hinblick auf alle hier dargestellten Hyperparameter optimiert wird, nähere Informationen dazu lassen sich in Unterabschnitt 3.4.1 finden. Die Gründe für den Ausschluss von Hyperparametern liegen großteils in der Art des Trainings der verschiedenen Modelle.
Ein hoher Accuracy-Wert ist ein vielversprechender Hinweis darauf, dass das jeweilige Dimensionsreduktionsmodell eine geeignete latente Repräsentation unter Verwendung des spezifischen Hyperparameter-Wertes gelernt hat. Der VAE weist als einziges Modell über alle in dieser Arbeit untersuchten Hyperparametersets eine geringe Accuracy von $\approx 20$ % auf und scheint demnach nicht in der Lage zu sein, ein geeignetes Embedding aus den ARCHS-Daten zu generieren. Die Accuracy zeigt, dass die Drop-out-Rate (Abbildung 16c) sowie die Anzahl an Epochen (Abbildung 16d) wenig Einfluss auf die Qualität der untersuchten Modelle hat. Dies lässt auf zwei Tatsachen schließen: zum einen, dass der Großteil des Trainings bereits in den ersten 50 Epochen stattfindet und zum anderen, dass der DAE sowie der VAE keine bis wenig Tendenz zum Overfitting aufweisen, da die Einführung der Drop-out-Rate keinen Effekt zeigt. Dies kann unter anderem auf den „Early Stopping“-Ansatz, welcher während des Trainings der DL-Modelle verwendet wurde, zurückgeführt werden. Der „Early Stopping“-Ansatz verhindert Overfitting durch Abbruch des Trainings bei Stagnierung der entsprechenden Metrik. Die Analyse der Anzahl der verdeckten Schichten (Abbildung 16a)




**(a)** Anzahl der verdeckten Schichten

**(b)** Lernrate

**(c)** Drop-out-Rate

**(d)** Anzahl der Epochen

**(e)** Dimensionalität des Embeddings

**Abbildung 16:** Hyperparameteranalyse auf Grundlage des ARCHS4-Datensatzes. Für jedes Dimensionsreduktionsmodell wurde eine Hyperparameteroptimierung mit Grid-Search durchgeführt. Die Diagramme a-e stellen jeweils einen zu optimierenden Hyperparameter (Anzahl der verdeckten Schichten, Lernrate, Drop-out-Rate, Anzahl der Epochen, Dimensionalität des Embeddings) dar. Der Suchraum der einzelnen Hyperparameter ist jeweils auf der x-Achse dargestellt. Die y-Achsen geben jeweils die mittlere Accuracy über die Klassifikatoren und Hyperparametersets mit dem auf der x-Achse dargestellten festgesetzten Parameter an. Die verschiedenen Dimensionsreduktionsmodelle (PCA, SOM, DAE, SDAE, VAE, SNN) sowie ein Baseline-Modell werden durch unterschiedliche Farben differenziert. PCA = Hauptkomponentenanalyse; SOM = Selbstorganisierende Karte; DAE = Deep Autoencoder; SDAE = Stacked Denoising Autoencoder; VAE = Variational Autoencoder; SNN = siamesisches neuronales Netz.

zeigt für jedes untersuchte Modell ein optimales Ergebnis bei der minimalen Konfiguration, die lediglich aus drei verdeckten Schichten besteht. Neuronale Netze, die eine zu tiefe Architektur für das zu bewältigende Problem besitzen, neigen zum Overfitting [124]. Die verwendeten neuronalen Netze sind in der Lage, die Dimensionsreduktion mit nur zwei zusätzlichen Abstraktionsebenen zu bewältigen. Der VAE weist keine Verbesserung oder Verschlechterung durch Zu- oder Abnahme der Schichten auf. Der DAE zeigt durch Zunahme der verdeckten Schichten eine geringfügige Abnahme der Accuracy. Der SDAE und das SNN weisen die stärkste Verschlechterung von $\approx$ 20 % durch Zunahme der Schichten auf. Bei 7 Schichten zeigen der SDAE und das SNN eine starke Abnahme der durchschnittlichen Accuracy. Ein möglicher Grund für die rapide Leistungsabnahme bei 7 Schichten könnte Overfitting sein [124]. Die Lernrate (Abbildung 16b) hat sowohl auf den VAE als auch auf den DAE keinen großen Einfluss. Die Leistung des SDAE sowie des SNNs steigt mit zunehmender Lernrate, wobei die Leistungszunahme zwischen SDAEs/SNNs mit einer Lernrate von 0,01 und SDAEs/SNNs mit einer Lernrate von 0,005 nicht erheblich ist. Die Dimensionalität des Embeddings (Abbildung 16e) hat auf die PCA, die SOM, den DAE sowie den VAE nur einen geringfügigen bis keinen Einfluss. Das SNN und der SDAE erzielen bessere Ergebnisse bei zunehmender Dimensionalität des Embeddings. Der VAE erreicht über keine Embeddinggröße eine

angemessene Qualität. Die PCA, die SOM sowie der DAE sind in der Lage, eine geeignete latente Repräsentation mit geringer Dimensionalität zu lernen, wohingegen das SNN und der SDAE um annähernd gleichwertige Ergebnisse zu erzielen, eine höhere Dimensionalität des Embeddings benötigen. Die intrinsische Dimensionalität des ARCHS4-Datensatzes lässt sich nicht identifizieren. Allerdings lässt sich sagen, dass eine intrinsische Dimensionalität von 500 für den Großteil der hier untersuchten Methoden, ausgenommen dem VAE, ausreichend ist.
Es fällt auf, dass sowohl PCA und SOM als auch DAE und VAE robust gegenüber Veränderungen der hier verwendeten Hyperparameter sind. Das SNN und der SDAE reagieren dagegen empfindlich (teilweise mit einer Leistungsabnahme von über 20 %) auf Veränderungen der Anzahl an verdeckten Schichten, der Lernrate und der Dimensionalität des Embeddings.

**Tabelle 5:** Optimale Hyperparametersets der untersuchten Modelle auf Grundlage des ARCHS4-Datensatzes. Für jedes Modell wurde aus der Gesamtmenge der Hyperparameter-Permutationen pro Durchgang der Kreuzvalidierung ein optimales Set an Hyperparametern anhand der gemittelten Accuracy über die Klassifikatoren ausgewählt. Dementsprechend wurden pro Modell fünf optimale Hyperparametersets identifiziert. Die Tabelle führt die Hyperparameter zusammen mit ihrem metrischen Wert (Accuracy) auf. PCA = Hauptkomponentenanalyse; SOM = Selbstorganisierende Karte; DAE = Deep Autoencoder; SDAE = Stacked Denoising Autoencoder; VAE = Variational Autoencoder; SNN = siamesisches neuronales Netz.

| Model | Äußere Schleife der verschachtelten Kreuzvalidierung | Anzahl der verdeckten Schichten | Lernrate | Drop-out-Rate | Anzahl der Epochen | Dimensionalität des Embeddings | Accuracy |
|---|---|---|---|---|---|---|---|
| PCA | 1 | - | - | - | - | 1000 | 99.25 |
| | 2 | - | - | - | - | 1000 | 99.4 |
| | 3 | - | - | - | - | 1000 | 99.42 |
| | 4 | - | - | - | - | 1000 | 99.3 |
| | 5 | - | - | - | - | 1000 | 98.93 |
| SOM | 1 | - | - | - | - | 500 | 81.35 |
| | 2 | - | - | - | - | 500 | 87.2 |
| | 3 | - | - | - | - | 500 | 79.43 |
| | 4 | - | - | - | - | 500 | 97.5 |
| | 5 | - | - | - | - | 500 | 97.6 |
| DAE | 1 | 3 | 0.05 | 0.2 | 100 | 500 | 99.02 |
| | 2 | 3 | 0.05 | 0 | 100 | 500 | 98.88 |
| | 3 | 3 | 0.05 | 0.2 | 100 | 500 | 99.28 |
| | 4 | 3 | 0.05 | 0.2 | 100 | 500 | 99.2 |
| | 5 | 3 | 0.05 | 0.2 | 100 | 500 | 99.15 |
| SDAE | 1 | 3 | 0.005 | - | 50 | 1000 | 98.83 |
| | 2 | 3 | 0.005 | - | 50 | 500 | 98.9 |
| | 3 | 3 | 0.005 | - | 50 | 500 | 98.82 |
| | 4 | 3 | 0.005 | - | 50 | 500 | 99.18 |
| | 5 | 3 | 0.005 | - | 50 | 500 | 99.02 |
| VAE | 1 | 3 | 0.05 | 0.2 | 50 | 5000 | 20.88 |
| | 2 | 5 | 0.01 | 0.2 | 100 | 5000 | 19.5 |
| | 3 | 5 | 0.01 | 0.2 | 50 | 1000 | 19.23 |
| | 4 | 7 | 0.01 | 0.2 | 50 | 1000 | 20.58 |
| | 5 | 3 | 0.05 | 0.2 | 50 | 500 | 19.08 |
| SNN | 1 | 3 | 0.05 | - | 50 | 500 | 99.45 |
| | 2 | 3 | 0.05 | - | 100 | 500 | 99.7 |
| | 3 | 3 | 0.05 | - | 100 | 1000 | 99.52 |
| | 4 | 3 | 0.05 | - | 50 | 1000 | 99.72 |
| | 5 | 3 | 0.05 | - | 50 | 1000 | 99.82 |

Für jedes Modell wurde, wie in Unterabschnitt 3.4.1 bereits erläutert, aus der Gesamtmenge der Hyperparameter-Permutationen über alle Durchläufe der äußeren Kreuzvalidierung ein optimales Set an Hyperparametern anhand der gemittelten Accuracy über die Klassifikatoren ausgewählt. Die fünf optimalen Hyperparametersets jedes Modells sowie deren gemittelte Accuracy sind in Tabelle 5 dargestellt. Die in Tabelle 5 abgebildeten optimalen Hyperparameter-Sets decken sich großteils mit den zuvor ermittelten Ergebnissen der in Abbildung 16 dargestellten Hyperparameteranalyse.

Zu beachten ist, dass bereits geringe Veränderungen in der Accuracy, die im obigen Abschnitt als nicht erheblich identifiziert wurden, die Auswahl des optimalen Hyperparametersets beeinflussen. Die Analysen in den späteren Abschnitten basieren auf den Ergebnissen der optimalen Hyperparametersets (Tabelle 5). Die vollständigen Resultate der Hyperparameteroptimierung finden sich auf dem externen Speichermedium.

### 4.1.2 Quantitative Evaluierung der Dimensionsreduktionsmethoden anhand der Klassifikator-Genauigkeit

Die Merkmalsextraktionsqualität der Dimensionsreduktionsmodelle wurde quantitativ anhand der gemittelten Klassifikator-Genauigkeit über vier verschiedene Klassifikatoren (kNN, SVM, LR, RF) evaluiert. Diese Evaluierung beruht auf der Hypothese, dass Prädiktoren, die auf niederdimensionalen Darstellungen trainiert werden, eine bessere Vorhersagequalität erzielen [12]. Eine hohe Klassifikator-Genauigkeit ist dementsprechend ein vielversprechender Hinweis darauf, dass das jeweilige Dimensionsreduktionsmodell einen geeigneten latenten Raum gelernt hat. Um die Hypothese zu überprüfen, wurde zusätzlich ein Baseline-Modell verwendet, welches die Vorhersageleistung auf dem nicht dimensionsreduzierten Datensatz bewertet. In Abbildung 17 ist die Qualität der Dimensionsreduktionsmodelle anhand der medianen Klassifikator-Genauigkeit dargestellt.
Der Median der Vorhersageleistung der Klassifikatoren auf Grundlage der durch die Dimensionsreduktionsmodelle generierten Embeddings liegt bei allen Modellen mit Ausnahme des VAEs bei über 97 %. Daraus lässt sich schließen, dass die Dimensionsreduktionsmodelle geeignete Merkmale aus dem ursprünglichen Datensatz extrahieren. Das durch den VAE generierte Embedding führt lediglich zu einer medianen Accuracy von $\approx$ 20 %. Eine zufällige Klassifizierung der Stichproben würde vergleichbare Ergebnisse wie der VAE erzielen. Dementsprechend lässt sich sagen, dass der VAE nicht in der Lage ist, eine geeignete latente Repräsentation aus den Eingabedaten zu extrahieren. Wie in Abschnitt 2.4 bereits erwähnt, existieren mehrere Publikationen, die bereits erfolgreich VAEs sowohl auf Microarray als auch auf RNA-Seq-Daten angewandt haben [26, 6, 25, 28, 12, 50]. Zur besseren Untersuchung der Gründe für das Versagen des VAEs werden die Vorverarbeitungsschritte, die verwendeten Hyperparameter, die zur Evaluation verwendete Metrik sowie die erzielten Ergebnisse von verwandten Arbeiten in Tabelle 6 zusammengefasst.
Eine Vermutung für die in dieser Arbeit generierte schlechte Modellqualität des VAEs ist der erhöhte Einfluss des „Fluchs der Dimensionalität“ und das damit verbundene Problem des Overfittings. Es ist deutlich zu erkennen, dass jede der in Tabelle 6 verglichenen Arbeiten eine deutlich höhere Stichprobenanzahl als der ARCHS4-Datensatz mit Ausnahme des Modells von Ai et al. [26] verwendet. Allerdings geht mit der erhöhten Anzahl an Stichproben auch eine entsprechende Erhöhung der Merkmalsanzahl einher. Demnach lässt sich die aufgestellte Vermutung, dass der „Fluch der Dimensionalität“ im Falle der im Rahmen dieser Arbeit untersuchten Datensätze größeren Einfluss auf die Modellqualität hat, verwerfen. Zudem führt der „Fluch der Dimensionalität“ zum Overfitting des Modells. Overfitting lässt sich als Ursache für die schlechte Modellqualität allerdings ebenfalls ausschließen, da der in dieser Arbeit präsentierte VAE selbst auf den Trainingsdaten kein geeignetes Embedding extrahieren kann. Neben der Unterscheidung anhand der Merkmalsanzahl und der Stichprobengröße lassen sich geringe Unterschiede in der Art der Datenvorverarbeitung

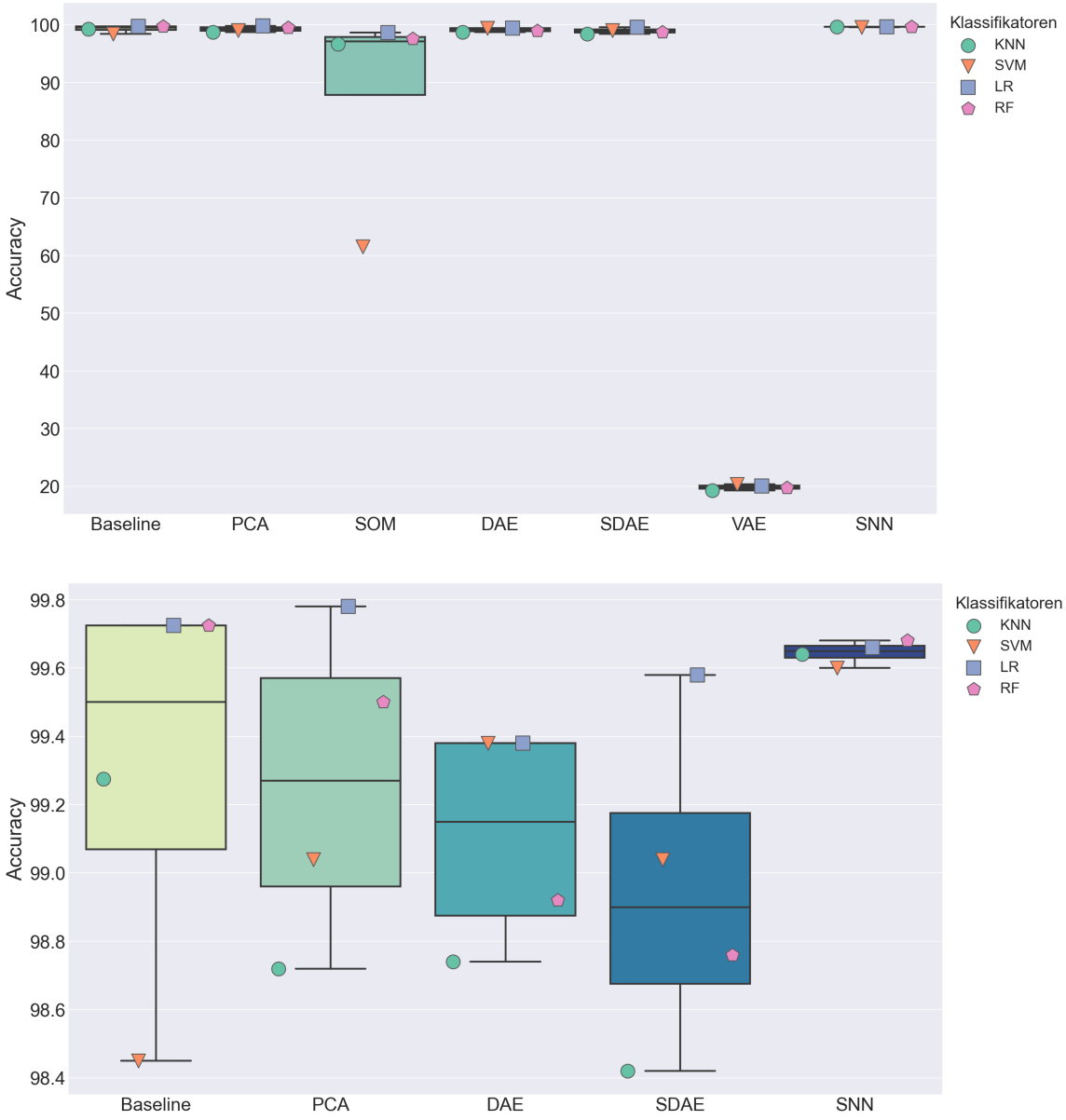



**Abbildung 17:** Quantitative Evaluierung der Dimensionsreduktionsmethoden anhand der Klassifikator-Genauigkeit auf Basis des ARCHS4-Datensatzes. Die obere Grafik stellt alle Modelle im Vergleich dar. Die untere Grafik exkludiert das VAE- sowie das SOM-Modell zur Verdeutlichung der Unterschiede in den Ergebnissen der übrigen Dimensionsreduktionsmodelle. Die Merkmalsextraktionsqualität der Dimensionsreduktionsmethoden, die auf dem ARCHS4-Datensatz basieren, wurde anhand der Zelllinien-Vorhersagegenauigkeit durch vier Klassifikatoren bewertet. Hauptkomponentenanalyse (PCA), Selbstorganisierende Karte (SOM), Deep Autoencoder (DAE), Stacked Denoising Autoencoder (SDAE), Variational Autoencoder (VAE) und ein siamesisches neuronales Netz (SNN) wurden angewendet, um die Dimension des ARCHS4-Datensatzes vor der Klassifizierung zu reduzieren. Zusätzlich wurde ein Baseline-Modell verwendet, um die Vorhersagequalität auf dem nicht dimensionsreduzierten Datensatz zu bewerten. Anschließend wurden verschiedenen Klassifizierungsalgorithmen (k-Nearest Neighbour (kNN, grün), Support Vector Machine (SVM, orange), Logistische Regression (LR, blau) und Random Forest (RF, rosa)) verwendet, um die Merkmalsextraktionsqualität zu evaluieren. Auf der y-Achse sind die verschiedenen Dimensionsreduktionsmethoden inklusive des Baseline-Modells aufgetragen. Die x-Achse stellt die durchschnittliche Vorhersagegenauigkeit über die Klassifikatoren der einzelnen Modelle dar.

**Tabelle 6:** Vergleich der Vorgehensweisen verwandter Arbeiten zur Generierung von Embeddings aus Genexpressionsdaten mit einem VAE-Modell. Die gestrichelte Linie trennt die in dieser Arbeit verwendete Vorgehensweise (oben) von den in verwandten Arbeiten verwendeten Vorgehensweisen (unten). Die mit einer Klammer versehenen Parameter stellen die untersuchten Werte für einen spezifischen Hyperparameter dar. CRC = Darmkrebs; GDC = Genomic Data Commons; RNA-Seq = RNA-Sequenzierung; TCGA = The Cancer Genome Atlas; SGD = Stochastic Gradient Descent; Acc = Accuracy; ADAM = Adaptive Moment Estimation.

| Datensatz | Stichprobenanzahl, Dimensionalität, Latente Dimensionalität | Vorverarbeitung | Hyperparameter (Optimierungsalgorithmus, Anzahl der verdeckten Schichten, Lernrate, Drop-out-Rate, Initialisierung der Gewichte, Anzahl der Epochen, Batchgröße) | Evaluation | Ergebnis |
|---|---|---|---|---|---|
| ARCHS4 | 5000; 35.143; (500; 1000; 5000) | Selektion von 5000 Proben; Duplikate entfernen; Ausschluss von Genen mit 100 % Nullen; log2-Transformation; Quantil-Normalisierung; Batch-Effekt-Korrektur; Standardisierung | SGD; (3; 5; 7); (0,05; 0,01; 0,005); (0; 0,2); Zero; (50; 100); 10 | Klassifikation | Acc = 20,04% |
| CRC-Microarray [26] | 238; 1159; 10 | Batch-Effekt-Korrektur; Auswahl von Gene anhand differenzieller Genexpressionsanalyse; Standardisierung | Adam; 5; 0,0005; 0; glorot uniform; 6; 20 | Klassifikation | Acc = 96,92% |
| GDC-RNA-Seq [6] | 11.538; 58.043; 128 | Ausschluss von Gene die auf das Y-Chromosom abzielen, Gene mit Null-Expressionsniveau in allen Proben und Gene mit fehlenden Werten in mehr als 10 % der Proben; Normalisierung | Adam; 5; 0,0001; 0; unbekannt; 50; 32 | Klassifikation | Acc = 96,76% |
| TCGA-RNA-Seq [25] | 9081; 58.043; 128 | Ausschluss von Gene die auf das Y-Chromosom abzielen, Gene mit Null-Expressionsniveau in allen Proben und Gene mit fehlenden Werten in mehr als 10 % der Proben; Normalisierung | Adam; 5; 0,001; 0; unbekannt; 100; 32 | Klassifikation | Acc = 96,85% |
| TCGA-RNA-Seq [90] | 9081; 450.804; 128 | log2-Transformation; Ausschluss von Gene die auf das Y-Chromosom abzielen, Gene mit Null-Expressionsniveau in allen Proben und Gene mit fehlenden Werten in mehr als 10 % der Proben; Normalisierung | Adam; 5; 0,001; 0; unbekannt; 100; 32 | Klassifikation | Acc = 97,49% |

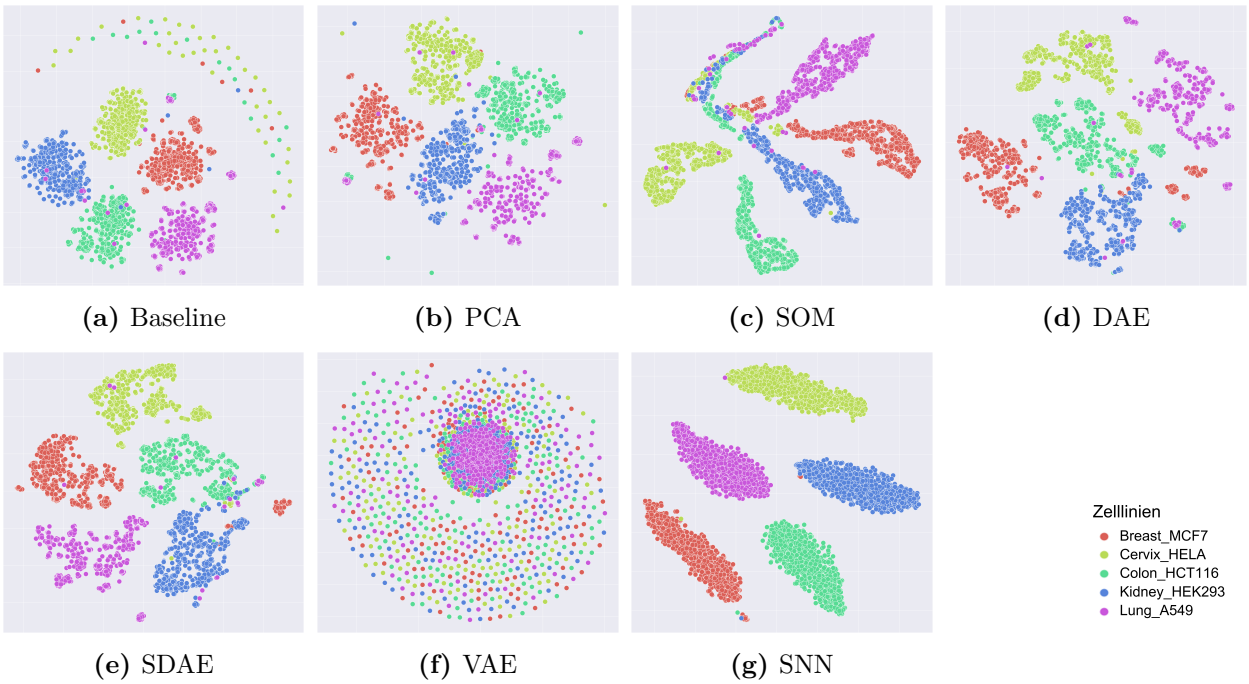
identifizieren. Auffällig ist, dass ein Großteil der verwandten Arbeiten die zu untersuchenden Merkmale im Vorfeld bereits stark selektiert. In dieser Arbeit wurde eine Selektion dieser Art lediglich durch Entfernung von Genen, die über keine der verbleibenden Proben eine Expression größer null gezeigt haben, durchgeführt. Trotz der im Vorfeld durchgeführten Selektion der Gene ist die Anzahl an Merkmalen in einem Großteil der verwandten Arbeiten weiterhin größer als in den hier untersuchten Datensätzen. Die in dieser Arbeit durchgeführte Datentransformation bestehend aus $Log_2$-Transformation, Quantil-Normalisierung, Batch-Effekt-Korrektur sind Standardverfahren zur Vorverarbeitung von Genexpressionsdaten und wurden zudem auf der Web-Ressource ARCHS4 zur Vorverarbeitung vorgeschlagen. Dementsprechend kann auch die Art der Datenvorverarbeitung als Ursache für die schlechte Modellqualität ausgeschlossen werden. Des Weiteren gibt es Unterschiede in den verwendeten Hyperparametern. Dazu gehören die Dimensionalität der latenten Repräsentation, der Optimierungsalgorithmus, die Lernrate sowie die Initialisierung der Gewichte und die Batchgröße. Auffällig ist, dass jede der in Tabelle 6 verglichenen Arbeiten den Adaptive Moment Estimation (ADAM) Optimierungsalgorithmus, eine geringere Lernrate sowie eine höhere Batchgröße verwendet. Die in der vorliegenden Arbeit verwendetet geringere Batchgröße kann als Ursache für die schlechte Modellqualität ausgeschlossen werden. In verschiedenen Arbeiten konnte im Gegensatz gezeigt werden, dass eine Erhöhung der Batchgröße oftmals zu einer Verringerung Testgenauigkeit führt [125, 126, 127, 128]. Der Stochastic Gradient Descent (SGD) sowie die erhöhte Lernrate können Ursachen für die schlechte Modellqualität des VAEs sein. Smith et al. konnten jedoch den SGD-Algorithmus erfolgreich bei einem VAE-Modell, das auf einem vergleichbaren Da-

tensatz trainiert wurde, anwenden [12]. Die erhöhte Lernrate kann ein weiteres Problem darstellen , da diese zu einer Verlangsamung des Trainingsprozesses führen kann [129]. Im Vergleich zu den Weiteren in dieser Arbeit untersuchten Dimensionsreduktionsmethoden konnte ein verlangsamter Trainingsprozess des VAE-Modells festgestellt werden. Allerdings sollte im Falle einer zu hohen Lernrate eine Erhöhung der Epochenanzahl zu einer Verbesserung der Leistung führen, dies konnte jedoch nicht beobachtet werden, wie in Abbildung 16 zu sehen ist. Die anfängliche Initialisierung der Gewichte durch Nullen kommt ebenfalls die Ursache für die schlechte Modellqualität des VAEs in Betracht. Die Untersuchung der Initialisierung der Gewichte gestaltet sich jedoch als schwierig, da ein Großteil der Arbeiten den zugehörigen Wert nicht publiziert haben [6, 90, 25]. Eine gute Initialisierung der Gewichte des AEs ist allerdings essenziell, um die optimale niederdimensionale Repräsentation zu erlernen [130]. Durch die Initialisierung der Anfangsgewichte mit kleinen Werten kann das Problem des verschwindenden Gradienten auftreten [131]. Das Problem des verschwindenden Gradienten äußert sich durch kleine Gradienten. Diese haben zur Folge, dass die Gewichte nur geringfügig optimiert werden, wodurch sich der Trainingsprozess stark verlangsamt [131]. Einige Beobachtungen deuten auf das Problem des verschwindenden Gradienten hin. Zum einen trainiert das VAE Modell sehr langsam und zum anderen laufen die Gradienten regelmäßig gegen null. Des Weiteren führt die Initialisierung der Gewichte mit einem konstanten Wert zu einer Symmetrie in der Gewichtsaktualisierung, die während des gesamten Trainingsprozesses nicht gebrochen werden kann [61]. Berger und Sebag vermuten des Weiteren, dass die Wahl der Verteilungsklasse des Encoders Voraussetzung für ein erfolgreiches VAE-Training ist [132]. Allerdings wurden VAEs mit der hier verwendeten gaußschen Normalverteilung bereits erfolgreich in den präsentierten verwandten Arbeiten (Tabelle 6) auf Microarray-Daten sowie RNA-Seq-Daten verwendet [26, 6, 25].
Neben dem VAE liefert die Vorverarbeitung durch die SOM über alle Klassifikatoren die geringste Accuracy von $\approx 97$ %. Die aufgestellte Hypothese lässt sich durch die hier dargestellten Ergebnisse nicht bestätigen, da im Allgemeinen keine Verbesserung der Vorhersageleistung durch die Dimensionsreduktions-Modelle festgestellt werden kann. Es kann lediglich eine geringfügige Verbesserung der Vorhersageleistung von $\approx 0{,}2$ % durch die Vorverarbeitung mittels SNN im Vergleich zur direkten Verwendung der normalisierten Expressionsdaten (Baseline-Modell) festgestellt werden. Es lässt sich demnach sagen, dass die Integration von Klasseninformationen zur Extrahierung einer geeigneten Repräsentation die Vorhersageleistung verbessern kann. Die Vorverarbeitung durch die restlichen Modelle sorgt für eine Verschlechterung der Vorhersageleistung um bis zu 79 %. Auch lassen sich keine Unterschiede zwischen linearen und nichtlinearen oder traditionellen und DL-Dimensionsreduktionsmethoden erkennen. Der „Fluch der Dimensionalität“ scheint auf Basis des ARCHS4-Datensatzes keinen großen Einfluss auf die Klassifikatoren zu haben. Dies könnte an der verhältnismäßig hohen Stichprobengröße des ARCHS4-Datensatzes liegen sowie an der geringen Komplexität der prädiktiven Aufgabe. Zu erwarten wäre eine Verbesserung der nichtlinearen Dimensionsreduktionsmethoden gegenüber der PCA, aufgrund der Nichtlinearität realer Daten. Allerdings haben Fournier et al. gezeigt, dass sich der Unterschied zwischen linearen und nichtlinearen Dimensionsreduktionsmethoden bei hoher Dimensionalität des Embeddings verringert [133]. Möglicherweise würde auch in dieser Arbeit eine Verringerung der Dimensionalität des Embeddings zu größeren Differenzen in den Ergebnissen führen.
Der Interquartilsabstand ist über alle Modelle mit Ausnahme der SOM sehr gering, was eine geringe

Variabilität der Vorhersageleistung zeigt. Die SOM zeigt mit Abstand die höchste Variabilität der Vorhersageleistung. Dies lässt sich auf die Kombination aus SOM und nachfolgender Klassifikation mittels SVM zurückführen. Die Kombination aus SOM und SVM erreicht lediglich eine Accuracy von $\approx 60$ %. Dies könnte möglicherweise an der Wahl der Kernel-Funktion des SVM-Klassifikators liegen, obwohl der hier verwendetet Radial Basis Funktion Kernel bereits erfolgreich auf nichtlineare Daten wie Genexpressionsdaten angewandt wurde [6]. Das SNN besitzt hingegen den kleinsten Interquartilsabstand und somit auch die geringste Variabilität der Vorhersageleistung. Dies hängt vermutlich mit den nur bei dem SNN verwendeten Seed zusammen, der dafür sorgt, dass die von Tensorflow generierten Pseudozufallszahlen deterministisch sind. Neben der minimalen Variabilität zeigt das SNN auch die beste mediane Accuracy. Dementsprechend lässt sich das SNN als optimales Dimensionsreduktionsmodell identifizieren.
Smith et al. konnten keine Verbesserung der prädiktiven Aufgabe durch Vorverarbeitung mit verschiedenen Dimensionsreduktionsmethoden (PCA, SDAE und VAE) zeigen. Die überwachten Modelle lieferten für die nicht eingebetteten Daten für alle Gene die beste Leistung [12]. Gupta et al. zeigten hingegen eine Verbesserung für das Clustering von Genexpressionsdaten durch Vorverarbeitung mit einem DAE gegenüber den Rohdaten [22]. Des Weiteren konnte die PCA keine geeignete Repräsentation der zugrunde liegenden Verteilung lernen. Franco et al. haben gezeigt, dass die Qualität verschiedener AEs über unterschiedliche Datensätze variiert. Im Durchschnitt haben der Vanilla-AE und der VAE die beste Leistung bei der Subtyperkennung von Genexpressionsdaten gezeigt [50]. Franco et al. verglichen die Leistung der AE-Modelle jedoch nicht mit der Leistung der Rohdaten. Somit kann nicht bestimmt werden, ob die AE-Modelle zu einer Verbesserung der Subtyperkennung geführt haben.

### 4.1.3 Semi-quantitative Bewertung der Fähigkeit von Dimensionsreduktionsmethoden zur Klassentrennung mittels t-distributed Stochastic Neighbor Embedding

Um die generierten niedrigdimensionslaen Repräsentationen weiter zu untersuchen, wurden die Dimensionsreduktionsmethoden auch semi-quantitativ verglichen. Dazu wurden die mittels der Dimensionsreduktionsmethoden generierten Embeddings mit Hilfe von t-SNE in eine zweidimensionale Darstellung transformiert. Anhand dieser zweidimensionalen Visualisierung werden die Dimensionsreduktionsmodelle hinsichtlich ihrer Fähigkeit zur Klassentrennung bewertet. T-SNE ist ein Algorithmus, der verwendet wird, um hochdimensionale Datenpunkte in einem zweidimensionalen Raum so anzuordnen, dass Ereignisse, die durch viele Variablen stark miteinander verwandt sind, am ehesten nebeneinanderliegen. Zu beachten ist jedoch, dass physische Entfernungen zwischen Clustern auf einer t-SNE-Karte nicht im Zusammenhang mit dem Verwandschaftsverhältnis der Daten stehen. Theoretisch ist die optimale Merkmalsdarstellung eine Darstellung mit gleichzeitig maximalem Abstand zwischen Clustern (hohe Trennschärfe) und minimalem Abstand innerhalb von Clustern (hohe Dichte), wobei die mit derselben Farbe markierten „Kreise“ als ein Cluster betrachtet werden können. In Abbildung 18 ist von jedem Modell die zweidimensionale Visualisierung des generierten Embeddings des ersten Durchlaufs der äußeren Kreuzvalidierung dargestellt. Alle Weiteren t-SNE-Visualisierungen sind in Anhang A, Unterabschnitt A.1, Abbildung 24 zu finden.




**(a)** Baseline **(b)** PCA **(c)** SOM **(d)** DAE

**(e)** SDAE **(f)** VAE **(g)** SNN

**Abbildung 18:** Zweidimensionale Visualisierung der erlernten Repräsentationen des ARCHS4-Datensatzes mit t-SNE. Jedes einzelne Diagramm stellt die Visualisierung der generierten Embeddings der Dimensionsreduktionsmodelle (Baseline, PCA, SOM, DAE, SDAE, SNN) dar. Die mit t-SNE generierten 2D-Merkmale aus den Embeddings sind in einem Streudiagramm dargestellt. Die t-SNE-Merkmale 1 und 2 sind auf der x-Achse und y-Achse dargestellt. Die Achsenskala wurde in dieser Arbeit nicht dargestellt, da diese keine Relevanz besitzt [122]. Alle Punkte einer Farbe stehen für die Stichproben der entsprechenden Klasse, wie die Legende (rechts unten) zeigt. PCA = Hauptkomponentenanalyse; SOM = Selbstorganisierende Karte; DAE = Deep Autoencoder; SDAE = Stacked Denoising Autoencoder; VAE = Variational Autoencoder; SNN = siamesisches neuronales Netz.

Die mit t-SNE generierten 2D-Merkmale wurde in einem Streudiagramm aufgetragen. Die Visualisierung des Baseline-Modells (Abbildung 18a), der PCA(Abbildung 18b), der SOM (Abbildung 18c), des DAEs (Abbildung 18d), des SDAE (Abbildung 18e) und des SNNs (Abbildung 18g) zeigen eine klare, deutliche Trennung der fünf Zelllinien-Klassen, was die zugrunde liegenden unterschiedlichen Genexpressionsmuster repräsentiert (Abbildung 18). Die Visualisierung der VAE-Merkmale (Abbildung 18f) lässt keine Trennung der Klassen erkennen. Die Punkte sind kreisförmig und annähernd gleichmäßig verteilt in dem latenten Raum angeordnet, wobei die Dichte der Punkte zur Mitte hin zunimmt. Im Zentrum der Visualisierung scheinen sich lediglich Lung_A549 Stichproben zu befinden, allerdings täuscht die Darstellung aufgrund von Überlagerung. Dies unterstützt die in Unterabschnitt 4.1.2 getroffene Aussage, dass der hier vorgestellte VAE nicht in der Lage ist, geeignete latente Merkmale aus dem Datensatz zu extrahieren. Bei genauerer Untersuchung der sechs weiteren Dimensionsreduktionsmodelle lassen sich weitere deutliche Unterschiede in der Qualität der Klassentrennung erkennen. Es sollte beachtet werden, dass in jedem Streudiagramm mehrere Proben nicht dem erwarteten Cluster zugeordnet wurden, was darauf hindeutet, dass ein kleiner Teil der Proben unterschiedlicher Klassen ähnliche Genexpressionsmuster aufweist. Bei der Visualisierung der nicht-dimensionsreduzierten Expressionsdaten konnten die Klassen der Zelllinien im Allgemeinen getrennt werden, jedoch zeigt die Visualisierung neben den Clustern eine Punktewolke, in der die Merkmale keinem der bestehenden Cluster zugeordnet werden konnten.

Die Dimensionsreduktionsmethoden, mit Ausnahme des VAEs, konnten die undefinierte Punktewolke eliminieren. Die Cluster sind bei der PCA weniger dicht als bei dem Baseline-Modell. Die Visualisierungen der DAE- und SDAE-Merkmale weisen eine hohe Ähnlichkeit auf. Es lässt sich lediglich feststellen, dass die generierten Cluster der SDAE-Merkmale minimal dichter sind. Die SOM zeigt teilweise eine hohe Trennschärfe sowie Dichte der Cluster. Allerdings weisen die Cluster auch eine hohe Überschneidungsrate auf. Das SNN zeigt bei weitem die besten Ergebnisse in der Trennung der Klassen. Die Cluster des SNNs weisen eine hohe Trennschärfe sowie Dichte auf, und es existieren lediglich ein paar wenige Ausreißer. Dies lässt sich auf die Art, wie das neuronale Netz trainiert wurde, zurückführen. Das SNN minimiert den Abstand zwischen zwei Stichproben derselben Klasse und maximiert den Abstand zwischen zwei Stichproben unterschiedlicher Klassen (Abbildung 10). Die Integration der Klasseninformationen bei dem SNN scheint nützlich für die Klassentrennung zu sein. Die Darstellungen der Embeddings sind ein visueller Beweis dafür, dass die Vorverarbeitung durch Dimensionsreduktionsverfahren die Fähigkeit von t-SNE, Datenpunkte zu gruppieren, verbessert. Mit Ausnahme des VAEs weisen alle Visualisierungen der dimensionsreduzierten Merkmale eine höhere Klassentrennung als das Baseline-Modell auf. Dies legt nahe, dass die Informationen im latenten Raum weniger verrauscht sind als die Eingabedaten und somit durch die Dimensionsreduktionsmethoden aussagekräftige und robuste biologische Merkmale generiert wurden. Des Weiteren lässt sich sagen, dass t-SNE in der Lage ist, auf Grundlage der reduzierten Merkmale ähnliche Genexpressionsprofile zu gruppieren.

### 4.1.4 Quantitative Evaluierung der Klassifikatoren

Zuletzt werden die Klassifikatoren (kNN-, SVM-, LR- und RF-Modell) auf Unterschiede in der Qualität bei der Zelllinien-Vorhersage untersucht. In Abbildung 19 ist die Qualität der Klassifikatoren anhand der Accuracy dargestellt. Die Accuracy wird über die verschiedenen, zur Vorverarbeitung der Datensätze verwendeten Dimensionsreduktionsmethoden gemittelt.
Alle Klassifikatoren erreichen eine mediane Accuracy von $\approx$ 98 %. Der SVM-Klassifikator weist den größten Interquartilsabstand und somit auch die größte Variabilität in der Vorhersageleistung auf. Dies lässt sich, wie bereits in Unterabschnitt 4.1.2 beschreiben, auf die Kombination aus SOM und SVM zurückführen. Die Kombination der durch die SOM extrahierten Merkmale sowie der Klassifikation mit der SVM liefert eine geringe Accuracy bei der Zelllinien-Vorhersage. Die Accuracy des VAE-Modells liegt über alle Klassifikatoren bei $\approx$ 20 %. Daher wurden die Ergebnisse des VAEs als Ausreißer behandelt. Gründe für die schlechte Leistung des VAEs wurden bereits in Unterabschnitt 4.1.2 diskutiert. Der kNN und der RF-Klassifikator weisen ähnliche Ergebnisse von $\approx$ 98 % auf. Die LR ist für sechs der sieben Dimensionsreduktionsmodelle die leistungsstärkste Methode mit einer Accuracy von über 99 %. Des Weiteren ist die LR auch die konsistenteste und robusteste Methode. Dementsprechend lässt sich die LR als optimaler Klassifikator identifizieren. Smith et al. konnten ebenfalls gute Ergebnisses mit der LR sowie dem RF erzielen [12].

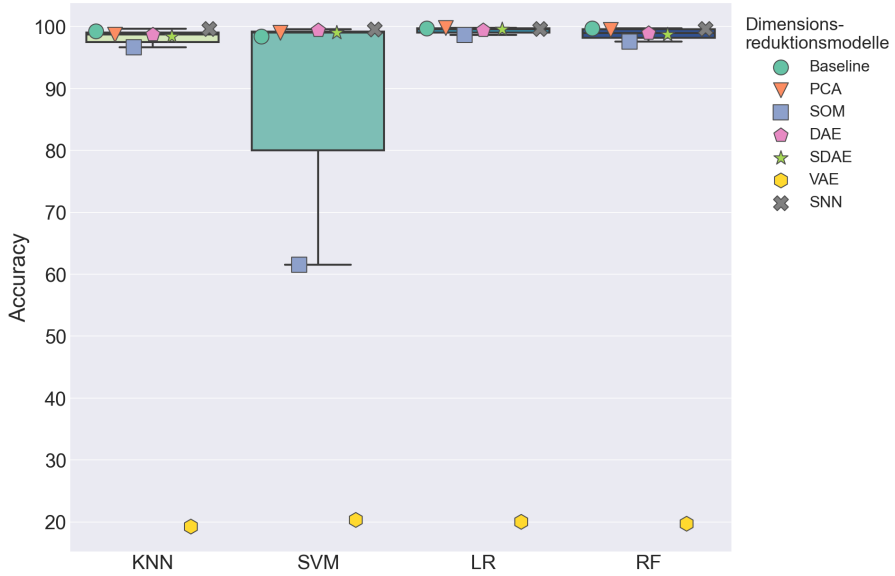



**Abbildung 19:** Qualität der Zelllinien-Vorhersage für den ARCHS4-Datensatz. Die Vorhersageleistung verschiedener Methoden, die auf dem ARCHS4-Datensatz basieren, wurde anhand der Accuracy bewertet. Hauptkomponentenanalyse (PCA; orange), Selbstorganisierende Karte (SOM; blau), Deep Autoencoder (DAE; rosa), Stacked Denoising Autoencoder (SDAE; hellgrün) und Variational Autoencoder (VAE; gelb) und ein siamesisches neuronales Netz (SNN; grau) wurden angewendet, um die Dimension der Datensätze vor der Klassifizierung zu reduzieren. Anschließend wurden k-Nearest Neighbour (kNN), Support Vector Machine (SVM), Logistische Regression (LR) und Random Forest (RF)-Klassifizierungsalgorithmen verwendet, um die Zelllinien-Vorhersagequalität über die verschiedenen Klassifikatoren zu bewerten. Die y-Achse repräsentiert die hier untersuchten Klassifikatoren. Die x-Achse stellt die durchschnittliche Vorhersagegenauigkeit über die Vorverarbeitung durch die Dimensionsreduktionsmodelle inklusive des Baseline-Modells (grün) dar.

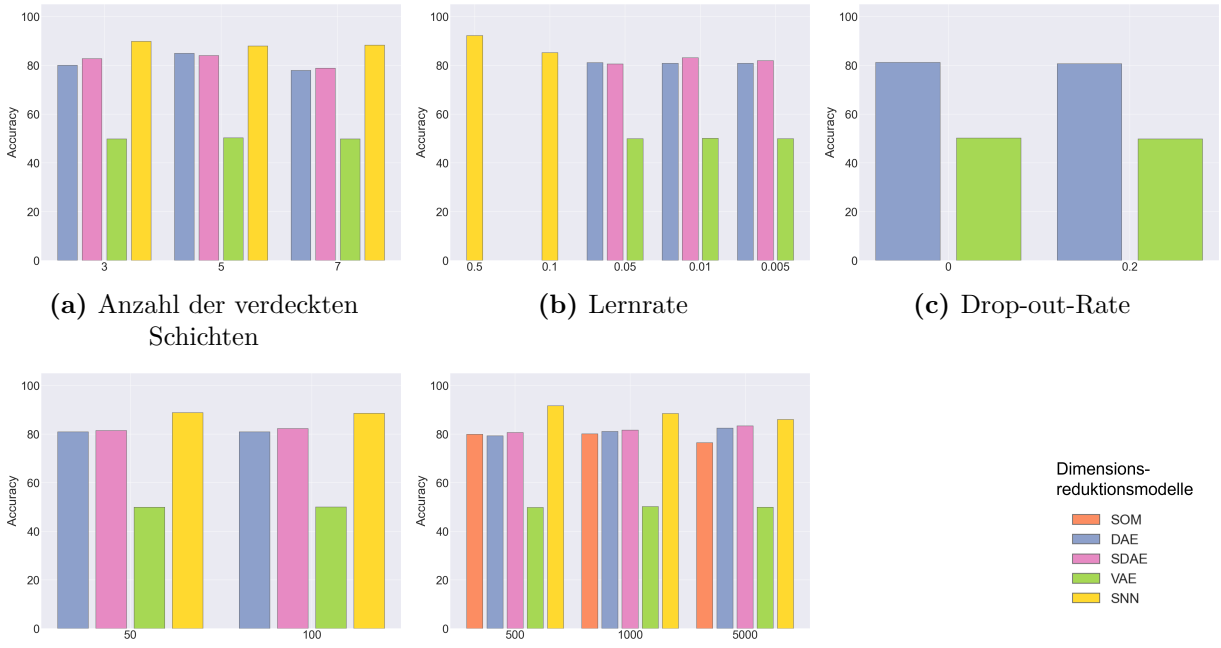
## 4.2 Prostatakrebs-Datensatz

In diesem Kapitel werden die Ergebnisse der auf dem Prostatakrebs-Datensatz beruhenden Experimente vorgestellt und diskutiert. Die PCA konnte aufgrund der geringen Stichprobengröße des Prostatakrebs-Datensatzes nicht zur hier verwendeten Dimensionsreduktion verwendet werden. Dies beruht auf der Tatsache, dass die Anzahl der Hauptkomponenten der PCA $\leq$ der Anzahl an Stichproben sowie der Anzahl an Merkmalen sein muss. Mit einer Stichprobenanzahl von 182 lässt sich dementsprechend selbst das minimale Embedding mit einer Dimensionalität von 500 nicht generieren.

### 4.2.1 Hyperparameteranalyse

Die Dimensionsreduktionsmodelle wurden einer Hyperparameteroptimierung unterzogen. Die Qualität der Dimensionsreduktionsmodelle über alle Hyperparameter-Permutationen hinweg wird in Abbildung 20 zusammengefasst. Die Accuracy für die Werte der einzelnen Hyperparameter setzt sich aus der mittleren Accuracy über alle Klassifikatoren und über alle Hyperparametersets mit dem spezifischen, zu untersuchenden Hyperparameter-Wert zusammen. Zu beachten ist, dass nicht jedes Modell im Hinblick auf alle hier dargestellten Hyperparameter optimiert wird. Die Gründe für den Ausschluss von Hyperparametern liegen großteils in der Art des Trainings der verschiedenen

Modelle. Nähere Informationen dazu lassen sich in Unterabschnitt 3.4.1 finden.




**(a)** Anzahl der verdeckten Schichten **(b)** Lernrate **(c)** Drop-out-Rate

**(d)** Anzahl der Epochen **(e)** Dimensionalität des Embeddings

**Abbildung 20:** Hyperparameteranalyse auf Grundlage des Prostatakrebs-Datensatzes. Für jedes Dimensionsreduktionsmodell wurde eine Hyperparameteroptimierung mit Grid-Search durchgeführt. Die Diagramme a-e stellen jeweils einen zu optimierenden Hyperparameter (Anzahl der verdeckten Schichten, Lernrate, Drop-out-Rate, Anzahl der Epochen, Dimensionalität des Embeddings) dar. Die y-Achse gibt die mittlere Accuracy über die Klassifikatoren und Hyperparametersets mit dem auf der x-Achse dargestellten festgesetzten Parameter an. Der Suchraum der einzelnen Hyperparameter ist jeweils auf der x-Achse dargestellt. Die verschiedenen Dimensionsreduktionsmodelle (SOM, DAE, SDAE, VAE, SNN) sowie ein Baseline-Modell werden durch verschiedene Farben differenziert. SOM = Selbstorganisierende Karte; DAE = Deep Autoencoder; SDAE = Stacked Denoising Autoencoder; VAE = Variational Autoencoder; SNN = siamesisches neuronales Netz.

Ein hoher Accuracy-Wert ist ein vielversprechender Hinweis darauf, dass das jeweilige Dimensionsreduktionsmodell einen geeigneten latenten Raum unter Verwendung des spezifischen Hyperparameter-Wertes gelernt hat. Die Accuracy zeigt, dass die Drop-out-Rate (Abbildung 20c) einen geringen bis keinen Einfluss auf die Leistung der untersuchten Modelle hat. Der VAE zeigt keine Veränderung durch Hinzufügen eines Drop-out-Wertes. Der DAE schneidet ohne Drop-out sogar geringfügig besser ab. Dementsprechend lässt sich sagen, dass der DAE sowie der VAE keine bis wenig Tendenz zum Overfitting aufweisen. Die Anzahl der Epochen wirkt sich ebenfalls nur geringfügig auf die untersuchten Modelle aus. Der DAE, der VAE sowie das SNN zeigen keine Veränderungen durch Erhöhung der Epochen-Anzahl (Abbildung 20d). Daraus lässt sich schließen, dass der Großteil des Trainings dieser Modelle bereits in den ersten 50 Epochen stattgefunden hat. Der SDAE erreicht eine geringfügig bessere Accuracy bei 100 Epochen. Die weiteren Hyperparameter haben größeren Einfluss auf die Qualität der Modelle. Der VAE zeigt über keinen der Hyperparameterwerte eine Veränderung. Dies hängt mit den in Unterabschnitt 4.1.2 bereits erläuterten grundlegenden Problemen des hier verwendeten VAEs zusammen. Die Anzahl der verdeckten Schichten tangiert den DAE, den SDAE sowie das SNN (Abbildung 20a). Der DAE sowie der SDAE erreichen ihre optimale Leistung bei 5 verdeckten Schichten. Das SNN benötigt hingegen nur 3 verdeckte Schichten.

Das SNN scheint demnach mit einer niedrigen Modelltiefe besser mit komplexeren Daten umgehen zu können als der DAE oder der SDAE. Die Lernrate hat den größten Einfluss auf den SDAE (Abbildung 20b). Die Qualität des SDAEs nimmt bei abnehmender Lernrate zu, wobei die Differenz der Qualität bei Verwendung einer Lernrate von 0,01 und einer Lernrate von 0,005 sehr gering ist. Der DAE erbringt eine robuste Leistung über die verschiedenen Lernraten. Die optimale Leistung des SNNs zeigt sich bei einer Lernrate von 0,5. Die Veränderung der Dimensionalität des Embeddings wirkt sich auf alle Modelle mit Ausnahme des VAE-Modells aus (Abbildung 20e). Die Qualität der SOM und des SNNs nimmt bei zunehmender Embeddinggröße ab, wohingegen die Qualität des DAEs und des SDAEs bei zunehmender Embeddinggröße zunimmt. Die intrinsische Dimensionalität des Prostatakrebs-Datensatzes lässt sich nicht identifizieren.
Es fällt auf, dass lediglich der VAE robust gegenüber Veränderungen der hier verwendeten Hyperparameter ist. Die restlichen Modelle reagieren dagegen auf Veränderungen der Anzahl an verdeckten Schichten, der Lernrate und der Dimensionalität des Embeddings. Die Veränderungen der Leistung über die Hyperparameter-Werte sind jedoch großteils gering. Die Robustheit des VAEs lässt sich vermutlich auf die Probleme des VAEs zurückführen. Es wurde bereits gezeigt, dass VAEs sehr empfindlich auf Veränderungen der Hyperparameter regieren [134].

**Tabelle 7:** Optimale Hyperparametersets der verwendeten Modelle auf Grundlage des Prostatakrebs-Datensatzes. Für jedes Modell wurde aus der Gesamtmenge der Hyperparameter-Permutationen pro Durchgang der Kreuzvalidierung ein optimales Set an Hyperparametern anhand der gemittelten Accuracy über die Klassifikatoren ausgewählt. Dementsprechend wurden fünf optimale Hyperparametersets pro Modell identifiziert. Die Tabelle führt die Hyperparameter zusammen mit ihrem metrischen Wert (Accuracy) auf. PCA = Hauptkomponentenanalyse; SOM = Selbstorganisierende Karte; DAE = Deep Autoencoder; SDAE = Stacked Denoising Autoencoder; VAE = Variational Autoencoder; SNN = siamesisches neuronales Netz.

| Model | Äußere Schleife der verschachtelten Kreuzvalidierung | Anzahl der verdeckten Schichten | Lernrate | Drop-out-Rate | Anzahl der Epochen | Dimensionalität des Embeddings | Accuracy |
|---|---|---|---|---|---|---|---|
| SOM | 1 | - | - | - | - | 500 | 83.78 |
| | 2 | - | - | - | - | 1000 | 84.46 |
| | 3 | - | - | - | - | 1000 | 85.81 |
| | 4 | - | - | - | - | 500 | 79.05 |
| | 5 | - | - | - | - | 1000 | 75.68 |
| DAE | 1 | 3 | 0.05 | 0.2 | 100 | 1000 | 81.08 |
| | 2 | 5 | 0.05 | 0 | 100 | 500 | 80.41 |
| | 3 | 5 | 0.005 | 0.2 | 100 | 5000 | 82.43 |
| | 4 | 5 | 0.01 | 0 | 100 | 5000 | 83.11 |
| | 5 | 5 | 0.01 | 0.2 | 50 | 5000 | 79.05 |
| SDAE | 1 | 3 | 0.05 | - | 50 | 1000 | 85.14 |
| | 2 | 5 | 0.01 | - | 100 | 5000 | 85.14 |
| | 3 | 5 | 0.01 | - | 100 | 5000 | 83.11 |
| | 4 | 5 | 0.01 | - | 100 | 5000 | 83.78 |
| | 5 | 5 | 0.01 | - | 100 | 5000 | 82.43 |
| VAE | 1 | 5 | 0.005 | 0 | 50 | 1000 | 46.62 |
| | 2 | 5 | 0.05 | 0 | 50 | 1000 | 45.27 |
| | 3 | 3 | 0.005 | 0 | 50 | 5000 | 45.95 |
| | 4 | 5 | 0.05 | 0.2 | 100 | 500 | 39.19 |
| | 5 | 3 | 0.01 | 0 | 50 | 1000 | 51.35 |
| SNN | 1 | 3 | 0.5 | - | 50 | 500 | 97.97 |
| | 2 | 3 | 0.5 | - | 50 | 1000 | 95.27 |
| | 3 | 3 | 0.5 | - | 50 | 500 | 97.97 |
| | 4 | 3 | 0.5 | - | 100 | 500 | 100 |
| | 5 | 3 | 0.5 | - | 50 | 500 | 93.92 |

Für jedes Modell wurde, wie in Unterabschnitt 3.4.1 bereits erläutert, aus der Gesamtmenge der Hyperparameter-Permutationen über alle Durchläufe der äußeren Kreuzvalidierung ein optimales Set an Hyperparametern anhand der gemittelten Accuracy über die Klassifikatoren ausgewählt. Die

fünf optimalen Hyperparametersets jedes Modells sowie deren gemittelte Accuracy sind in Tabelle 7 dargestellt. Die in Tabelle 7 dargestellten optimalen Hyperparametersets decken sich großteils mit den zuvor ermittelten Ergebnissen der in Abbildung 20 dargestellten Hyperparameteranalyse. Die Analysen in den späteren Abschnitten basieren auf den Ergebnissen der optimalen Hyperparametersets (Tabelle 7). Die vollständigen Ergebnisse der Hyperparameteroptimierung finden sich auf dem externen Speichermedium.

### 4.2.2 Quantitative Evaluierung der Dimensionsreduktionsmethoden anhand der Klassifikator-Genauigkeit

Die Merkmalsextraktionsqualität der Dimensionsreduktionsmodelle wurde quantitativ anhand der gemittelten Klassifikator-Genauigkeit über vier verschiedene Klassifikatoren (kNN, SVM, LR, RF) evaluiert. Diese Evaluierung beruht auf der Hypothese, dass Prädiktoren, die auf niederdimensionalen Darstellungen trainiert werden, eine bessere Vorhersagequalität erzielen [12]. Um diese Hypothese zu überprüfen, wurde zusätzlich ein Baseline-Modell verwendet, welches die Vorhersageleistung auf dem nicht dimensionsreduzierten Datensatz bewertet. Eine hohe Klassifikator-Genauigkeit ist ein vielversprechender Hinweis darauf, dass das jeweilige Dimensionsreduktionsmodell einen geeigneten latenten Raum gelernt hat. In Abbildung 21 ist die Leistung der Dimensionsreduktionsmodelle anhand der gemittelten Klassifikator-Genauigkeit dargestellt.
Die Vorhersageleistung auf Grundlage der durch die Dimensionsreduktionsmodelle generierten Embeddings liegt bei allen Modellen mit Ausnahme des VAEs bei über 82 % Accuracy. Die beste durchschnittliche Accuracy von $\approx$ 98 % generiert das Embedding des SNN. Dahinter folgt das Baseline-Modell mit einer Accuracy von $\approx$ 96 %, der SDAE mit einer Accuracy von $\approx$ 94 %, der DAE mit einer Accuracy von $\approx$ 89 % und die SOM mit einer Accuracy von $\approx$ 83 %. Das durch den VAE generierte Embedding führt lediglich zu einer Accuracy von $\approx$ 45 %. Der VAE scheint nicht in der Lage zu sein, eine geeignete latente Repräsentation aus den Eingabedaten zu extrahieren. Mögliche Ursachen hierfür wurden in Unterabschnitt 4.1.2 bereits erläutert. Die aufgestellte Hypothese lässt sich durch die hier dargestellten Ergebnisse nicht bestätigen. Es kann zwar eine Verbesserung der Vorhersageleistung durch die Vorverarbeitung mit SNNs im Vergleich zur direkten Verwendung der normalisierten Expressionsdaten (Baseline-Modell) festgestellt werden, allerdings liegt die Differenz bei lediglich 2 %. Des Weiteren ist das SNN das einzige Dimensionsreduktionsmodell, das eine bessere Vorhersageleistung als das Baseline-Modell erzielt, alle weiteren Modelle sorgen für eine Verschlechterung der Vorhersageleistung. Im Übrigen lassen sich keine Unterschiede zwischen linearen und nichtlinearen oder traditionellen und DL-Dimensionsreduktionsmethoden erkennen. Der Interquartilsabstand ist über das Baseline-, das SNN- sowie das VAE-Modell gering und über die SOM, den DAE sowie den SDAE verhältnismäßig hoch, wobei der DAE den höchsten Interquartilsabstand besitzt. Der Interquartilsabstand gibt die Variabilität der Vorhersageleistung an. Auch wenn der Qualitätsunterschied zwischen Baseline-Modell und SNN gering ist, lässt sich das SNN als optimales Modell identifizieren. Die Zuhilfenahme der Klassenbezeichnung beim Trainieren eines Dimensionsreduktionsmodells scheint die Qualität der Merkmalsextraktionsleistung zu verbessern.

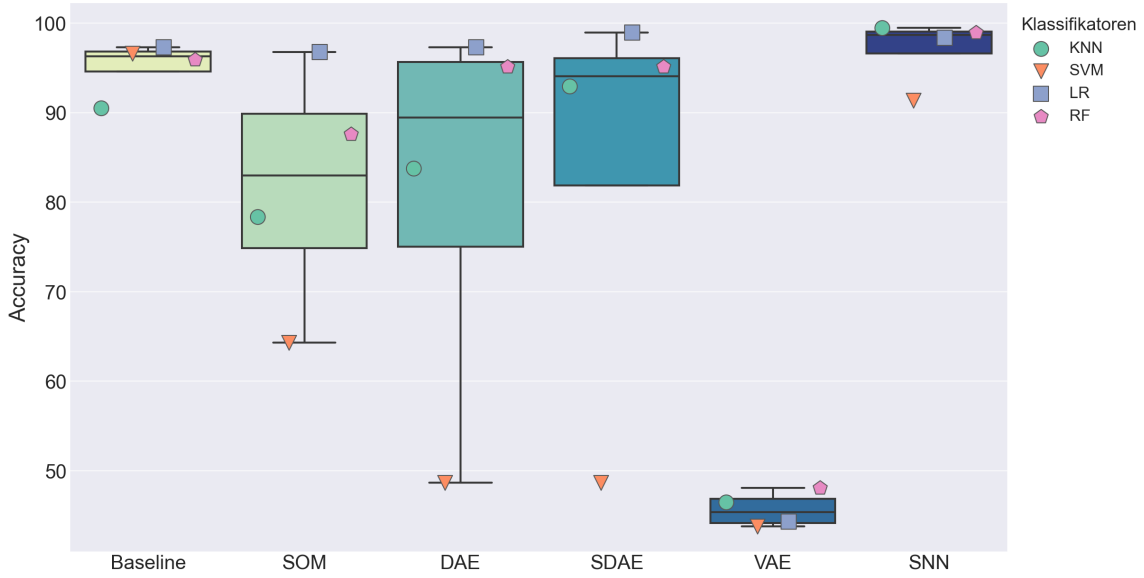
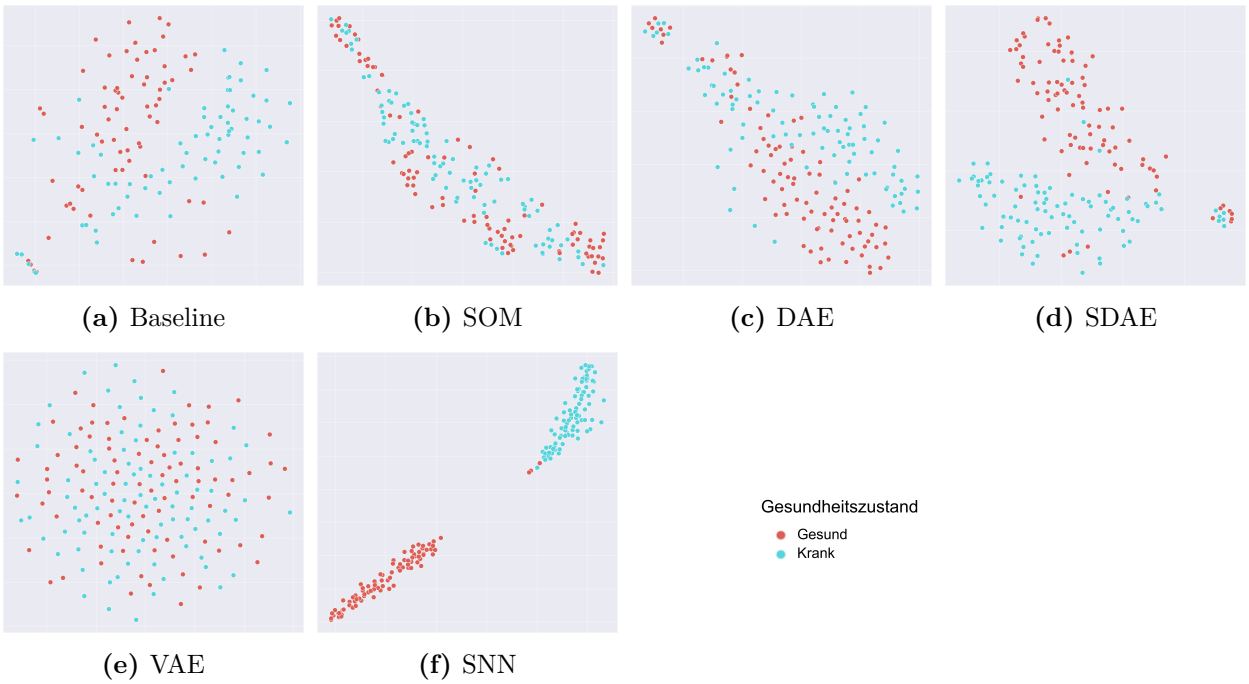



**Abbildung 21:** Quantitative Evaluierung der Dimensionsreduktionsmethoden anhand der Klassifikator-Genauigkeit auf Basis des Prostatakrebs-Datensatzes. Die Merkmalsextraktionsqualität der Dimensionsreduktionsmethoden, die auf dem Prostatakrebs-Datensatz basieren, wurde anhand der Unterscheidung zwischen Prostatakrebs-Proben und Proben aus gesundem Gewebe durch vier Klassifikatoren bewertet. Hauptkomponentenanalyse (PCA), Selbstorganisierende Karte (SOM), Deep Autoencoder (DAE), Stacked Denoising Autoencoder (SDAE) und Variational Autoencoder (VAE) und ein siamesisches neuronales Netz (SNN) wurden angewendet, um die Dimension des Prostatakrebs-Datensatzes vor der Klassifizierung zu reduzieren. Anschließend wurden verschiedenen Klassifizierungsalgorithmen (k-Nearest Neighbour (kNN, grün), Support Vector Machine (SVM, orange), Logistische Regression (LR, blau) und Random Forest (RF, rosa)) verwendet, um die Merkmalsextraktionsqualität zu bewerten. Die y-Achse repräsentiert die verschiedenen Dimensionsreduktionsmethoden inklusive des Baseline-Modells. Die x-Achse stellt die durchschnittliche Vorhersagegenauigkeit über die Klassifikatoren der einzelnen Modelle dar.

### 4.2.3 Semi-quantitative Bewertung der Fähigkeit von Dimensionsreduktionsmethoden zur Klassentrennung mittels t-distributed Stochastic Neighbor Embedding

Zur weiteren Untersuchung der generierten Embeddings wurden die traditionellen Dimensionsreduktionsmethoden sowie die DL-Dimensionsreduktionsmethoden auch semi-quantitativ verglichen. Dazu wurden die mittels der Dimensionsreduktionsmethoden generierten Embeddings mit Hilfe von t-SNE in eine zweidimensionale Darstellung transformiert. Anhand dieser zweidimensionalen Visualisierung werden die Dimensionsreduktionsmodelle hinsichtlich ihrer Fähigkeit zur Klassentrennung bewertet. T-SNE ist ein Algorithmus, der verwendet wird, um hochdimensionale Datenpunkte in einem zweidimensionalen Raum so anzuordnen, dass verwandte Datenpunkte, am wahrscheinlichsten nebeneinanderliegen. In Abbildung 22 ist von jedem Modell die zweidimensionale Visualisierung des generierten Embeddings des ersten Durchlaufs der äußeren Kreuzvalidierung dargestellt. Alle weiteren t-SNE-Visualisierungen sind in Anhang B, Unterabschnitt B.1, Abbildung 25 zu finden.
Die mit t-SNE generierten 2D-Merkmale wurden in einem Streudiagramm aufgetragen. Es konnte lediglich durch das SNN eine räumliche Trennung der Klassen erzeugt werden. Allerdings lässt sich in den übrigen Visualisierungen mit Ausnahme der Darstellung des VAE-Embeddings eine lineare Trennlinie mit einer niedrigen Anzahl an falsch klassifizierten Punkten identifizieren. Die fehlende




**(a)** Baseline **(b)** SOM **(c)** DAE **(d)** SDAE

**(e)** VAE **(f)** SNN

**Abbildung 22:** Zweidimensionale Visualisierung der erlernten Repräsentationen des Prostatakrebs-Datensatzes mit t-SNE. Die Diagramme a-f stellen jeweils die Visualisierung eines Dimensionsreduktionsmodells (SOM, DAE, SDAE, VAE, SNN) sowie eines Baseline-Modells dar. Die mit t-SNE generierten 2D-Merkmale aus den mittels der Dimensionsreduktionsmethoden generierten Embeddings sind in einem Streudiagramm dargestellt. Die t-SNE-Merkmale 1 und 2 sind auf der x-Achse und y-Achse dargestellt. Die Achsenskala wurde in dieser Arbeit nicht dargestellt, da diese keine Relevanz besitzt [122]. Alle Punkte einer Farbe stehen für die Stichproben der entsprechenden Klasse, wie die Legende (rechts unten) zeigt. Wie bereits erläutert, konnte keine PCA angewandt werden. PCA = Hauptkomponentenanalyse; SOM = Selbstorganisierende Karte; DAE = Deep Autoencoder; SDAE = Stacked Denoising Autoencoder; VAE = Variational Autoencoder; SNN = siamesisches neuronales Netz.

Fähigkeit zur Klassentrennung der Dimensionsreduktionsmodelle ist jedoch nicht auf die Dimensionsreduktion an sich, sondern vielmehr auf die spärliche Stichprobenanzahl zurückzuführen. Diese Aussage kann mit der ebenfalls mangelnden Fähigkeit zur Klassentrennung des Baseline-Modells begründet werden. Die Visualisierungen des Baseline-Modells (Abbildung 22a) und des DAEs (Abbildung 22c) sind vergleichbar, die DAE-Merkmale erzeugen lediglich eine dichtere Darstellung als die unveränderten Merkmale des Baseline-Modells. Die Visualisierung der SDAE-Merkmale (Abbildung 22d) erreicht eine Verbesserung der Klassentrennung gegenüber den DAE-Merkmalen. In der Visualisierung des SDAEs liegen die Klassen in einer größeren räumlichen Distanz voneinander vor. In der Visualisierung der Baseline-Merkmale, der DAE-Merkmale und der SDAE-Merkmale lässt sich ein einheitliches Muster erkennen. Die Klassen bilden 2 Stränge, die an einem Ende gut getrennt vorliegen und an dem anderen Ende miteinander verschmelzen. Die Visualisierung der SOM (Abbildung 22b) weist eine schlechte Klassentrennung auf. Die Punkte beider Klassen liegen verteilt auf einer Geraden in dem latenten Raum. Möglicherweise würde hier eine 3D-Visualisierung die Klassentrennung verbessern. In der Visualisierung der VAE-Merkmale (Abbildung 22e) lässt sich keine Trennung der Klassen erkennen, da die Punkte gleichmäßig verteilt in dem latenten Raum vorliegen. Dies unterstützt die in Unterabschnitt 4.2.2 getroffene Aussage, dass der hier vorgestellte VAE nicht in der Lage ist, geeignete latente Merkmale aus dem Datensatz zu extrahieren.

Das SNN (Abbildung 22f) zeigt bei Weitem die besten Ergebnisse in der Trennung der Klassen. Die Klassen liegen als zwei Cluster mit großer räumlicher Distanz im latenten Raum vor. Dies lässt sich wie schon in Unterabschnitt 4.1.3 beschrieben, auf die Art, wie das neuronale Netz trainiert wurde, zurückführen. Die Visualisierung der SNN-Merkmale sowie der SDAE-Merkmale weisen eine Verbesserung gegenüber des Baseline-Modells auf. Dies legt nahe, dass die Informationen der SNN-Merkmale sowie der SDAE-Merkmale weniger verrauscht sind als die Eingabedaten und dass somit durch die Dimensionsreduktionsmethoden aussagekräftigere und robustere biologische Merkmale generiert wurden. Der SDAE, der DAE sowie das SNN sind in der Lage, die Fähigkeit von t-SNE, Datenpunkte zu gruppieren, zu verbessern. Des Weiteren lässt sich sagen, dass t-SNE in der Lage ist, auf Grundlage der reduzierten-Merkmale Genexpressionsprofile mit ähnlichen Merkmalen zu gruppieren.

### 4.2.4 Quantitative Evaluierung der Klassifikatoren

Zuletzt werden Unterschiede in der Qualität zwischen den kNN-, SVM-, LR- und RF-Modellen bei der Unterscheidung zwischen Prostatakrebs-Proben und Proben aus gesundem Gewebe untersucht. In Abbildung 23 ist die Qualität der Klassifikatoren anhand der Accuracy dargestellt.

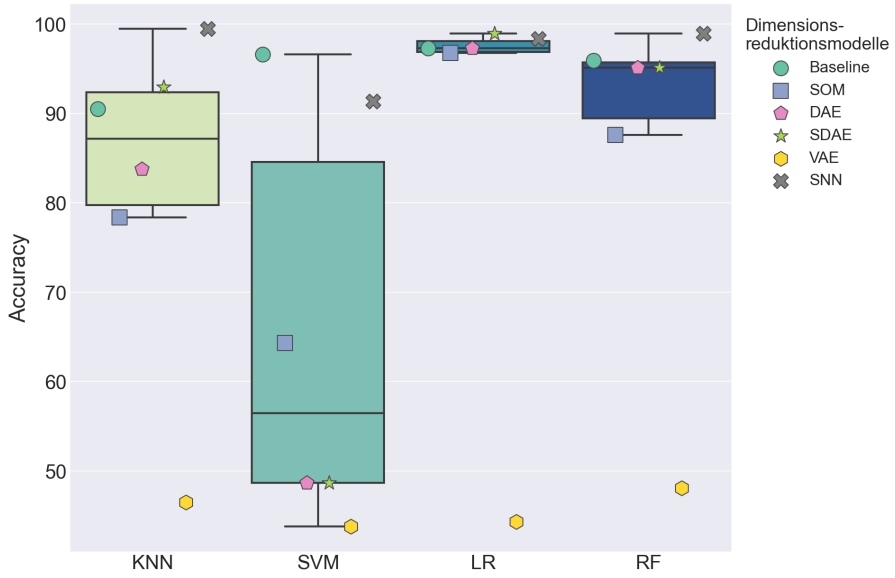



**Abbildung 23:** Leistung der Unterscheidung zwischen Prostatakrebs-Proben und Proben aus gesundem Gewebe für den Prostatakrebs-Datensatz. Die Vorhersageleistung verschiedener Klassifikations-Methoden, die auf dem Prostatakrebs-Datensatz basieren, wurde anhand der Accuracy bewertet. Selbstorganisierende Karte (SOM; blau), Deep Autoencoder (DAE; rosa), Stacked Denoising Autoencoder (SDAE; hellgrün), Variational Autoencoder (VAE; gelb) sowie ein siamesisches neuronales Netz (SNN; grau) wurden angewendet, um die Dimension der Datensätze vor der Klassifizierung zu reduzieren. Anschließend wurden k-Nearest Neighbour (kNN), Support Vector Machine (SVM), Logistische Regression (LR) und Random Forest (RF)-Klassifizierungsalgorithmen verwendet, um die Unterscheidung zwischen Prostatakrebs-Proben und Proben aus gesundem Gewebe über die verschiedenen Klassifikatoren zu bewerten. Die x-Achse repräsentiert die verschiedenen Klassifikatoren. Die y-Achse stellt die durchschnittliche Vorhersagegenauigkeit über die Vorverarbeitung durch die Dimensionsreduktionsmodelle inklusive des Baseline-Modells (grün) dar.

Die Accuracy wird über die verschiedenen, zur Vorverarbeitung der Datensätze verwendeten Di-

mensionsreduktionsmethoden, gemittelt. Alle Klassifikatoren, mit Ausnahme der SVM, erreichen eine mediane Accuracy von über 87 %. Die LR erreicht eine Accuracy von 97 %, der RF eine Accuracy von 95 % und der kNN eine Accuracy von 87 %. Der SVM-Klassifikator weist die geringste Accuracy von lediglich $\approx$ 56 % sowie den größten Interquartilsabstand auf und somit auch die größte Variabilität in der Vorhersageleistung. Die schlechte Qualität der SVM lässt sich jedoch nicht auf die Prostatakrebs-Daten zurückführen, da die SVM auf den Merkmalen des Baseline-Modells gute Ergebnisse erzielt. Vielmehr lässt sich die Qualität der SVM auf die vorherige Merkmalsextraktion zurückführen. Möglicherweise liegt die Ursache für die schlechten Ergebnisse des SVM-Klassifikators in der Wahl der Kernelfunktion begründet. Der kNN weist nach der SVM die schlechteste mediane Accuracy auf sowie den zweitgrößten Interquartilsabstand. Der RF-Klassifikator zeigt die zweitbeste mediane Accuracy. Die LR ist für vier der sechs Dimensionsreduktionsmodelle die leistungsstärkste Methode. Des Weiteren weist die LR auch den geringsten Interquartilsabstand und somit auch die geringste Variabilität in der Vorhersageleistung auf und lässt sich daher als optimaler Klassifikator identifizieren. Smith et al. konnten ebenfalls gute Ergebnisses mit der LR sowie dem RF erzielen [12].

## 4.3 Vergleich der Ergebnisse hinsichtlich der Datensätze

In diesem Kapitel werden die in den vorherigen Abschnitten präsentierten Ergebnisse im Hinblick auf Unterschiede zwischen den Genexpressionsdatensätzen (ARCHS4- und Prostatakrebs-Datensatz) verglichen. Unterschiede in der Qualität der Ergebnisse zwischen den hier untersuchten Datensätzen können auf die Stichprobengröße, die Varianz innerhalb und zwischen den Klassen sowie die Art der Klassifikation (binäre Klassifizierung und Mehrklassenklassifizierung) zurückgeführt werden. Der Prostatakrebs-Datensatz besitzt aufgrund der höheren biologischen Variabilität der Proben eine höhere Varianz innerhalb einer Klasse. Wie bereits im Laufe der Arbeit erläutert, leiden Datensätze mit einer geringen Stichprobengröße unter dem „Fluch der Dimensionalität“, sind dementsprechend oft überangepasst und weisen eine hohe Varianz auf. Anhand der Stichprobengröße und der Varianz innerhalb und zwischen den Klassen lässt sich vermuten, dass die Qualität der Ergebnisse des ARCHS4-Datensatzes im Allgemeinen die Ergebnisse des Prostatakrebs-Datensatzes übertreffen. Diese Vermutung lässt sich durch die präsentierten Ergebnisse des ARCHS4- und des Prostatakrebs-Datensatzes bestätigen. Der ARCHS4-Datensatz weist einen deutlich geringeren Interquartilsabstand bei der Evaluierung der Dimensionsreduktionsmethoden (Unterabschnitt 4.1.2, Unterabschnitt 4.2.2) als auch bei der Evaluierung der Klassifikatoren (Unterabschnitt 4.1.4, Unterabschnitt 4.2.4) als der Prostatakrebs-Datensatz auf. Dies lässt sich auf die Unterschiede in der Stichprobengröße zurückführen, da sowohl die Dimensionsreduktionsmethoden als auch die Klassifikatoren bei einer geringen Stichprobengröße zum Overfitting neigen. Bei Betrachtung der zweidimensionalen Visualisierung der Merkmale lässt sich feststellen, dass t-SNE die dimensionsreduzierten ARCHS4-Merkmale in geeignete Cluster separieren kann, wohingegen die dimensionsreduzierten Prostatakrebs-Merkmale die Fähigkeit der manuellen Clusterbildung von t-SNE beeinträchtigen. Die Ursache dafür liegt möglicherweise in einer hohen Varianz innerhalb oder in einer niedrigen Varianz zwischen den Klassen des Prostatakrebs-Datensatzes. Auffällig ist,

dass das VAE-Modell unter Verwendung des Prostatakrebs-Datensatzes eine höhere Accuracy als unter Verwendung des ARCHS4-Datensatzes aufweist. Die Ursache ist vermutlich die Unterscheidung der prädiktiven Aufgabe zwischen binärer Klassifikation auf Grundlage des Prostatakrebs-Datensatzes und Mehrklassenklassifizierung auf Grundlage des ARCHS4-Datensatzes. Bei beiden Datensätzen mit zugehöriger prädiktiver Aufgabe würde eine zufällige Klassifikation der Stichproben vergleichbare Ergebnisse wie der VAE erzielen. Der empirische Vergleich der Klassifikatoren zeigt eine Verbesserung der Accuracy durch die Verwendung der LR gegenüber den weiteren Klassifikatoren (kNN, SVM, RF) in beiden Datensätzen. Des Weiteren zeigt die SVM für beide Datensätze die schlechteste Accuracy. Die Ursache für die schlechte Leistung der SVM besteht möglicherweise in der Verwendung einer suboptimalen Kernelfunktion. Das SNN liefert für beide Datensätze die besten Ergebnisse, während der VAE am schlechtesten abschnitt. Die in dieser Arbeit präsentierten Ergebnisse deuten auf eine minimale Verbesserung der Vorhersagegenauigkeit durch die überwachte Dimensionsreduktion mit einem SNN gegenüber unüberwachter Dimensionsreduktion sowie keiner Dimensionsreduktion hin. Eine Verbesserung der Genauigkeit der nichtlinearen Dimensionsreduktionsmethoden gegenüber den linearen Dimensionsreduktionsmethoden lässt sich nicht erkennen, genauso wenig wie eine Verbesserung der Genauigkeit der DL-Dimensionsreduktionsmethoden gegenüber den traditionellen Dimensionsreduktionsmethoden.

# 5 Zusammenfassung und Ausblick

In dieser Arbeit wurde die Qualität von vier verschiedenen DL-Methoden zur Dimensionsreduktion (DAE, VAE, SDAE, SNN) anhand einer nachfolgenden Klassifikationsaufgabe an zwei unterschiedlichen Genexpressionsdatensätzen demonstriert. Das Hauptziel dieser Arbeit war es, die Wirksamkeit der Dimensionsreduktion als Methode zur Extraktion von Genmerkmalen für die Klassifizierung von Genexpressionsdaten zu untersuchen. Die DL-Methoden wurden mit traditionellen Dimensionsreduktionsmethoden sowie einem Baseline-Modell, welches die Daten nicht reduziert, qualitativ verglichen, um zu untersuchen, inwieweit DL-Methoden eine Verbesserung gegenüber den traditionellen Dimensionsreduktionsmethoden/dem Baseline-Modell erreichen können. Die Merkmalsextraktionsqualität der Dimensionsreduktionsmodelle wurde quantitativ anhand der gemittelten Klassifikator-Genauigkeit über vier verschiedene Klassifikatoren und semi-quantitativ anhand der zweidimensionalen Visualisierung mit t-SNE evaluiert. Die Evaluierung anhand der Klassifikation beruht auf der Hypothese, dass Prädiktoren, die auf niederdimensionalen Darstellungen trainiert werden, eine bessere Vorhersagequalität erzielen können [12]. Anhand der zweidimensionalen Visualisierung lässt sich die Fähigkeit der Dimensionsreduktionsmethoden zur Trennung der Datenklassen bewerten. Es konnte gezeigt werden, dass das SNN über beide Datensätze die besten Ergebnisse sowohl bei der Visualisierung als auch bei der Evaluierung anhand der Klassifikation lieferte, während der VAE über beide Datensätze am schlechtesten abschnitt. Das SNN konnte als einziges Modell die Vorhersagegenauigkeit über die Datensätze hinweg verbessern. Dementsprechend lässt sich sagen, dass die Verwendung des SNNs als Methode zur Extraktion von Genen die Klassifikation von Genexpressionsdaten verbessern kann.
Des Weiteren wurden die Qualität von vier Klassifikatoren (kNN, SVM, LR, RF) auf Grundlage der von den Dimensionsreduktionsmethoden generierten Embeddings bewertet. Aus der Beobachtung der Genauigkeit ging hervor, dass die LR über beide dimensionsreduzierte Datensätze die besten Ergebnisse liefert, während die SVM über beide dimensionsreduzierte Datensätze am schlechtesten abschnitt. Insgesamt lässt sich sagen, dass Dimensionsreduktionsverfahren die Klassifikation von Genexpressionsdaten nicht zwangsläufig verbessern. Ebenso lässt sich keine Verbesserung der Genauigkeit der DL-Dimensionsreduktionsmethoden gegenüber den traditionellen Dimensionsreduktionsmethoden erkennen.

Neben der Verbesserung der Datenqualität für nachfolgende prädiktive Aufgaben werden Dimensionsreduktionsmethoden auch häufig zur Reduzierung der Speicherauslastung und des Rechenaufwands verwendet. Mit diesem Ziel ist die Verbesserung der Vorhersageleistung, indem Prädiktoren auf niederdimensionalen Darstellungen trainiert werden, zwar wünschenswert, jedoch nicht essenziell. Solange die Embeddings der Dimensionsreduktionsmethoden sowie das Baseline-Modell vergleichbare Qualität der Vorhersageleistung liefern, lassen sich die Embeddings zur Reduzierung der Speicherauslastung und des Rechenaufwands verwenden. Dies ist bei der Integration verschiedener medizinischer Daten, die ergänzende Informationen für denselben Patienten liefern, besonders nützlich, da Omics-Datensätze tendenziell eine hohe Dimensionalität aufweisen. Die Analyse von Multi-Omics-Daten ist in der Lage, die klinische Entscheidungsfindung zu erleichtern [6]. Die in dieser Arbeit generierten Embeddings können in nachfolgenden wissenschaftlichen Arbeiten als Entitä-

ten in sogenannten Wissensgraphen verwendet werden. Wissensgraphen bieten die Möglichkeit, heterogene Daten, wie zum Beispiel Multi-Omics Daten und ihre Beziehung zueinander effektiv darzustellen.

Neben der Verwendung der Embeddings in Wissensgraphen gibt es weitere Forschungsrichtungen, welche die vorliegende Arbeit ausbauen können:

- Die Leistung der in dieser Arbeit verwendeten Dimensionsreduktionsmethoden sollte erneut anhand einer geringeren Dimensionalität der Embeddings verglichen werden, um festzustellen, ob die Unterschiede zwischen DL/traditionell sowie linear und nichtlinearen Dimensionsreduktionsmodellen mit abnehmender Embeddinggröße zunehmen.
- Des Weiteren konnten in dieser Arbeit keine zufriedenstellenden Ergebnisse mit den VAE-Modell erzeugt werden. Es wäre sinnvoll das VAE-Modell auf Grundlage der in Unterabschnitt 4.1.2 diskutierten möglichen Probleme zu überarbeiten und erneut zu bewerten.
- Darüberhinaus sollten weitere Dimensionsreduktionsmethoden sowie weitere Genexpressionsdatensätze in den Vergleich miteinbezogen werden. Als Richtlinie für die Genexpressionsdatensätze sollten drei weitere Szenarien betrachtet werden: 1) große Stichprobenanzahl und große Merkmalsanzahl, 2) große Stichprobenanzahl und kleine Merkmalsanzahl, 3) kleine Stichprobenanzahl und kleine Merkmalsanzahl. Des Weiteren kann der vorgeschlagene Evaluierungsworkflow mit geringen Anpassungen auf Datensätze unterschiedlicher Bereiche angewendet werden. So könnte man zum Beispiel die Dimensionsreduktionsverfahren auf ergänzende Omics-Daten anwenden.
- Obwohl die Reduzierung der Merkmale mit DL-Verfahren die Leistung von nachfolgenden prädiktiven Aufgaben verbessern kann, erschwert es die Interpretation des Vorhersagemodells. Die Ursache für die erschwerte Interpretierbarkeit besteht in der Undurchsichtigkeit und Komplexität der DL-Verfahren. Dies führt dazu, dass sich die Beziehungen zwischen den gelernten Repräsentationen und den beobachteten Variablen nicht ableiten lassen [135]. Demnach ist es wünschenswert, die Undurchsichtigkeit der DL-Verfahren aufzulösen.

# Literatur


[1] Nicholas J. Schork. „Personalized medicine: Time for one-person trials“. In: *Nature* 520.7549 (2015), S. 609–611. ISSN: 1476-4687. DOI: 10.1038/520609a. URL: https://pubmed.ncbi.nlm.nih.gov/25925459/.

[2] Monique G. P. van der Wijst u. a. „An integrative approach for building personalized gene regulatory networks for precision medicine“. In: *Genome medicine* 10.1 (2018), S. 96–111. ISSN: 1756-994X. DOI: 10.1186/s13073-018-0608-4. URL: https://pubmed.ncbi.nlm.nih.gov/30567569/.

[3] Francis S. Collins und Harold Varmus. „A new initiative on precision medicine“. In: *The New England Journal of Medicine* 372.9 (2015), S. 793–795. ISSN: 0028-4793. DOI: 10.1056/NEJMp1500523. URL: https://pubmed.ncbi.nlm.nih.gov/25635347/.

[4] National Research Council (US) Committee on A Framework for Developing a New Taxonomy of Disease. *Toward Precision Medicine: Building a Knowledge Network for Biomedical Research and a New Taxonomy of Disease*. Washington (DC): National Academies Press (US), 2011. DOI: 10.17226/13284. URL: https://pubmed.ncbi.nlm.nih.gov/22536618/.

[5] Ignacio San Segundo-Val und Catalina S. Sanz-Lozano. „Introduction to the Gene Expression Analysis“. In: *Methods in molecular biology (Clifton, N.J.)* 1434 (2016), S. 29–43. ISSN: 1940-6029. DOI: 10.1007/978-1-4939-3652-6_3. URL: https://pubmed.ncbi.nlm.nih.gov/27300529/.

[6] Xiaoyu Zhang u. a. „OmiEmbed: A Unified Multi-Task Deep Learning Framework for Multi-Omics Data“. In: *Cancers* 13.12 (2021), S. 3047–3069. ISSN: 2072-6694. DOI: 10.3390/cancers13123047. URL: https://pubmed.ncbi.nlm.nih.gov/34207255/.

[7] Vladislav Berdunov u. a. „Cost-effectiveness analysis of the Oncotype DX Breast Recurrence Score® test in node-positive early breast cancer“. In: *Journal of medical economics* (2022), S. 1–48. ISSN: 1941-837X. DOI: 10.1080/13696998.2022.2066399. URL: https://pubmed.ncbi.nlm.nih.gov/35416089/.

[8] Christos Sotiriou und Martine J. Piccart. „Taking gene-expression profiling to the clinic: when will molecular signatures become relevant to patient care?“ In: *Nature reviews. Cancer* 7.7 (2007), S. 545–553. ISSN: 1474-175X. DOI: 10.1038/nrc2173. URL: https://pubmed.ncbi.nlm.nih.gov/17585334/.

[9] Athanasios Voulodimos u. a. „Deep Learning for Computer Vision: A Brief Review“. In: *Computational intelligence and neuroscience* 2018 (2018), S. 1–13. ISSN: 1687-5273. DOI: 10.1155/2018/7068349. URL: https://pubmed.ncbi.nlm.nih.gov/29487619/.

[10] Tom Young u. a. „Recent Trends in Deep Learning Based Natural Language Processing [Review Article]“. In: *IEEE Computational Intelligence Magazine* 13.3 (2018), S. 55–75. ISSN: 1556-603X. DOI: 10.1109/MCI.2018.2840738.

[11] Yann LeCun, Yoshua Bengio und Geoffrey Hinton. „Deep learning“. In: *Nature* 521.7553 (2015), S. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://pubmed.ncbi.nlm.nih.gov/26017442/.

[12] Aaron M. Smith u. a. „Standard machine learning approaches outperform deep representation learning on phenotype prediction from transcriptomics data“. In: *BMC bioinformatics* 21.1 (2020), S. 1–18. ISSN: 1471-2105. DOI: 10.1186/s12859-020-3427-8. URL: https://pubmed.ncbi.nlm.nih.gov/32197580/.

[13] Weihua Guo, You Xu und Xueyang Feng. *DeepMetabolism: A Deep Learning System to Predict Phenotype from Genome Sequencing*. Hrsg. von arXiv. 2017. DOI: 10.1101/135574. URL: https://arxiv.org/pdf/1705.03094.

[14] Vahid H. Gazestani und Nathan E. Lewis. „From Genotype to Phenotype: Augmenting Deep Learning with Networks and Systems Biology“. In: *Current opinion in systems biology* 15 (2019), S. 68–73. DOI: 10.1016/j.coisb.2019.04.001. URL: https://pubmed.ncbi.nlm.nih.gov/31777764/.

[15] Richard Bellman. *Dynamic programming*. 1. Aufl. Princeton Landmarks in mathematics. Princeton, NJ: Princeton University Press, 2010. ISBN: 9780691146683.

[16] Zena M. Hira und Duncan F. Gillies. „A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data“. In: *Advances in Bioinformatics* 2015 (2015), S. 1–13. ISSN: 1687-8027. DOI: 10.1155/2015/198363. URL: https://www.hindawi.com/journals/abi/2015/198363/.

[17] Ian Goodfellow, Aaron Courville und Yoshua Bengio. *Deep learning*. 1. Aufl. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016. ISBN: 0262337371. URL: http://www.deeplearningbook.org.

[18] Rizgar Zebari u. a. „A Comprehensive Review of Dimensionality Reduction Techniques for Feature Selection and Feature Extraction“. In: *Journal of Applied Science and Technology Trends* 1.2 (2020), S. 56–70. ISSN: 2708-0757. DOI: 10.38094/jastt1224.

[19] Raia Hadsell, Sumit Chopra und Yann LeCun. „Dimensionality Reduction by Learning an Invariant Mapping“. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2 (2006), S. 1735–1742. DOI: 10.1109/CVPR.2006.100.

[20] Shuicheng Yan u. a. „Graph Embedding and Extensions: A General Framework for Dimensionality Reduction“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.1 (2007), S. 40–51. ISSN: 0162-8828. DOI: 10.1109/tpami.2007.250598.

[21] Jie Tan u. a. „Unsupervised feature construction and knowledge extraction from genome-wide assays of breast cancer with denoising autoencoders“. In: *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* 20 (2015), S. 132–143. DOI: 10.1142/9789814644730_0014. URL: https://pubmed.ncbi.nlm.nih.gov/25592575/.

[22] Aman Gupta, Haohan Wang und Madhavi Ganapathiraju. „Learning structure in gene expression data using deep architectures, with an application to gene clustering“. In: *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (2015), S. 1328–1335. DOI: 10.1109/BIBM.2015.7359871.

[23] Vitor Teixeira, Rui Camacho und Pedro G. Ferreira. „Learning influential genes on cancer gene expression data with stacked denoising autoencoders“. In: *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (2017), S. 1201–1205. DOI: 10.1109/BIBM.2017.8217828.

[24] Padideh Danaee, Reza Ghaeini und DAVID A. HENDRIX. „A Deep Learning Approach for Cancer Detection and Relevant Gene Identificatio“. In: *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* 22 (2017), S. 219–229. DOI: 10.1142/9789813207813_0022. URL: https://pubmed.ncbi.nlm.nih.gov/27896977/.

[25] Eloise Withnell u. a. „XOmiVAE: an interpretable deep learning model for cancer classification using high-dimensional omics data“. In: *Briefings in bioinformatics* 22.6 (2021), S. 1–11. DOI: 10.1093/bib/bbab315. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8575033/.

[26] Dongmei Ai u. a. „Colorectal Cancer Prediction Based on Weighted Gene Co-Expression Network Analysis and Variational Auto-Encoder“. In: *Biomolecules* 10.9 (2020), S. 1–11. ISSN: 2218-273X. DOI: 10.3390/biom10091207. URL: https://pubmed.ncbi.nlm.nih.gov/32825264/.

[27] Zhenxing Wang und Yadong Wang. „Extracting a biologically latent space of lung cancer epigenetics with variational autoencoders“. In: *BMC bioinformatics* 20.18 (2019), S. 1–7. ISSN: 1471-2105. DOI: 10.1186/s12859-019-3130-9. URL: https://pubmed.ncbi.nlm.nih.gov/31760935/.

[28] Gregory P. Way und Casey S. Greene. „Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders“. In: *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* 23 (2018), S. 80–91. DOI: 10.1142/9789813235533_0008. URL: https://pubmed.ncbi.nlm.nih.gov/29218871/.

[29] Shaoliang Peng u. a. „A deep metric learning algorithm for similarity measure of the gene expression profile“. In: *2020 IEEE International Conference on E-health Networking, Application Services (HEALTHCOM)* (2021), S. 1–6. DOI: 10.1109/HEALTHCOM49281.2021.9398919.

[30] Minji Jeon u. a. „ReSimNet: drug response similarity prediction using Siamese neural networks“. In: *Bioinformatics (Oxford, England)* 35.24 (2019), S. 5249–5256. DOI: 10.1093/bioinformatics/btz411. URL: https://pubmed.ncbi.nlm.nih.gov/31116384/.

[31] Xuan Huang, Lei Wu und Yinsong Ye. „A Review on Dimensionality Reduction Techniques“. In: *International Journal of Pattern Recognition and Artificial Intelligence* 33.10 (2019), S. 1–25. DOI: 10.1142/S0218001419500174.

[32] Yehudit Hasin, Marcus Seldin und Aldons Lusis. „Multi-omics approaches to disease“. In: *Genome Biology* 18.1 (2017), S. 83. ISSN: 1474-760X. DOI: 10.1186/s13059-017-1215-1. URL: https://pubmed.ncbi.nlm.nih.gov/28476144/.

[33] C. Kenneth Waters. „Genes Made Molecular“. In: *Philosophy of Science* 61.2 (1994), S. 163–185. ISSN: 0031-8248. DOI: 10.1086/289794.

[34] Francis Crick. „Central dogma of molecular biology“. In: *Nature* 227.5258 (1970), S. 561–563. ISSN: 1476-4687. DOI: 10.1038/227561a0. URL: https://pubmed.ncbi.nlm.nih.gov/4913914/.

[35] Hong-Hai Do, Toralf Kirsten und Erhard Rahm. „Comparative Evaluation of Microarray-based Gene Expression Databases“. In: *BTW 2003–Datenbanksysteme für Business, Technologie und Web* 10 (2003), S. 482–501.

[36] Roberto Fritsche-Neto und Aluízio Borém. „Omics: Opening up the “Black Box” of the Phenotype“. In: *Omics in plant breeding*. Hrsg. von Aluízio Borém und Roberto Fritsche-Neto. Ames, Iowa: John Wiley & Sons Inc, 2014, S. 1–11. ISBN: 9781118820971. DOI: 10.1002/9781118820971.ch1.

[37] Frank W. Albert und Leonid Kruglyak. „The role of regulatory variation in complex traits and disease“. In: *Nature Reviews Genetics* 16.4 (2015), S. 197–212. ISSN: 1471-0064. DOI: 10.1038/nrg3891. URL: https://www.nature.com/articles/nrg3891.

[38] Mark. Schena u. a. „Quantitative monitoring of gene expression patterns with a complementary DNA microarray“. In: *Science (New York, N.Y.)* 270.5235 (1995), S. 467–470. ISSN: 1095-9203. DOI: 10.1126/science.270.5235.467. URL: https://pubmed.ncbi.nlm.nih.gov/7569999/.

[39] Rory Stark, Marta Grzelak und James Hadfield. „RNA sequencing: the teenage years“. In: *Nature reviews. Genetics* 20.11 (2019), S. 631–656. ISSN: 1471-0056. DOI: 10.1038/s41576-019-0150-2. URL: https://pubmed.ncbi.nlm.nih.gov/31341269/.

[40] M. Schena. „Genome analysis with gene expression microarrays“. In: *BioEssays* 18.5 (1996), S. 427–431. ISSN: 1521-1878. DOI: 10.1002/bies.950180513.

[41] David B. Allison u. a. „Microarray data analysis: from disarray to consolidation and consensus“. In: *Nature reviews. Genetics* 7.1 (2006), S. 55–65. ISSN: 1471-0056. DOI: 10.1038/nrg1749. URL: https://pubmed.ncbi.nlm.nih.gov/16369572/.

[42] Zhong Wang, Mark Gerstein und Michael Snyder. „RNA-Seq: a revolutionary tool for transcriptomics“. In: *Nature Reviews Genetics* 10.1 (2009), S. 57–63. ISSN: 1471-0064. DOI: 10.1038/nrg2484. URL: https://pubmed.ncbi.nlm.nih.gov/19015660/.

[43] Christine A. White und Lois A. Salamonsen. „A guide to issues in microarray analysis: application to endometrial biology“. In: *Reproduction (Cambridge, England)* 130.1 (2005), S. 1–13. ISSN: 1470-1626. DOI: 10.1530/rep.1.00685.

[44] Michael L. Metzker. „Sequencing technologies - the next generation“. In: *Nature Reviews Genetics* 11.1 (2010), S. 31–46. ISSN: 1471-0064. DOI: 10.1038/nrg2626. URL: https://pubmed.ncbi.nlm.nih.gov/19997069/.

[45] Qun Pan u. a. „Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing“. In: *Nature Genetics* 40.12 (2008), S. 1413–1415. ISSN: 1546-1718. DOI: 10.1038/ng.259. URL: https://pubmed.ncbi.nlm.nih.gov/18978789/.

[46] Mitchell Guttman u. a. „Ab initio reconstruction of cell type-specific transcriptomes in mouse reveals the conserved multi-exonic structure of lincRNAs“. In: *Nature biotechnology* 28.5 (2010), S. 503–510. ISSN: 1546-1696. DOI: 10.1038/nbt.1633. URL: https://pubmed.ncbi.nlm.nih.gov/20436462/.

[47] Jacob F. Degner u. a. „Effect of read-mapping biases on detecting allele-specific expression from RNA-sequencing data“. In: *Bioinformatics (Oxford, England)* 25.24 (2009), S. 3207–3212. DOI: 10.1093/bioinformatics/btp579. URL: https://pubmed.ncbi.nlm.nih.gov/19808877/.

[48] Henrik Edgren u. a. „Identification of fusion genes in breast cancer by paired-end RNA-sequencing“. In: *Genome Biology* 12.1 (2011), S. 1–13. ISSN: 1474-7596. DOI: 10.1186/gb-2011-12-1-r6. URL: https://pubmed.ncbi.nlm.nih.gov/21247443/.

[49] Laurens van der Maaten, Eric Postma und Jaap van den Herik. „Dimensionality Reduction: A Comparative Review“. In: *Journal of Machine Learning Research* 10.1 (2007), S. 13–49. ISSN: 1533-7928.

[50] Edian F. Franco u. a. „Performance Comparison of Deep Learning Autoencoders for Cancer Subtype Detection Using Multi-Omics Data“. In: *Cancers* 13.9 (2021), S. 1–17. ISSN: 2072-6694. DOI: 10.3390/cancers13092013. URL: https://pubmed.ncbi.nlm.nih.gov/33921978/.

[51] Arul kumar Venugopal. „A Survey on Dimensionality Reduction Technique“. In: *International Journal of Emerging Trends & Technology in Computer Science* 3.6 (2014), S. 36–41. ISSN: 2278-6856. URL: https://www.ijettcs.org/Volume3Issue6/IJETTCS-2014-11-12-33.pdf.

[52] Benyamin Ghojogh, Fakhri Karray und Mark Crowley. *On Manifold Hypothesis: Hypersurface Submanifold Embedding Using Osculating Hyperspheres*. Hrsg. von arXiv. 2022. DOI: 10.48550/arXiv.2202.01619. URL: https://arxiv.org/pdf/2202.01619.

[53] Keinosuke Fukunaga und David. R. Olsen. „An Algorithm for Finding Intrinsic Dimensionality of Data“. In: *IEEE Transactions on Computers* C-20.2 (1971), S. 176–183. ISSN: 1557-9956. DOI: 10.1109/T-C.1971.223208.

[54] Kevin P. Murphy. *Machine learning - a probabilistic perspective*. 1. Aufl. Bd. 1. Adaptive Computation and Machine Learning series. Cambridge, Massachusetts: MIT Press, 2012. ISBN: 9780262304320.

[55] Karl Pearson. „LIII. On lines and planes of closest fit to systems of points in space“. In: *Philosophical Magazine* 2 (1901), S. 559–572. DOI: 10.1080/14786440109462720.

[56] Hervé Abdi und Lynne J. Williams. „Principal component analysis“. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2.4 (2010), S. 433–459. ISSN: 19395108. DOI: 10.1002/wics.101.

[57] Teuvo Kohonen. „Self-organized formation of topologically correct feature maps“. In: *Biological Cybernetics* 43.1 (1982), S. 59–69. ISSN: 1432-0770. DOI: 10.1007/BF00337288. URL: https://link.springer.com/article/10.1007/BF00337288.

[58] Warren S. McCulloch und Walter Pitts. „A logical calculus of the ideas immanent in nervous activity“. In: *The bulletin of mathematical biophysics* 5.4 (1943), S. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://link.springer.com/article/10.1007/BF02478259.

[59] Kenji Suzuki. *Artificial Neural Networks - Methodological Advances and Biomedical Applications*. 1. Aufl. IntechOpen, 2011. ISBN: 9789533072432.

[60] Oded Z. Maimon und Lior Rokach. *Data Mining and Knowledge Discovery Handbook*. 1. Aufl. SpringerLink Bücher. Boston, MA: Springer US, 2005. ISBN: 9780387254654. DOI: 10.1007/b107408.

[61] David E. Rumelhart, Geoffrey E. Hinton und Ronald J. Williams. „Learning representations by back-propagating errors“. In: *Nature* 323.6088 (1986), S. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: https://www.nature.com/articles/323533a0.

[62] Ayon Dey. „Machine Learning Algorithms : A Review“. In: *International Journal of Computer Science and Information Technologies* 7.3 (2016), S. 1174–1179. ISSN: 0975-9646. URL: http://ijcsit.com/docs/Volume%207/vol7issue3/ijcsit2016070332.pdf.

[63] Geoffrey Hinton und Terrence J. Sejnowski. *Unsupervised Learning: foundations of neural computation*. 1. Aufl. Cambridge, Massachusetts: The MIT Press, 1999. ISBN: 978-0262581684. DOI: 10.7551/mitpress/7011.001.0001.

[64] Ons Aouedi, Kandaraj Piamrat und Dhruvjyoti Bagadthey. „A Semi-supervised Stacked Autoencoder Approach for Network Traffic Classification“. In: *2020 IEEE 28th International Conference on Network Protocols (ICNP)* (2020), S. 1–6. DOI: 10.1109/ICNP49622.2020.9259390.

[65] Nikolaus Kriegeskorte und Tal Golan. „Neural network models and deep learning“. In: *Current biology : CB* 29.7 (2019), S. 231–236. DOI: 10.1016/j.cub.2019.02.034.

[66] Peng Liu, Peijun Zheng und Ziyu Chen. „Deep Learning with Stacked Denoising Auto-Encoder for Short-Term Electric Load Forecasting“. In: *Energies* 12.12 (2019), S. 1–15. DOI: 10.3390/en12122445.

[67] Teuvo Kohonen. „The self-organizing map“. In: *Proceedings of the IEEE* 78.9 (1990), S. 1464–1480. DOI: 10.1109/5.58325.

[68] Yann Lecun. „PhD thesis: Modeles connexionnistes de l'apprentissage (connectionist learning models)“. Diss. Universite P. et M. Curie, 1987.

[69] Pascal Vincent u. a. „Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion“. In: *Journal of Machine Learning Research* 11.110 (2010), S. 3371–3408. ISSN: 1533-7928.

[70] Pierre Baldi und Kurt Hornik. „Neural networks and principal component analysis: Learning from examples without local minima“. In: *Neural Networks* 2.1 (1989), S. 53–58. ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90014-2.

[71] Pascal Vincent u. a. „Extracting and composing robust features with denoising autoencoders“. In: *Proceedings of the 25th International Conference on Machine Learning* (2008), S. 1096–1103. DOI: 10.1145/1390156.1390294.

[72] Yoshua Bengio u. a. „Greedy layer-wise training of deep networks“. In: *Proceedings of the 19th International Conference on Neural Information Processing Systems* 19 (2006), S. 153–160. DOI: 10.7551/mitpress/7503.003.0024.

[73] Hugo Larochelle u. a. „Exploring Strategies for Training Deep Neural Networks“. In: *Journal of Machine Learning Research* 10 (2009), S. 1–40. ISSN: 1533-7928. URL: http://jmlr.org/papers/v10/larochelle09a.html.

[74] Marc'Aurelio Ranzato u. a. „Efficient Learning of Sparse Representations with an Energy-Based Model“. In: *Proceedings of the 19th International Conference on Neural Information Processing Systems* 19 (2006), S. 1137–1144. DOI: 10.7551/mitpress/7503.003.0147.

[75] Junyuan Xie, Linli Xu und Enhong Chen. „Image Denoising and Inpainting with Deep Neural Networks“. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems* 1 (2012), S. 341–349.

[76] Geoffrey E. Hinton und Ruslan R. Salakhutdinov. „Reducing the dimensionality of data with neural networks“. In: *Science (New York, N.Y.)* 313.5786 (2006), S. 504–507. ISSN: 1095-9203. DOI: 10.1126/science.1127647. URL: https://pubmed.ncbi.nlm.nih.gov/16873662/.

[77] Diederik P. Kingma und Max Welling. *Auto-Encoding Variational Bayes*. Hrsg. von arXiv. 2014. DOI: 10.48550/arXiv.1312.6114. URL: https://arxiv.org/pdf/1312.6114.

[78] Diederik P. Kingma und Max Welling. „An Introduction to Variational Autoencoders“. In: *Foundations and Trends® in Machine Learning* 12.4 (2019), S. 307–392. ISSN: 1935-8237. DOI: 10.1561/2200000056.

[79] Jane Bromley u. a. „Signature verification using a Siamese time delay neural network“. In: *International Journal of Pattern Recognition and Artificial Intelligence* 7.4 (1993), S. 669–688. DOI: 10.1142/S0218001493000339.

[80] Gregory Koch, Richard Zemel und Ruslan Salakhutdinov. „Siamese Neural Networks for One-shot Image Recognition“. In: *Proceedings of the 32nd International Conference on Machine Learning* 37.1 (2015), S. 1–8. URL: www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf.

[81] Sumit Chopra, Raia Hadsell und Yann LeCun. „Learning a Similarity Metric Discriminatively, with Application to Face Verification“. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 1 (2005), S. 539–546. DOI: 10.1109/CVPR.2005.202.

[82] Florian Schroff, Dmitry Kalenichenko und James Philbin. „FaceNet: A Unified Embedding for Face Recognition and Clustering“. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition* 1 (2015), S. 815–823. DOI: 10.1109/CVPR.2015.7298682.

[83] Aurélien Bellet, Amaury Habrard und Marc Sebban. *A Survey on Metric Learning for Feature Vectors and Structured Data*. Hrsg. von arXiv. 2014. DOI: 10.48550/arXiv.1306.6709. URL: https://arxiv.org/pdf/1306.6709.

[84] Jason V. Davis u. a. „Information-theoretic metric learning“. In: *Proceedings of the 24th international conference on Machine learning* (2007), S. 209–216. DOI: 10.1145/1273496.1273523.

[85] Kilian Q. Weinberger und Lawrence K. Saul. „Distance metric learning for large margin nearest neighbor classification“. In: *The Journal of Machine Learning Research* 10 (2009), S. 207–244.

[86] Lilei Zheng u. a. „Triangular similarity metric learning for face verification“. In: *11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)* 1 (2015), S. 1–7. DOI: 10.1109/FG.2015.7163085.

[87] Mohammad Shorfuzzaman und M. Shamim Hossain. „MetaCOVID: A Siamese neural network framework with contrastive loss for n-shot diagnosis of COVID-19 patients“. In: *Pattern recognition* 113 (2021), S. 1–11. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2020.107700. URL: https://pubmed.ncbi.nlm.nih.gov/33100403/.

[88] Ka Yee Yeung und Walter L. Ruzzo. „Principal component analysis for clustering gene expression data“. In: *Bioinformatics* 17.9 (2001), S. 763–774. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/17.9.763. URL: https://pubmed.ncbi.nlm.nih.gov/11590094/.

[89] Rasool Fakoor u. a. „Using deep learning to enhance cancer diagnosis and classication“. In: *Proceedings of the 30th International Conference on Machine Learning* 28.1 (2013), S. 3937–3949.

[90] Xiaoyu Zhang u. a. „Integrated Multi-omics Analysis Using Variational Autoencoders: Application to Pan-cancer Classification“. In: *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (2019), S. 765–769. DOI: 10.1109/BIBM47256.2019.8983228.

[91] Alexander J. Titus, Carly A. Bobak und Brock C. Christensen. „A New Dimension of Breast Cancer Epigenetics - Applications of Variational Autoencoders with DNA Methylation“. In: *Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2018)* 3 (2018), S. 140–145. DOI: 10.5220/0006636401400145.

[92] Ayse B. Dincer, Joseph D. Janizek und Su-In Lee. „Adversarial deconfounding autoencoder for learning robust gene expression embeddings“. In: *Bioinformatics (Oxford, England)* 36.Suppl_2 (2020), S. 573–582. DOI: 10.1093/bioinformatics/btaa796. URL: https://pubmed.ncbi.nlm.nih.gov/33381842/.

[93] Christoph Bartenhagen u. a. „Comparative study of unsupervised dimension reduction techniques for the visualization of microarray gene expression data“. In: *BMC bioinformatics* 11.1 (2010), S. 1–11. ISSN: 1471-2105. DOI: 10.1186/1471-2105-11-567. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2998530/.

[94] Micheal O. Arowolo u. a. „A Comparative Analysis of Feature Extraction Methods for Classifying Colon Cancer Microarray Data“. In: *EAI Endorsed Transactions on Scalable Information Systems* 4.14 (2017), S. 1–6. ISSN: 2032-9407. DOI: 10.4108/eai.25-9-2017.153147.

[95] David Pratella u. a. „A Survey of Autoencoder Algorithms to Pave the Diagnosis of Rare Diseases“. In: *International journal of molecular sciences* 22.19 (2021), S. 1–14. ISSN: 1422-0067. DOI: 10.3390/ijms221910891. URL: https://pubmed.ncbi.nlm.nih.gov/34639231/.

[96] Alexander Lachmann u. a. „Massive mining of publicly available RNA-seq data from human and mouse“. In: *Nature Communications* 9.1 (2018), S. 1–10. ISSN: 2041-1723. DOI: 10.1038/s41467-018-03751-6. URL: https://pubmed.ncbi.nlm.nih.gov/29636450/.

[97] Markus Kreuz u. a. „ProstaTrend-A Multivariable Prognostic RNA Expression Score for Aggressive Prostate Cancer“. In: *European urology* 78.3 (2020), S. 452–459. DOI: 10.1016/j.eururo.2020.06.001. URL: https://pubmed.ncbi.nlm.nih.gov/32631745/.

[98] Ron Edgar, Michael Domrachev und Alex E. Lash. „Gene Expression Omnibus: NCBI gene expression and hybridization array data repository“. In: *Nucleic Acids Research* 30.1 (2002), S. 207–210. ISSN: 0305-1048. DOI: 10.1093/nar/30.1.207. URL: https://pubmed.ncbi.nlm.nih.gov/11752295/.

[99] Yuichi Kodama, Martin Shumway und Rasko Leinonen. „The Sequence Read Archive: explosive growth of sequencing data“. In: *Nucleic Acids Research* 40.1 (2012), S. 54–56. ISSN: 0305-1048. DOI: 10.1093/nar/gkr854. URL: https://pubmed.ncbi.nlm.nih.gov/22009675/.

[100] Mike Folk, Robert E. McGrath und Nancy Yeager. „HDF: an update and future directions“. In: *IEEE 1999 International Geoscience and Remote Sensing Symposium. IGARSS'99 (Cat. No.99CH36293)* 5 (1999), S. 273–275. DOI: 10.1109/IGARSS.1999.773469.

[101] Kay H. Brodersen u. a. „The Balanced Accuracy and Its Posterior Distribution“. In: *2010 20th International Conference 2010* 1 (2010), S. 3121–3124. DOI: 10.1109/ICPR.2010.764.

[102] Benjamin M. Bolstad u. a. „A comparison of normalization methods for high density oligonucleotide array data based on variance and bias“. In: *Bioinformatics* 19.2 (2003), S. 185–193. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/19.2.185. URL: https://pubmed.ncbi.nlm.nih.gov/12538238/.

[103] Sean Simmons u. a. „Discovering What Dimensionality Reduction Really Tells Us About RNA-Seq Data“. In: *Journal of Computational Biology* 22.8 (2015), S. 715–728. ISSN: 1066-5277. DOI: 10.1089/cmb.2015.0085. URL: https://pubmed.ncbi.nlm.nih.gov/26098139/.

[104] W. Evan Johnson, Cheng Li und Ariel Rabinovic. „Adjusting batch effects in microarray expression data using empirical Bayes methods“. In: *Biostatistics* 8.1 (2007), S. 118–127. ISSN: 1465-4644. DOI: 10.1093/biostatistics/kxj037. URL: https://pubmed.ncbi.nlm.nih.gov/16632515/.

[105] Jeffrey T. Leek u. a. „The sva package for removing batch effects and other unwanted variation in high-throughput experiments“. In: *Bioinformatics (Oxford, England)* 28.6 (2012), S. 882–883. DOI: 10.1093/bioinformatics/bts034. URL: https://pubmed.ncbi.nlm.nih.gov/22257669/.

[106] Fabian Pedregosa u. a. „Scikit-learn: Machine learning in Python“. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830. ISSN: 1533-7928. URL: https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html.

[107] Yu Chen, Byungkyu Park und Kyungsook Han. „Qualitative reasoning of dynamic gene regulatory interactions from gene expression data“. In: *BMC Genomics* 11.4 (2010), S. 1–14. DOI: 10.1186/1471-2164-11-S4-S14.

[108] Bradley J. Erickson u. a. „Toolkits and Libraries for Deep Learning“. In: *Journal of Digital Imaging* 30.4 (2017), S. 400–405. ISSN: 1618-727X. DOI: 10.1007/s10278-017-9965-6. URL: https://pubmed.ncbi.nlm.nih.gov/28315069/.

[109] Mark A. Kramer. „Nonlinear principal component analysis using autoassociative neural networks“. In: *AIChE Journal* 37.2 (1991), S. 233–243. ISSN: 0001-1541. DOI: 10.1002/aic.690370209.

[110] Nitish Srivastava u. a. „Dropout: a simple way to prevent neural networks from overfitting“. In: *Journal of Machine Learning Research* 15 (2014), S. 1929–1958. ISSN: 1533-7928.

[111] Thomas Cover und Peter Hart. „Nearest neighbor pattern classification“. In: *IEEE Transactions on Information Theory* 13.1 (1967), S. 21–27. ISSN: 1557-9654. DOI: 10.1109/TIT.1967.1053964.

[112] P. Chaitra und Saravana R. Kumar. „A review of multi-class classification algorithms“. In: *International Journal of Pure and Applied Mathematics* 118.14 (2018), S. 17–26.

[113] Corinna Cortes und Vladimir Vapnik. „Support-vector networks“. In: *Machine Learning* 20.3 (1995), S. 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018. URL: https://link.springer.com/article/10.1007%2FBF00994018.

[114] Edwin Raczko und Bogdan Zagajewski. „Comparison of support vector machine, random forest and neural network classifiers for tree species classification on airborne hyperspectral APEX images“. In: *European Journal of Remote Sensing* 50.1 (2017), S. 144–154. DOI: 10.1080/22797254.2017.1299557.

[115] Sveinung Attestog u. a. „Mixed Fault Classification of Sensorless PMSM Drive in Dynamic Operations Based on External Stray Flux Sensors“. In: *Sensors* 22.3 (2022), S. 1–19. ISSN: 1424-8220. DOI: 10.3390/s22031216. URL: https://pubmed.ncbi.nlm.nih.gov/35161959/.

[116] David W. Hosmer, Rodney X. Sturdivant und Stanley Lemeshow. *Applied logistic regression*. 3. Aufl. Wiley series in probability and statistics. Hoboken, New Jersey: John Wiley & Sons, Ltd, 2013. ISBN: 978-0-470-58247-3.

[117] Leo Breiman. „Random Forests“. In: *Machine Learning* 45.1 (2001), S. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. URL: https://link.springer.com/article/10.1023/A:1010933404324.

[118] Mohammadmehdi Saberioon u. a. „Comparative Performance Analysis of Support Vector Machine, Random Forest, Logistic Regression and k-Nearest Neighbours in Rainbow Trout (Oncorhynchus Mykiss) Classification Using Image-Based Features“. In: *Sensors* 18.4 (2018), S. 10–25. ISSN: 1424-8220. DOI: 10.3390/s18041027.

[119] Ramón Díaz-Uriarte und Sara Alvarez de Andrés. „Gene selection and classification of microarray data using random forest“. In: *BMC bioinformatics* 7.1 (2006), S. 1–13. ISSN: 1471-2105. DOI: 10.1186/1471-2105-7-3. URL: https://link.springer.com/article/10.1186/1471-2105-7-3.

[120] Mohammadreza Sheykhmousa u. a. „Support Vector Machine Versus Random Forest for Remote Sensing Image Classification: A Meta-Analysis and Systematic Review“. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 13 (2020), S. 6308–6325. DOI: 10.1109/JSTARS.2020.3026724.

[121] Mohammad Hossin und Md Nasir Sulaiman. „A Review on Evaluation Metrics for Data Classification Evaluations“. In: *International Journal of Data Mining & Knowledge Management Process* 5.2 (2015), S. 1–11. ISSN: 2231-007X. DOI: 10.5121/ijdkp.2015.5201.

[122] Laurens van der Maaten und Geoffrey Hinton. „Visualizing Data using t-SNE“. In: *Journal of Machine Learning Research* 9 (2008), S. 2579–2605. ISSN: 1533-7928. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[123] Geoffrey Hinton und Sam Roweis. „Stochastic Neighbor Embedding“. In: *Advances in Neural Information Processing Systems* 15.1 (2003), S. 1–8. ISSN: 1049-5258. URL: https://cs.nyu.edu/~roweis/papers/sne_final.pdf.

[124] Muhammad Uzair und Noreen Jamil. „Effects of Hidden Layers on the Efficiency of Neural networks“. In: *2020 IEEE 23rd International Multitopic Conference (INMIC)*. 2020, S. 1–6. DOI: 10.1109/INMIC50486.2020.9318195.

[125] Priya Goyal u. a. *Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour*. Hrsg. von arXiv. 2017. URL: https://arxiv.org/pdf/1706.02677.

[126] Elad Hoffer, Itay Hubara und Daniel Soudry. „Train longer, generalize better: closing the generalization gap in large batch training of neural networks“. In: *Advances in Neural Information Processing Systems 30* 30.1 (2017), S. 1–11.

[127] Nitish S. Keskar u. a. *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*. Hrsg. von arXiv. 2016. URL: https://arxiv.org/pdf/1609.04836.

[128] Mu Li u. a. „Efficient mini-batch training for stochastic optimization“. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), S. 661–670. DOI: 10.1145/2623330.2623612.

[129] Matthew D. Zeiler. *Adadelta: An Adaptive Learning Rate Method*. Hrsg. von arXiv. 2012. URL: https://arxiv.org/pdf/1212.5701.

[130] Xuan Li u. a. „Guided autoencoder for dimensionality reduction of pedestrian features“. In: *Applied Intelligence* 50.12 (2020), S. 4557–4567. DOI: 10.1007/s10489-020-01813-1. URL: https://link.springer.com/article/10.1007/s10489-020-01813-1.

[131] Meenal V. Narkhede, Prashant P. Bartakke und Mukul S. Sutaone. „A review on weight initialization strategies for neural networks“. In: *Artificial Intelligence Review* 55.1 (2022), S. 291–322. ISSN: 1573-7462. DOI: 10.1007/s10462-021-10033-z. URL: https://link.springer.com/article/10.1007/s10462-021-10033-z.

[132] Michele Sebag, Victor Berger und Michèle Sebag. *Variational Auto-Encoder: not all failures are equal*. Hrsg. von arXiv. 2020. DOI: 10.48550/arXiv.2003.01972. URL: https://arxiv.org/pdf/2003.01972.

[133] Quentin Fournier und Daniel Aloise. „Empirical Comparison between Autoencoders and Traditional Dimensionality Reduction Methods“. In: *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. 2019, S. 211–214. DOI: 10.1109/AIKE.2019.00044.

[134] Qiwen Hu und Casey S. Greene. „Parameter tuning is a key part of dimensionality reduction via deep variational autoencoders for single cell RNA transcriptomics“. In: *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* 24 (2019), S. 362–373. URL: https://pubmed.ncbi.nlm.nih.gov/30963075/.

[135] Moritz Hess, Maren Hackenberg und Harald Binder. „Exploring generative deep learning for omics data using log-linear models“. In: *Bioinformatics (Oxford, England)* 36.20 (2020), S. 5045–5053. DOI: 10.1093/bioinformatics/btaa623. URL: https://pubmed.ncbi.nlm.nih.gov/32647888/.

# Erklärung

Ich versichere, dass ich die vorliegende Arbeit mit dem Thema:

*„Evaluierung von Deep-Learning-Methoden zur Generierung von Embeddings aus Genexpressionsdaten“*

selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann.

Leipzig, den 09.06.2022

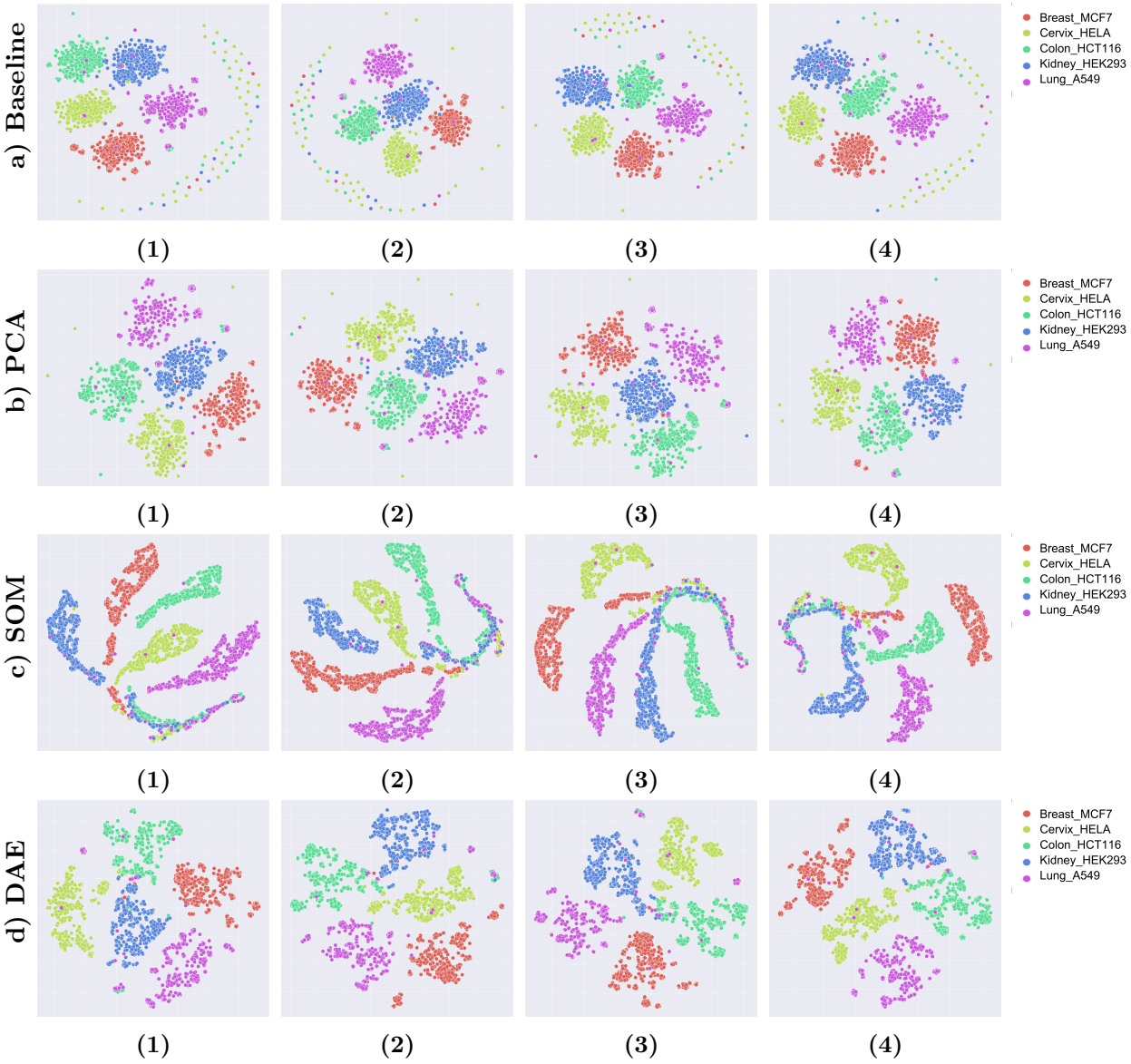
_____

Leonie Preker

# Anhang

## A ARCHS4-Datensatz

### A.1 Zweidimensionale Visualisierung der erlernten Repräsentationen des ARCHS4-Datensatzes mittels t-distributed Stochastic Neighbor Embedding



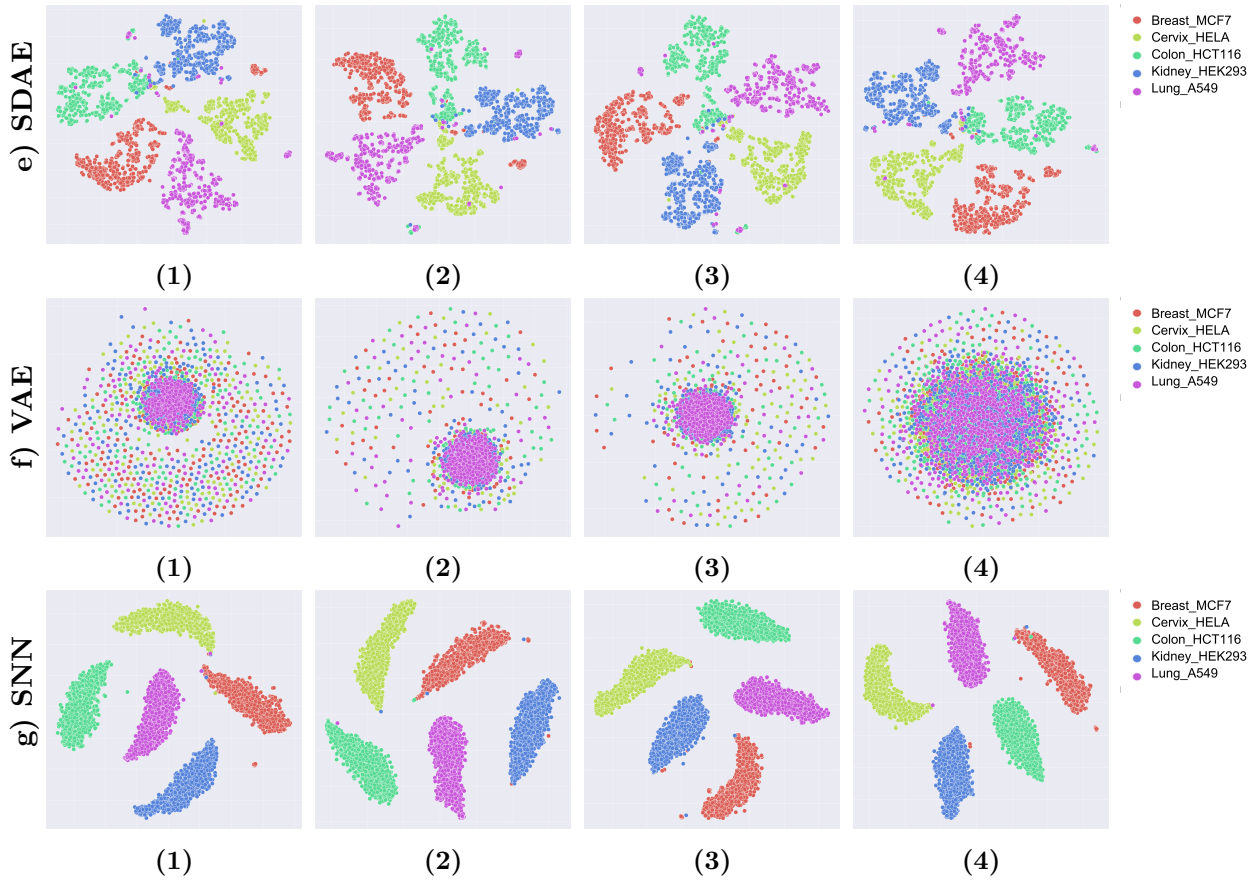
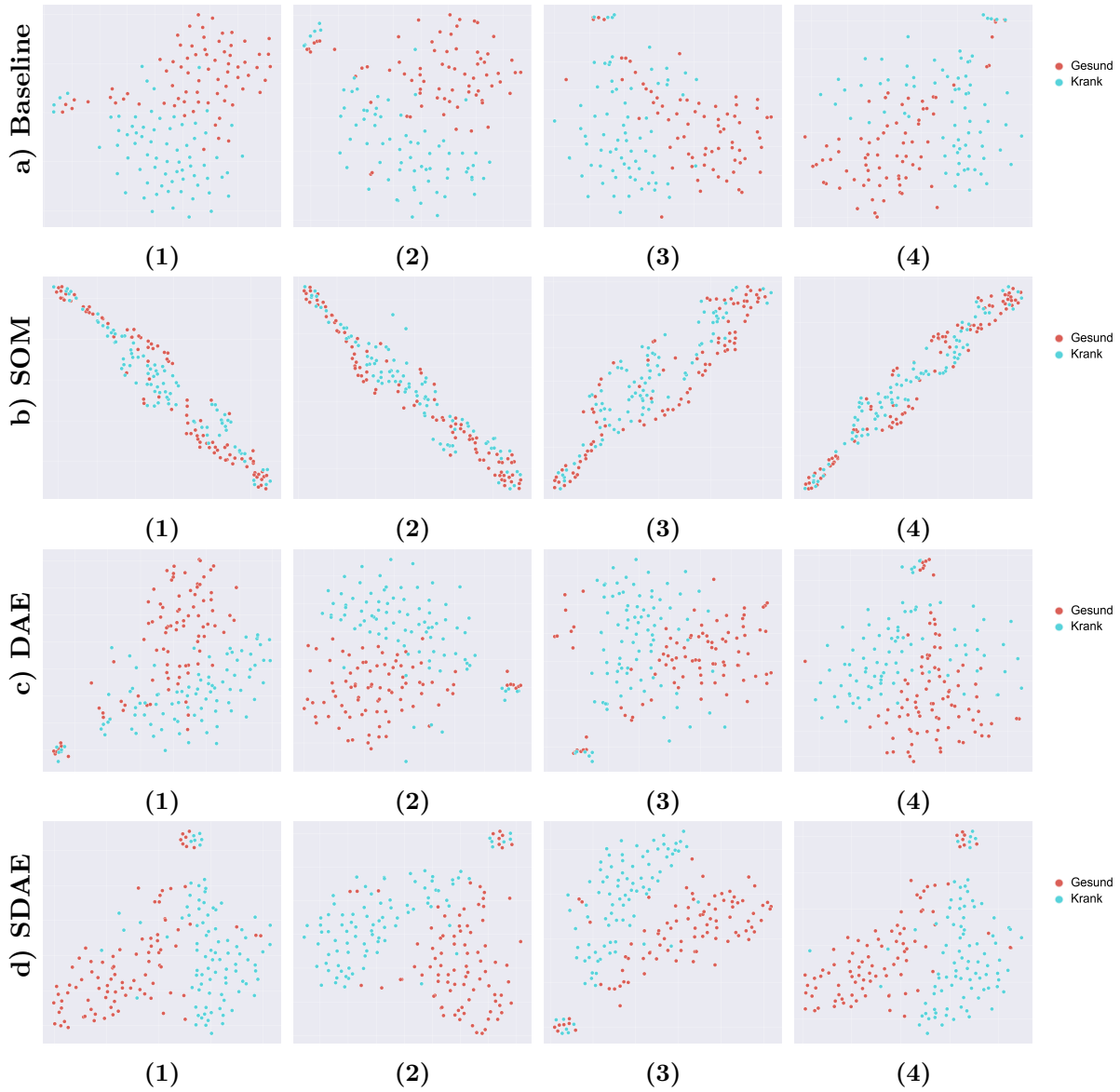



**Abbildung 24:** Zweidimensionale Visualisierung der erlernten Repräsentationen des ARCHS4-Datensatzes mit t-SNE. Die Darstellungen der Zeilen a-g stellen die Visualisierungen der Dimensionsreduktionsmodelle (PCA, SOM, DAE, SDAE, VAE, SNN) sowie ein Baseline-Modell dar. Die Darstellungen 1-4 der einzelnen Modelle visualisieren die generierten Embeddings der Durchläufe 2-5 der äußeren Kreuzvalidierung. Die mit t-SNE generierten 2D-Merkmale sind in einem Streudiagramm dargestellt. Die x-Achse repräsentiert das t-SNE-Merkmal 1 und die y-Achse repräsentiert das t-SNE-Merkmal 2. Die Achsenskala wurde in dieser Arbeit nicht dargestellt, da diese keine Relevanz besitzt [122]. Jeder Punkt einer Farbe repräsentiert eine Probe der entsprechenden Klasse, wie die Legende zeigt. T-SNE = t-distributed Stochastic Neighbor Embedding; PCA = Hauptkomponentenanalyse; SOM = Selbstorganisierende Karte; DAE = Deep Autoencoder; SDAE = Stacked Denoising Autoencoder; VAE = Variational Autoencoder; SNN = siamesisches neuronales Netz.

# B Prostatakrebs-Datensatz

## B.1 Zweidimensionale Visualisierung der erlernten Repräsentationen des Prostatakrebs-Datensatzes mittels t-distributed Stochastic Neighbor Embedding



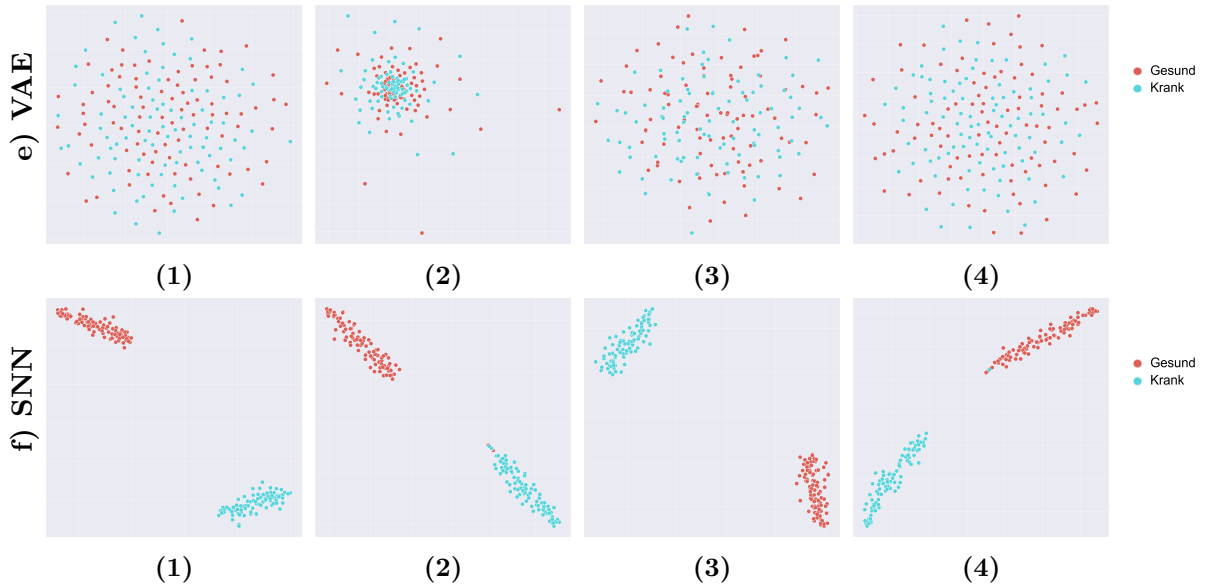



**Abbildung 25:** Zweidimensionale Visualisierung der erlernten Repräsentationen des Prostatakrebs-Datensatzes mit t-SNE. Die Darstellungen der Zeilen a-f stellen die Visualisierungen der Dimensionsreduktionsmodelle (SOM, DAE, SDAE, VAE, SNN) sowie ein Baseline-Modell dar. Die Darstellungen 1-4 der einzelnen Modelle visualisieren die generierten Embeddings der Durchläufe 2-5 der äußeren Kreuzvalidierung. Die mit t-SNE generierten 2D-Merkmale sind in einem Streudiagramm dargestellt. Die x-Achse repräsentiert das t-SNE-Merkmal 1 und die y-Achse repräsentiert das t-SNE-Merkmal 2. Die Achsenskala wurde in dieser Arbeit nicht dargestellt, da diese keine Relevanz besitzt [122]. Jeder Punkt einer Farbe repräsentiert eine Probe der entsprechenden Klasse, wie die Legende zeigt. Wie bereits in Abschnitt 4.2 erläutert, konnte keine PCA angewandt werden. T-SNE = t-distributed Stochastic Neighbor Embedding; PCA = Hauptkomponentenanalyse; SOM = Selbstorganisierende Karte; DAE = Deep Autoencoder; SDAE = Stacked Denoising Autoencoder; VAE = Variational Autoencoder; SNN = siamesisches neuronales Netz.