INSTITUT FÜR INFORMATIK
Algorithmische Bioinformatik

Universitätsstr. 1 D–40225 Düsseldorf

Heinrich Heine Universität Düsseldorf

Klassifizierung von Krebsarten durch Tumor-DNA mittels Support-Vektor-Maschinen und künstlichen neuronalen Netzen

Roman Jumatov

Bachelorarbeit

Beginn der Arbeit: 02. April 2020
Abgabe der Arbeit: 14. August 2020
Gutachter: Univ.-Prof. Dr. G. Klau
Univ.-Prof. Dr. S. Harmeling

Zusammenfassung

In der Krebsforschung mangelt es häufig an Verfahren die vielen genomischen Daten für eine zuverlässige Bestimmung des Krebstypen nutzen zu können. Über eine öffentlich zugängliche Datenbank für Onkogenomik namens *cBioPortal* erfolgte eine Datenerhebung von mehr als 6000 Stichproben für 28 verschiedene Krebstypen. In dieser Arbeit wurde eine Support-Vektor-Maschine und ein künstlich neuronales Netzwerk implementiert, um den Krebstypen anhand der aus Tumor-DNA erhobenen Daten vorauszusagen. Bisher betrug die Genauigkeit einer Vorhersage der Krebsart über Tumor-DNA 88.1%, welche lediglich von einer Support-Vektor-Maschine erreicht wurde. In dieser Arbeit haben sowohl die Support-Vektor-Maschine als auch ein künstliches neuronales Netzwerk unter Verwendung einer Teildatenmenge eine Genauigkeit von 90% erreicht. Das künstliche neuronale Netz eignet sich somit zur Krebsbestimmung und birgt aufgrund vielseitiger Gestaltungsmöglichkeiten das Potenzial, bessere Ergebnisse erzielen zu können.

Inhaltsverzeichnis

1 Einleitung **1**

2 Theorien und Konzepte **2**

2.1 Machine Learning 2

2.2 Aufbau der Daten 3

2.3 Support Vector Machine 3

2.4 Artificial Neural Network 5

2.4.1 Neuronen 6

2.4.2 Aktivierungsfunktionen 7

2.4.3 Verlustfunktion 8

2.4.4 Backpropagation 8

2.4.5 Optimierer 8

2.5 Evaluationsmethodik 9

2.5.1 Accuracy 9

2.5.2 Confusion Matrix 9

2.5.3 Precision, Recall und F1-Score 10

2.5.4 ROC-Curve und AUC-Score 11

2.5.5 Kreuzvalidierung 12

3 Methodik **13**

3.1 Train- und Testsplit 13

3.2 LinearSVC 13

3.2.1 Squared Hinge Loss 14

3.2.2 Regularisierungsparameter 14

3.3 PyTorch 14

3.3.1 Schichtenarchitektur 14

3.3.2 SELU 15

3.3.3 LogSoftmax 16

3.3.4 Negative Log Likelihood Loss 17

3.3.5 Adabound 17

3.4 Feature Selection 18

3.4.1 Extra Tree Classifier 19

3.4.2 Recursive Feature Elimination 20

3.4.3 RFE mit Kreuzvalidierung 20

4 Auswertung **21**

4.1 Evaluierung am ganzen Set 21

4.2 Top-50 und Top-100 Feature Sets 27

4.3 Optimale Anzahl an Features 27

5 Fazit **29**

5.1 Zusammenfassung 29

5.2 Ausblick 30

A SVM Tuning **31**

B ANN Tuning **31**

C Ergänzung zur Evaluation **33**

Literaturverzeichnis **37**

1 Einleitung

In der heutigen medizinischen Forschung ist eine frühzeitige Bestimmung der Krebsart für den weiteren Behandlungsverlauf von großer Bedeutung. Jedoch sind die genomischen Veränderungen in den Tumorzellen äußerst komplex. Selbst innerhalb einer Krebsart zeigen sich große Unterschiede in den Tumorgenotypen eines jeden Individuums. Machine-Learning-Verfahren können dafür genutzt werden eine Vielzahl von Genveränderungen in mehreren Tumorproben bei der Bestimmung der Krebsart zu berücksichtigen.

Anlass dieser Arbeit ist die, in [Soh+17], vorgenommene Bewertung des Potenzials der Verwendung von Mutationen aus Tumor-DNA zur Identifizierung von Krebsarten über maschinelle Verfahren. Ferner wurden alle für die Klassifizierung benötigten Genom-Daten aus der Web-Plattform *cBioPortal*, einer öffentlich zugänglichen Datenbank für onkogenomische Forschung, erhoben. Die in dem Portal zur Verfügung gestellten Stichproben von erkrankten Patienten wurden dafür verwendet, um *Somatic Point Mutations* (SPM) und *Copy Number Alterations* (CNA) für ausgewählte Krebstypen zu entnehmen. Somatic Point Mutations beschreiben erworbene, nicht vererbbare Mutationen, wohingegen Copy Number Alterations Änderungen der Chromosomenstruktur beschreiben, welche zu veränderten Anzahlen der Kopien von DNA-Abschnitten führen und demnach vererbbar sind [Ber+10]. Die aus den Mutationen resultierenden Veränderungen können eine Zelltransformation hervorrufen, welche zum Tumor und letztlich zu Krebs führen kann. Bei den in [Soh+17] verwendeten Techniken zur Klassifizierung von Krebstypen handelt es sich um *Support Vector Machine*, *Logistic Regression* und *Random Forest*, wobei die Support Vector Machine die besten Resultate erzielt hat.

Das Ziel dieser Arbeit ist es, die Klassifizierung von Krebsarten erneut mittels einer Support Vector Machine und einem künstlichen neuronalen Netzwerk umzusetzen. Die Support Vector Machine wurde für direkte Vergleiche mit dem künstlichen neuronalen Netzwerk in Python reimplementiert. Ebenso stellt sich die Frage, inwiefern die bisherigen Ergebnisse mit einer Support Vector Machine übertroffen werden können. Darüber hinaus ist die Bestimmung mehrerer infrage kommender Krebsarten möglich. Beide Verfahren wurden getestet, bewertet und sowohl miteinander als auch mit Ergebnissen aus [Soh+17] verglichen.

Den nächsten Teil der Arbeit bilden Theorien und Konzepte hinsichtlich des Grundbegriffs *Machine Learning*, den verwendeten Daten, Verfahren und Bewertungsmetriken. Es folgen Methodiken, die Entscheidungsfragen bezüglich der Implementierung klären. Die Evaluation stellt den vorletzten Abschnitt dar, worin die in der Theorie erläuterten Techniken zur Bewertung angewandt werden. Im letzten Kapitel der Arbeit findet sich ein Fazit in Anbetracht der zu Beginn definierten Zielsetzung. Des Weiteren erfolgt ein Ausblick bezüglich möglicher Erweiterungen.

2 Theorien und Konzepte

Im Folgenden werden die für das Verständnis notwendigen theoretischen Grundlagen dargelegt. Hierbei handelt es sich um die konzeptionelle Einführung des Begriffs Machine Learning sowie die Beschreibung der für diese Arbeit relevanten Techniken. Fachwörter werden vorzugsweise in der Fachsprache Englisch verwendet.

2.1 Machine Learning

Machine Learning ist ein Teilgebiet der künstlichen Intelligenz und befasst sich mit Verfahren, welche nach einem gewissen Performance-Kriterium optimiert werden. Das daraus resultierende Modell kann folglich Voraussagen treffen und bzw. oder neue Erkenntnisse aus den Daten liefern [vgl. Alp20, S. 3]. Üblicherweise durchläuft ein Modell eine Trainings- und Testphase, in denen jeweils eine Teildatenmenge der zur Verfügung stehenden Daten verwendet wird. In der Trainingsphase *lernt* das Modell aus den Trainingsdaten, welches in einem weiteren Schritt mit den Testdaten evaluiert wird (siehe Abbildung 1). Dafür bedarf es eines *Train-* und *Testsplits* der Daten, worin die Daten zufällig in die Teile Train und Test aufgeteilt werden. Bewährte Aufteilungen für Train- und Testsplits sind 70/30 und 80/20.

Die verwendeten Techniken lassen sich dem *Supervised Learning* zuordnen, da der Krebstyp der einzelnen Stichproben zur Trainingszeit bereits bekannt ist und zugleich bestimmt wird (*Klassifikation*). Der Krebstyp bildet somit das *Label*. Das Ziel von Klassifikationsalgorithmen besteht darin, ein Modell basierend auf den Trainingsdaten zu schaffen, das die Labels möglichst korrekt vorhersagt.

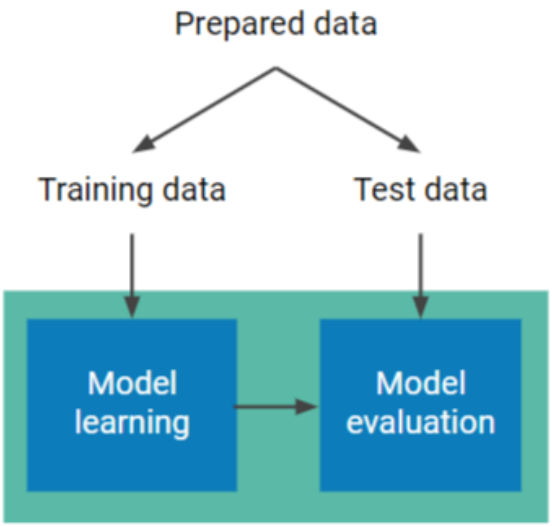


Abbildung 1: Das Lernen und Auswerten eines Modells über Train- und Testsplit. In Anlehnung an [Gon19].

2.2 Aufbau der Daten

Die Daten liegen in Form von drei binären Matrizen vor: SPM, CNA und SPM/CNA. In der SPM-Matrix werden ausschließlich Somatic Point Mutations aufgelistet, während die CNA-Matrix Copy Number Alterations aufweist. Die SPM/CNA-Matrix ist die Konkatenation beider Matrizen.

Die Matrizen sind derart aufgebaut, dass jede Zeile einen Patienten repräsentiert, welcher unter der ersten Spalte Patient-ID aufzufinden ist (siehe Tabelle 1). Mit *Sample* wird eine Zeile der Matrix bezeichnet. Die Spalte am Ende der Matrix weist einen der 28 Krebstypen für jedes Sample auf. Spalten mit Genbezeichnungen stellen *Features* dar. Die Gene mit einem Suffix am Ende der Genbezeichnung entstammen der CNA-Matrix, wobei *.p* für eine Amplifikation und *.n* für eine Deletion des Gens steht. Einsen oder Nullen innerhalb einer Matrix stehen für eine oder keine Mutation am entsprechenden Gen.

Patient-ID	A1CF	...	ZSWIM7.p	ZSWIM7.n	Label
TCGA.BL.A0C8.01	0	...	0	1	0
TCGA.BL.A13I.01	0	...	0	0	0
...	...	...	...	...	...
TCGA.YZ.A984.01	0	...	1	0	27
TCGA.YZ.A985.01	0	...	0	0	27

Tabelle 1: Auszug aus der SPM/CNA-Matrix.

Der vorhandene R-Code aus [Soh+17] wurde für eine erneute Datenerhebung angepasst und wiederverwendet. Wie in [Soh+17] wurden die Daten aus *cBioPortal* für die 28 Krebstypen erhoben und in binäre Matrizen prozessiert. Jegliche 0er-Zeilen und -Spalten aus den Daten wurden entfernt. Die hieraus hervorgehenden Dimensionen der Matrizen ohne Patient-ID und Label sind der Tabelle 2 zu entnehmen.

	Zeilen	Spalten
SPM	6581	7664
CNA	6076	6228
SPM/CNA	6595	13892

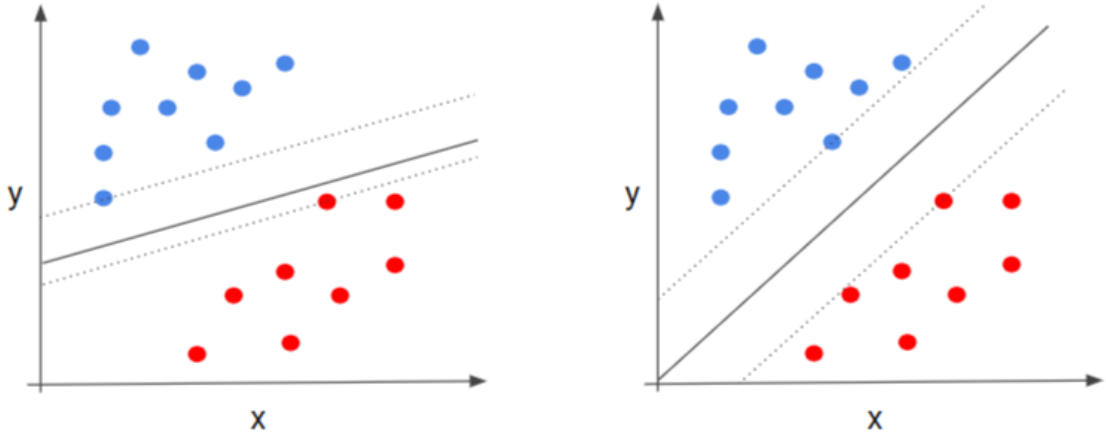
Tabelle 2: Feature-Dimensionen der SPM-, CNA- und SPM/CNA-Matrix.

2.3 Support Vector Machine

Eine Support Vector Machine (SVM) ist für binäre Klassifizierung ausgelegt. In einer solchen Klassifizierung können alle Labels, je nach Klassenzugehörigkeit, positiv oder negativ betrachtet werden.

Eine SVM sucht eine *Hyperebene*, um die Daten anhand ihres Labels bestmöglich zu separieren. Mit bestmöglicher Separierung ist die Maximierung des kleinsten Abstandes

der Trainingspunkte zur Hyperebene gemeint (*Maximum Margin*) [Nob06]. Im zweidimensionalen Fall zeigt sich die Hyperebene in einer Geraden. In Abbildung 2 ist der Unterschied zwischen einer möglichen Separierung der Daten und einer Separierung im Sinne des Maximum Margins dargestellt. Neue Datenpunkte werden abhängig von ihrer Lage zur Hyperebene, positiv oder negativ klassifiziert. Die Hyperebene stellt somit einen *Klassifikator* dar.



(a) Ohne Maxmimum Margin (b) Mit Maximum Margin

Abbildung 2: Hyperebenen ohne (a) und mit (b) Maximum Margin. Die blauen Datenpunkte stehen für positive Labels und die roten Datenpunkte für negative Labels. Die Punkte auf den gestrichelten Linien heißen *Support Vectors*.

Die Lage der Hyperebene mit Maximum Margin hängt lediglich von den Support Vectors ab und da der Abstand zu ihnen maximiert werden soll, befindet sich die Hyperebene mittig zwischen den Abständen der Support Vectors.

Aufgrund der Komplexität der Daten sind die Datenpunkte zumeist nicht vollständig linear separierbar. Dahingehend wird eine *Verlustfunktion* eingeführt, die den Fehler über den zu minimierenden Abstand zur Hyperebene bestimmt [HCL+03]. Die Gewichtung des Fehlers lässt sich über einen *Hyperparameter* bestimmen. Hyperparameter sind vor dem Training festgelegte Parameter, dessen optimale Werte nicht direkt aus den Daten hervorgehen und zumeist experimentell bestimmt werden. Hier wird $C > 0$ als Regularisierungsparameter bezeichnet. In dem Fall von nicht linear separierbaren Daten werden *Kernel*-Funktionen verwendet, welche die Daten in einen Raum höherer Dimension überführen und in dem Raum eine Hyperebene mit Maximum Margin finden [HCL+03]. Eine SVM mit linearem Kernel wird als lineare SVM bezeichnet.

Da SVMs im Wesentlichen auf binäre Klassifizierung beschränkt sind und in diesem Fall mehrere Klassen vorhergesagt werden sollen, bedarf es einer Umwandlung des Multi-Klassifizierungsproblems in binäre Klassifizierungsprobleme. Dies ist durch die Anwendung des *One-vs-Rest*-Prinzips möglich. Hierbei wird jede Klasse einzeln gegen alle anderen Klassen betrachtet, sodass für jede Klasse ein Klassifikator trainiert wird [vgl. Alp20, S. 413] (siehe Abbildung 3).

Die jeweilige Klasse wird für den entsprechenden Klassifikator als positives Label betrachtet, während alle anderen Klassen negativ gesehen werden. Dies resultiert wieder-

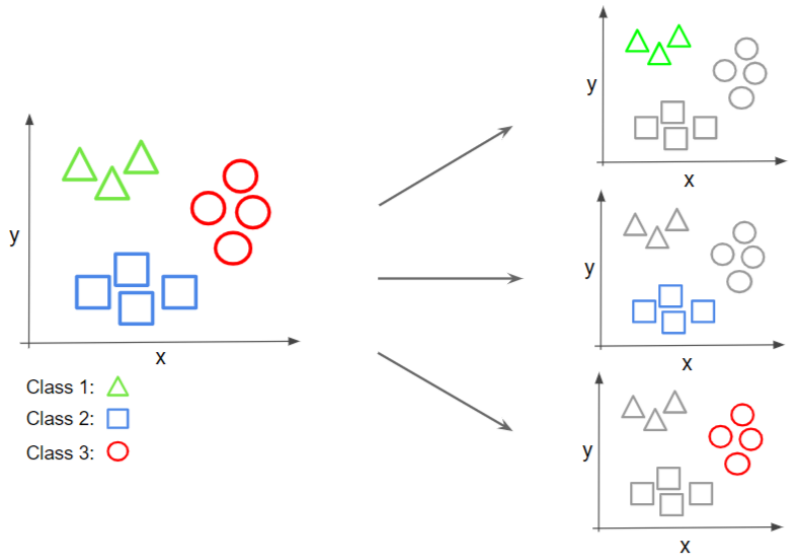


Abbildung 3: Funktionsweise eines One-vs-Rest-Klassifikators. Die grau markierten Datenpunkte werden als negative Labels gewertet. In Anlehnung an [Jat16].

um in mehreren binären Klassifizierungen, welche von der SVM gehandhabt werden können. Bei einer Multi-Klassifikation über das One-vs-Rest-Prinzip hängt die Ausgabe von dem binären Klassifikator ab, welcher das Optimierungsproblem bestmöglich lösen konnte.

2.4 Artificial Neural Network

Artificial Neural Networks sind an die Funktionsweise eines neuronalen Netzes im Gehirn angelehnt und eignen sich unter anderem für Klassifikation [Kro08]. Im Folgenden wird hier die Abkürzung ANN verwendet. ANNs bestehen aus mehreren miteinander verbundenen künstlichen Neuronen, die in *Layer* organisiert sind. Künstliche Neuronen verbinden verschiedene Layer, haben jedoch keine Verbindungen innerhalb einer Layer (siehe Abbildung 4).

Die erste Layer nennt sich *Input*-Layer und die letzte Layer bezeichnet man als *Output*-Layer. Weitere Layer dazwischen werden *Hidden*-Layer genannt und sind von der Größe und Anzahl her beliebig. Die Größe des Input- und Output-Layers ist von dem jeweiligen Problem und den zur Verfügung stehenden Daten abhängig. Einzelne Samples dienen als Eingabe für die Input-Layer, während die Ergebnisse der Output-Layer zu entnehmen sind. Ein ANN besteht aus mindestens einem Input- und Output-Layer.

Bei der Benennung des N-Layer-Modells wird die Input-Layer nicht in die Zählung miteinbezogen, da dort keine Berechnungen stattfinden. Demnach ist in Abbildung 4 ein 2-Layer-Neural-Network abgebildet.

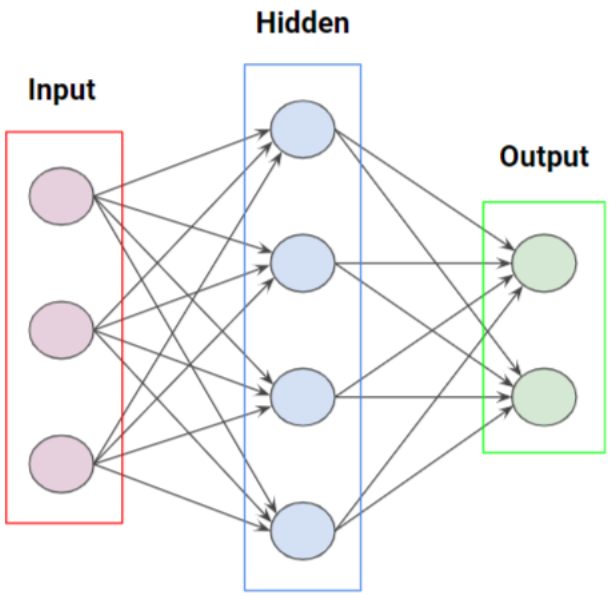


Abbildung 4: Schichtenmodell von ANNs.
In Anlehnung an [Mic20].

2.4.1 Neuronen

Künstliche Neuronen werden fortan als Neuronen bezeichnet. Sie können Signale senden und empfangen [Her18]. Signale stellen die Weiterleitung eines Wertes über ein- oder ausgehende Verbindungen dar. Ist ein Signal gleich 0, so findet keine Weiterleitung statt. Neuronen, die Signale aussenden, werden als aktiv bezeichnet, alle anderen Neuronen als inaktiv. Die Verbindungen zwischen Neuronen tragen Gewichte, welche im Lernprozess angepasst werden. Innerhalb der Neuronen findet eine Berechnung in Abhängigkeit der eingehenden Signale und Gewichte statt. Neuronen aus der Input-Layer hingegen leiten lediglich die Daten weiter.

Im ersten Schritt bestimmt ein Neuron die Summe der Produkte der eingehenden Signale mit den entsprechenden Gewichten, die Summe nennt sich auch *gewichtete Summe* [vgl. Zup94, S. 4]. Zumeist wird ein *Bias*-Neuron mit jedem Neuron aus den Hidden- und Output-Layer verbunden, um die Flexibilität des Netzwerks zu erhöhen, da das entsprechende Gewicht mit einem Signal ungleich 0 stets zur gewichteten Summe addiert und ebenfalls angepasst wird. Im nächsten Schritt wird der daraus resultierende Wert mit einer sogenannten *Aktivierungsfunktion* verrechnet (siehe Abbildung 5).

Für n eingehende Verbindungen eines Neurons seien $x_1, ..., x_n$ die eingehenden Signale und $w_1, ..., w_n$ die entsprechenden Gewichte. Sei $x_0 = 1$ das Bias-Neuron mit dem Gewicht w_0. Für die Berechnung des ausgehenden Signals y eines Neurons ergibt sich [vgl. Alp04, S. 233]:

$$y = f(\sum_{j=0}^{n} x_j w_j),$$

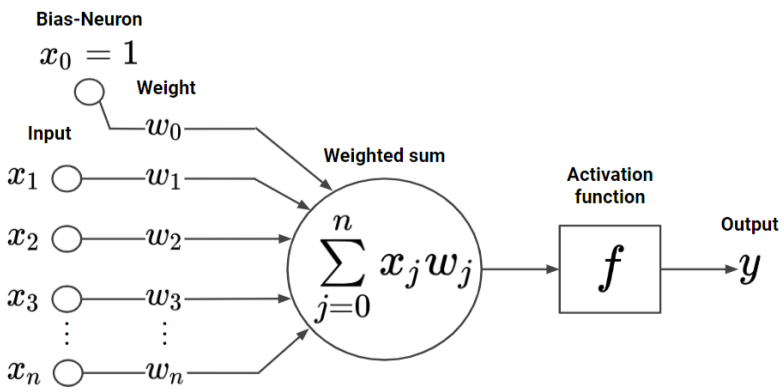


Abbildung 5: Funktionsweise eines Neurons von den eingehenden Signalen (Inputs) bis zum ausgehenden Signal (Output). In Anlehnung an [Art19].

wobei f eine Aktivierungsfunktion darstellt. Ausgehende Signale werden auch als *Activations* bezeichnet.

2.4.2 Aktivierungsfunktionen

Aktivierungsfunktionen sind verschieden und werden für die Hidden- und Output-Layer festgelegt. Sie skalieren die berechneten Ergebnisse der Neuronen auf einen neuen Wertebereich, welcher von der gewählten Funktion abhängig ist. Ebenso wird mit der Verwendung von Aktivierungsfunktionen Nicht-Linearität in das Netzwerk eingeführt [DHS12]. Nicht-Linearität bedeutet, dass der Output eines neuronalen Netzes nicht aus einer Linearkombination der Inputs reproduziert werden kann, was für die Lösung von nichtlinearen Problemen essenziell ist. Das zugrundeliegende Problem ist jedoch linear, da die lineare SVM in [Soh+17] die besten Resultate erzielt hat.

Zudem gibt es bestimmte Aktivierungsfunktionen, die sich besonders gut für Hidden- oder Output-Layer eignen. Die derzeit am meisten verwendete Aktivierungsfunktion für Hidden-Layer ist die *Rectified Linear Unit* (ReLU) [vgl. RZL17, S. 1].

$$\mathrm{ReLU}(x) = \max(0, x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \tag{1}$$

Hierbei werden alle Werte < 0 auf 0 gesetzt, während die anderen Werte erhalten bleiben. Der Wertebereich für ausgehende Signale nach der ReLU ist demnach $[0, \infty)$. In der Output-Layer werden üblicherweise Aktivierungsfunktionen verwendet, die einen Wertebereich zwischen 0 und 1 aufweisen. Für einen gegebenen Input in der Input-Layer und vorinitialisierte Gewichte lassen sich Schicht für Schicht Berechnungen für das Netzwerk durchführen. Der Weg von der Input-Layer bis zur Output-Layer wird *Forward Propagation* genannt und stellt lediglich einen Teil des Trainings dar, da die Signale nach der Output-Layer für die Gewichtsanpassung verwendet werden. Dafür bedarf es eines Maßes, inwieweit die Ergebnisse aus der Output-Layer den eigentlichen Werten entspre-

chen.

2.4.3 Verlustfunktion

Ein solches Maß ist mit der Verlustfunktion gegeben. Eine Verlustfunktion für die ANN bestimmt den Fehler einer Forward Propagation [vgl. And95, S. 242]. Es ist anzumerken, dass ANNs grundsätzlich in *Batches* trainiert werden [Rad17]. Batches sind zufällig ausgewählte Samples aus dem Trainingsdatensatz einer festgelegten Größe. Der Fehler wird für jede Batch berechnet, indem die Fehler der einzelnen Samples nach der Größe der Batch gemittelt werden. Das Ziel eines neuronalen Netzwerkes besteht darin, die Gewichte dermaßen anzupassen, dass der Fehler minimiert wird.

2.4.4 Backpropagation

Um den Fehler zu minimieren, muss die Verlustfunktion minimiert werden. Für diesen Schritt werden Gradienten benötigt, welche mit der *Backpropagation* berechnet werden. Der Gradient einer Funktion ist ein Vektor mit den ersten partiellen Ableitungen der Funktion und zeigt in Richtung des steilsten Anstiegs [Bro+89]. Für eine Minimierung der Verlustfunktion wird der berechnete Gradient negiert. Der Einfluss der Änderung eines Gewichts auf die Verlustfunktion kann daher über partielle Ableitungen der Verlustfunktion nach dem Gewicht bestimmt werden. Für die Gewichte $\theta = (w_1, w_2, ..., w_n)^T$ ist der Gradient von J nach θ wie folgt definiert.

$$\nabla_\theta J = \Big(\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, ..., \frac{\partial J}{\partial w_n}\Big)^T$$

Im Gegensatz zur Forward Propagation durchläuft die Backpropagation das Netzwerk in umgekehrter Reihenfolge von der Output-Layer zur Input-Layer und bestimmt die partiellen Ableitungen nach der Kettenregel. Für ein Neuron j sei die Activation durch a_j und dessen gewichtete Summe durch z_j gegeben. Die partielle Ableitung der Verlustfunktion J nach einem Gewicht w_{ij} von i nach j ist wie folgt [vgl. Alp04, S. 245] gegeben.

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial a_j} \frac{\partial a_j}{\partial z_j} \frac{\partial z_j}{\partial w_{ij}}$$

Für jede weitere Layer werden die partiellen Ableitungen auf die gleiche Weise berechnet und entsprechende Teilergebnisse der letzten Layer wiederverwendet.

2.4.5 Optimierer

Mit der Backpropagation werden lediglich die Gradienten berechnet. Diese werden von einem Optimierer zur Gewichtsanpassung verwendet. Es gibt verschiedene Optimierer, um die Verlustfunktion zu minimieren.

Ein bekannter Optimierer ist der *Stochastic Gradient Descent* (SGD) [Rud16]. SGD iteriert über alle Datenpunkte und aktualisiert die Gewichte unter Verwendung einer entspre-

chenden *Lernrate*. Die Lernrate gibt an, wie stark die Gewichte verändert werden sollen und wird über den *Lernparameter* α reguliert.

Ein Update der Gewichte für das betrachtete Sample x und dessen Label y ist durch

$$\theta = \theta - \alpha \underbrace{\nabla_\theta J(\theta; x, y)}_{\text{Backpropagation}}$$

gegeben.

Die Forward und Backpropagation des ganzen Trainingsdatensatzes wird mit *Epoche* bezeichnet. ANNs werden in mehreren Epochen trainiert, da jede weitere Epoche die Verringerung der Lernrate unter Verwendung der Gewichte aus der vorherigen Epoche erlaubt. Bleibt der Fehler in den Epochen nahezu unverändert, deutet dies auf eine Konvergenz des Optimierers hin und das Training kann beendet werden.

2.5 Evaluationsmethodik

Im Folgenden werden die gängigsten Methoden zur Evaluierung von Machine-Learning-Verfahren vorgestellt.

2.5.1 Accuracy

Die üblichste Form der Bewertung ist die *Accuracy*, welche die Performance eines Verfahrens wie folgt angibt.

$$\text{Accuracy} = \frac{\#\text{correct predictions}}{\#\text{all predictions}}$$

Die Accuracy steht für den prozentualen Anteil richtiger Vorhersagen in Anbetracht aller Vorhersagen.

2.5.2 Confusion Matrix

Für die weiteren Bewertungsverfahren ist die Einführung der Begrifflichkeiten *True Positives/Negatives* und *False Positives/Negatives* notwendig [HS15], wobei sich aus exemplarischen Gründen auf die binäre Klassifikation beschränkt wird. Demnach lassen sich die Samples bezüglich der einzelnen Klasse in positiv und negativ einordnen.

In Tabelle 3 ist die *Confusion Matrix* für eine solche Klassifikation zu sehen. Während die Spalten der Matrix die vorhergesagte Klasse repräsentieren, stehen die Zeilen für die eigentliche Klasse. Zu den True Positives (TP) gehören die richtig vorhergesagten positiven Samples und zu True Negatives (TN) die richtig vorhergesagten negativen Samples. False Positives (FP) sind eigentlich negative Samples, welche positiv fehlklassifiziert wurden und False Negatives (FN) stehen für Samples, die positiv sind und dennoch negativ

		Predicted	
		Positive	Negative
Actual	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

Tabelle 3: Schematische Darstellung der Confusion Matrix im binären Fall. In Anlehnung an [Koo18].

vorhergesagt wurden. Im Falle einer Multiklassen-Klassifikation lassen sich die eben beschriebenen Werte für jede einzelne Klasse auf dieselbe Weise bestimmen. Für die Evaluation wurden die absoluten Werte bezüglich ihrer eigentlichen Klasse normalisiert.

2.5.3 Precision, Recall und F1-Score

In [HS15] werden unter anderem *Precision*, *Recall* und *F1-Score* definiert, welche sich aus den Werten der Confusion Matrix berechnen lassen.

Die Precision gibt das Verhältnis von korrekt vorhergesagten positiven Samples zu den insgesamt positiv vorhergesagten Samples an.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Sie gibt an, wie viele der klassifizierten positiven Samples tatsächlich positiv sind. Eine hohe Precision steht somit gleichermaßen für eine geringe Anzahl an FPs.

Analog dazu lässt sich ein Recall bestimmen, welcher das Verhältnis von richtig vorhergesagten positiven Samples zu allen anderen positiven Samples darstellt.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Dieser gibt an, wie viele der positiven Samples positiv vorhergesagt wurden. Ein hoher Recall geht demnach mit einer geringen Anzahl an FNs einher.

Der F1-Score berechnet sich aus der Precision und dem Recall und bildet das harmonische Mittel beider Werte.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Folglich müssen für einen hohen F1-Score sowohl FPs als auch FNs möglichst gering sein. Precision, Recall und F1-Score werden für jede Klasse einzeln bestimmt. Für die Repräsentation dieser Scores über alle Klassen hinweg eignet sich ein gewichtetes Mittel, welches die Anzahl der Samples einer jeden Klasse für die Berechnung des Mittelwerts berücksichtigt.

2.5.4 ROC-Curve und AUC-Score

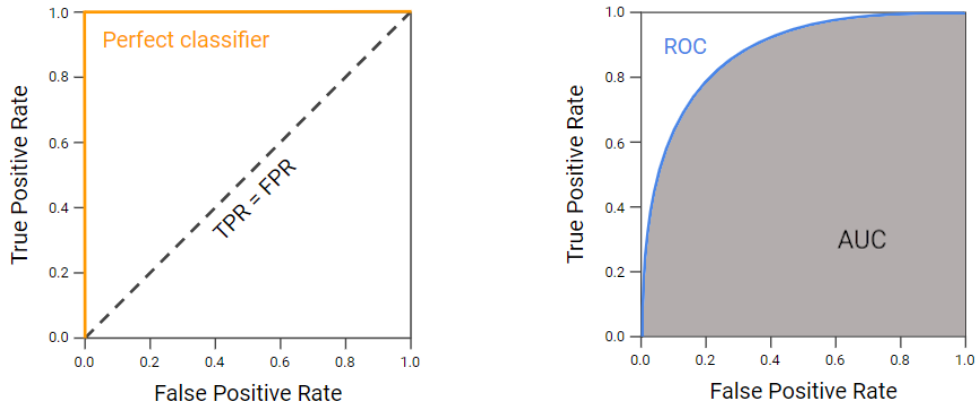
Für die Definition einer ROC-Curve (*Receiver Operating Characteristic Curve*) sind vorerst die Begriffe *True Positive Rate* (TPR) und *False Positive Rate* (FPR) zu erläutern [Faw06]. Die True Positive Rate ist wie der Recall definiert und zugleich als *Hit Rate* bekannt, da sie die Rate an positiv vorhergesagten positiven Samples angibt.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Die False Positive Rate ist unter dem Namen *False Alarm Rate* bekannt und gibt die Rate an positiv vorhergesagten negativen Samples an.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

Eine ROC-Curve plottet die True Positive Rate gegen die False Positive Rate. Für TPR = FPR (auch als *Reference Line* bezeichnet) ist das Modell am wenigsten aussagekräftig und gleicht einer zufälligen Voraussage, wohingegen eine optimale ROC-Curve durch die obere linke Ecke verläuft (siehe Abbildung 6a).



(a) (b)

Abbildung 6: In (a) ist die optimale ROC-Curve in gelb gekennzeichnet, während die Reference Line für TPR=FPR schwarz gestrichelt ist. In Anlehnung an [Rac19]. In (b) steht die grau markierte Fläche für die Fläche unterhalb der blau markierten ROC-Curve. In Anlehnung an [Sar18].

Der AUC-Score beschreibt die *Area Under the ROC Curve* (siehe Abbildung 6b). Je höher der AUC-Score, umso eher gleicht die entsprechende ROC-Curve der optimalen ROC-Curve. Hierbei ist die die TPR hoch, während die FPR möglichst gering ist. Dieser Score dient als Maß dafür, inwiefern das zugrundeliegende Modell zwischen den Klassen unterscheiden kann.

Die AUC-Scores befinden sich in einem Wertebereich von 0 und 1, wohingegen ein Score von 0.5 den schlechtesten Wert aufzeigt, da die Kurve für einen AUC-Score von < 0.5 entgegengesetzt gedeutet werden kann.

Im Falle einer Multi-Klassifikation wird die ROC-Curve und der entsprechende AUC-Score für jede Klasse nach dem One-vs-Rest-Prinzip bestimmt.

2.5.5 Kreuzvalidierung

Die Kreuzvalidierung ist ein wesentliches Konzept des Machine Learnings. Bei der k-fachen Kreuzvalidierung wird der Trainingsdatensatz in k gleichgroße Teilmengen aufgeteilt [HCL+03]. In k Durchläufen wird je eine dieser Teilmengen getestet, während das Modell bereits mit den $k - 1$ verbleibenden Teilmengen trainiert wurde. Die Kreuzvalidierung verwendet somit jedes Sample mindestens einmal im Train- und Testsplit und eignet sich insbesondere für die Bestimmung von Hyperparametern.

3 Methodik

Es folgen Methodiken bezüglich der Herangehensweise und Designentscheidungen hinsichtlich der Implementierung der Verfahren SVM und ANN. Der umgesetzte Code findet sich in [Rom20].

3.1 Train- und Testsplit

Da alle darauffolgenden Methoden dieselbe Trainings- und Testaufteilung erfordern, erfolgt die Aufteilung in Trainings- und Testdatensatz bereits im Voraus. Das Trainieren der Modelle sowie die Anpassung der Hyperparameter und die Auswahl an Features findet ausschließlich über den Trainingsdatensatz statt. 75% der Daten werden für das Training verwendet und 25% für die Evaluation [Soh+17]. Dabei ist zu beachten, dass der Datensatz einer unausgeglichenen Häufigkeitsverteilung unterliegt (siehe Abbildung 7).

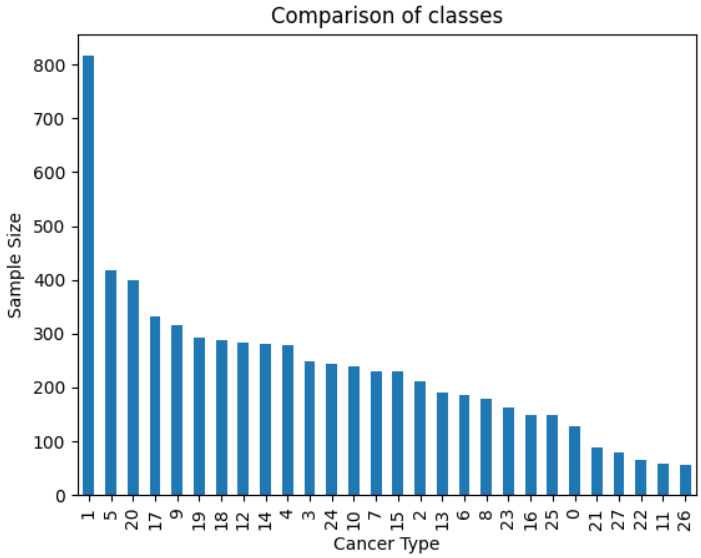


Abbildung 7: Häufigkeitsverteilung der 28 Krebstypen in der SPM/CNA-Matrix. Die Häufigkeiten sind absteigend von links nach rechts dargestellt.

Es lässt sich erkennen, dass die größte Klasse (1) mit über 800 Samples um ein Vielfaches größer ist als die kleinsten Klassen mit weniger als 100 Samples $(21, 27, 22, 11, 26)$. Hierfür wird eine Variation des Train- und Testsplits angewandt, in der die relativen Häufigkeiten jeder einzelnen Klasse bei der Aufteilung erhalten bleiben (*Stratified Split*).

3.2 LinearSVC

Für die Implementierung der linearen SVM eignen sich die Python-Module *SVC* und *LinearSVC* aus *scikit-learn* [Ped+11]. Im Gegensatz zu SVC stellt LinearSVC eine SVM mit

integriertem linearen Kernel dar und weist eine bei weitem höhere Skalierbarkeit auf [vgl. Fan+08, S. 1871]. In Anbetracht der Matrixdimensionen (siehe Tabelle 2) wird im Folgenden LinearSVC verwendet. LinearSVC regelt eine Multi-Klassifizierung standardmäßig über das One-vs-Rest-Prinzip.

3.2.1 Squared Hinge Loss

Bei der verwendeten Verlustfunktion für die SVM handelt es sich um den *Squared Hinge Loss* (L2-SVM) [vgl. Fan+08, S. 1872]. Für den anzupassenden Gewichtsvektor w und Features x_i des Samples $i \in \{1, ..., n\}$ mit zugehörigem Label $y \in \{-1, 1\}$ für n Samples ist der Squared Hinge Loss wie folgt definiert.

$$\xi(w; x_i, y_i) = \max(0, 1 - y_i w^T x_i)^2$$

Im Vergleich zum *Hinge Loss* (L1-SVM), welcher mit Ausnahme der Quadrierung identisch definiert ist, hat der Squared Hinge Loss für die SVM besser abgeschnitten (siehe Anhang A).

3.2.2 Regularisierungsparameter

Für die Bestimmung des Regularisierungsparameters C wurde die Library *optunity* verwendet [Cla+14]. Hierbei wurde der Parameter im Bereich zwischen 0 und 1 mittels 10-facher Kreuzvalidierung nach Genauigkeit und nach dem AUC-Score optimiert. Die daraus resultierenden C-Werte sind 0.0144 für die Genauigkeit und 0.2314 für den AUC-Score. Ersteres wird für die weiteren Evaluationen verwendet, da dieser einen höheren gewichteten F1-Score aufweist (siehe Anhang A).

3.3 PyTorch

Zur Implementierung des ANNs in Python eignet sich das Machine Learning Framework *PyTorch* [Pas+19].

3.3.1 Schichtenarchitektur

Da es sich bei dem gewählten Netzwerk um ein 2-Layer-Neural-Network handelt, hat das neuronale Netzwerk eine Input-Layer, eine Hidden-Layer und eine Output-Layer.

Die Größe des Input-Layers ist gleich der Anzahl der Features, diese ist für jede Matrix verschieden. Die Anzahl der Neuronen im Hidden-Layer wurde durch iterative Durchläufe in einem Bereich von 0 bis 10000 bestimmt (siehe Anhang B). Die Output-Layer ist durch die Anzahl vorauszusagender Klassen bestimmt (siehe Abbildung 8).

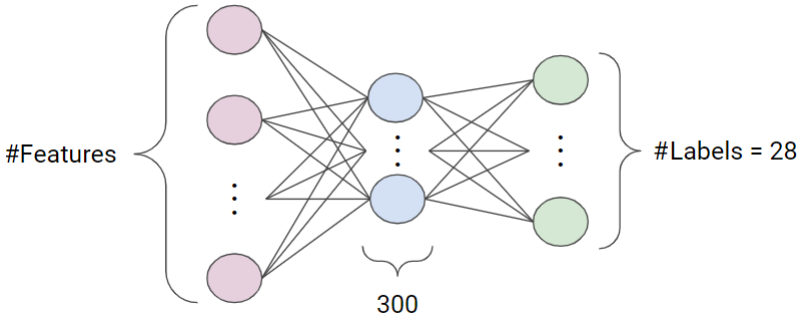


Abbildung 8: Die Größe der Input-Layer (links) entspricht der Anzahl an Features. Für die SPM-Matrix sind es 7664, 6228 für die CNA-Matrix und 13892 für die SPM/CNA-Matrix. Die Größe der Hidden-Layer (mittig) ist 300 und die der Output-Layer (rechts) entspricht der Anzahl der verschiedenen Labels und ist somit 28.

3.3.2 SELU

Bei der gewählten Aktivierungsfunktion nach der Hidden-Layer handelt es sich um SELU (*Scaled Exponential Linear Unit*) [Kla+17] mit fixem $\alpha \approx 1.6733$ und $\lambda \approx 1.0507$.

$$\text{SELU} = \lambda \begin{cases} x & x > 0 \\ \alpha e^x - \alpha & x \leq 0 \end{cases}$$

Da die durch Backpropagation bestimmte Ableitung der ReLU (1) für negativen Input ebenso 0 ist, ist der Gradient für ein entsprechendes Neuron 0. Bleibt der Input negativ findet kein Update des Gewichts mehr statt. Dieses Problem ist als *dying ReLU* bekannt [Lu+19]. Im Gegensatz zu ReLU weist SELU dieses Problem nicht auf.

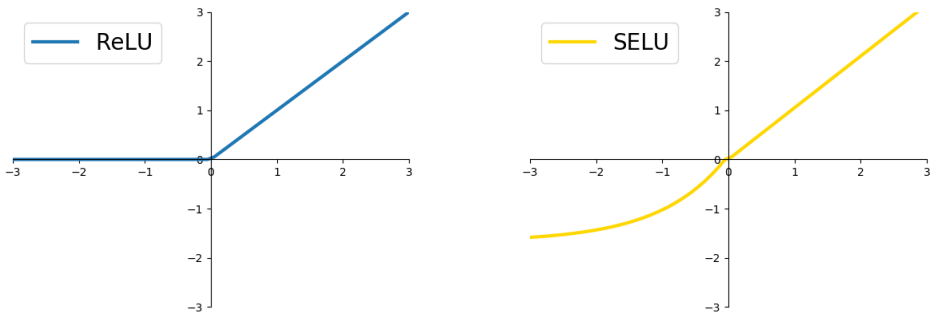


Abbildung 9: Zweidimensionale Darstellung von ReLU (links) und SELU (rechts) in einem Wertebereich zwischen -3 und 3.

In Abbildung 9 lässt sich erkennen, dass SELU im Vergleich zu ReLU eine skalierte Exponentialfunktion für negative Werte darstellt. Dessen Ableitung ist ungleich 0 und sorgt dafür, dass eine Gewichtsanpassung für negativen Input stattfinden kann.

SELU hat im Vergleich zu ReLU höhere Accuracies in den ersten 12 Epochen des Trainings erzielen können (siehe Abbildung 10).

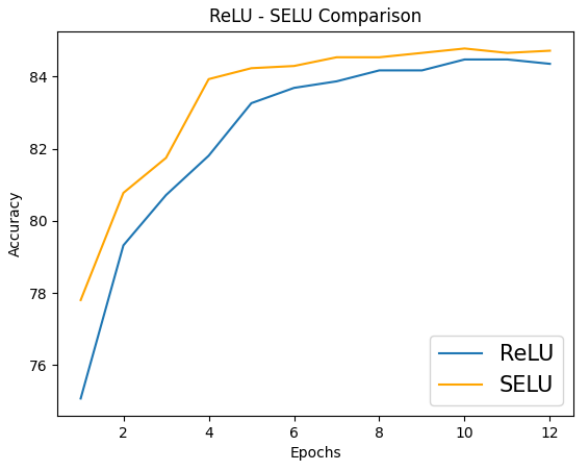


Abbildung 10: Graphischer Vergleich der Accuracies von ReLU (blau) und SELU (gelb) während des Trainings an der SPM/CNA-Matrix innerhalb die ersten 12 Epochen.

In Anhang B sind mehrere bekannte Aktivierungsfunktionen und dessen Genauigkeiten nach 12 Epochen gegenübergestellt.

3.3.3 LogSoftmax

Die Aktivierungsfunktion nach der Output-Layer ist *LogSoftmax*. Üblicherweise eignet sich die Verwendung einer *Softmax* nach der Output-Layer für eine Multi-Klassifikation. Diese bildet den Output auf den Wertebereich $[0, 1]$ ab [vgl. Goo+16, S. 184].

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

LogSoftmax ist als logarithmierter Softmax definiert und verwendet Wahrscheinlichkeiten auf logarithmischer Basis (*Log-Probabilities*) [Tor].

$$\text{LogSoftmax}(x_i) = \log\left(\frac{\exp(x_i)}{\sum_j \exp(x_j)}\right) = x_i - \log\left(\sum_j \exp(x_j)\right)$$

Bei einer solchen Transformation bleiben die Minima und Maxima der Wahrscheinlichkeiten aufgrund der monotonen Eigenschaft der Logarithmus-Funktion erhalten und werden lediglich auf einen neuen Wertebereich skaliert. Mit den Logarithmengesetzen erfolgen Ersparnisse in der Berechnung, welche bei besonders kleinen Wahrscheinlichkeiten für eine erhöhte numerische Stabilität sorgen.

3.3.4 Negative Log Likelihood Loss

Für Multiclass-Predictions eignet sich der *Negative Log Likelihood Loss* (NLLLoss) [Nie17]. Für n Samples einer Batch und N Klassen sei $\log(p_i)$ die *Log-Probability* aus der Output-Layer. Zudem sei y_i für genau ein x gleich 1 (das zutreffende Label) und sonst 0.

$$C = -\frac{1}{n} \sum_{x} \sum_{i=1}^{N} y_i \log(p_i)$$

Der Wert innerhalb der Summe wird als *Likelihood* bezeichnet und soll maximiert werden, da y_i nur für das korrekte Label 1 ergibt und höhere Log-Probabilities ebenfalls für eine höhere Wahrscheinlichkeit stehen. Dies ist äquivalent zu der Minimierung des Negative Log Likelihood Losses. Der Loss wird über die Größe der Batch gemittelt.

In der Python-Implementierung wird *CrossEntropyLoss* verwendet, welche die Aktivierungsfunktion LogSoftMax und die Verlustfunktion NLLLoss vereint.

3.3.5 Adabound

Bekannte Optimierer sind *Adam* und Stochastic Gradient Descent (SGD). Adam steht für Adaptive Moment Estimation und nutzt das Konzept des Moments [KB14]. Exemplarisch lässt sich das Moment anhand eines Balles, der eine Bodensenke hinunterrollt und dabei an zusätzlichem Moment gewinnt, verdeutlichen. Unter Verwendung des Moments können Optimierer mit weniger Gewichtsanpassungen konvergieren (siehe Abbildung 11).

(a) (b)

Abbildung 11: In beiden Graphiken sollen nach unten verlaufende Ebenen abgebildet sein. Der rote Punkt zeigt das Minimum an. In (a) sind die Schritte eines Optimierers ohne Nutzung des Moments abgebildet und in (b) die eines Optimierers mit Moment. In Anlehnung an [LeC+12].

Folgende Definitionen hinsichtlich Adam sind [KB14] entnommen worden. In Adam werden für jeden Schritt t Schätzungen des ersten (m_t) und des zweiten Moments (v_t) berechnet.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

g_t stellt den Gradienten über Backpropagation dar, während es sich bei $\beta_1, \beta_2 \in [0, 1)$ um Hyperparameter handelt. Für $t = 0$ werden m_t und v_t mit 0-Vektoren initialisiert, was dazu führt, dass die berechneten Werte von den erwarteten Werten abweichen. Um dem entgegenzuwirken, werden $\hat{m}_t$ und $\hat{v}_t$ berechnet.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Danach werden die korrigierten Werte mit der Lernrate α für das Update der Parameter verwendet.

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Der Parameter ϵ sorgt für eine erhöhte numerische Stabilität. Im Gegensatz zu SGD hat Adam für jedes Gewicht des neuronalen Netzwerks eine eigene Lernrate, die während des Trainings angepasst wird. Die Lernrate bei Adam sinkt proportional zur Quadratwurzel von t.

Adabound ist an Adam angelehnt und verwendet ebenso die Berechnung des ersten und zweiten Moments, sowie ein ähnliches Update der Parameter. Der wesentliche Unterschied zu Adam besteht in der Einführung von dynamischen Grenzen für die Lernrate α. Mit der *Clip-Operation* [Luo+19]

$$\text{Clip}(\alpha/\sqrt{v_t}, \eta_l(t), \eta_u(t)) = \max(\min(\alpha/\sqrt{v_t}, \eta_u(t)), \eta_l(t))$$

wird die Lernrate durch eine monoton steigende Funktion $\eta_l(t)$ für die untere Grenze und eine monoton fallende Funktion $\eta_u(t)$ für die obere Grenze beschränkt. Mit größer werdendem t konvergieren beide Funktionen zur finalen Lernrate α^* [vgl. Luo+19, S. 5]. Somit verhält sich Adabound zu Beginn des Trainings ähnlich wie Adam, da die Grenzen vorerst nur sehr wenig Einfluss auf die Lernraten ausüben. Mit jedem weiteren Iterationsschritt werden die Grenzen für den Lernparameter enger gesetzt und das Verfahren gleicht einer SGD.

3.4 Feature Selection

Die Auswahl von Features spielt eine wichtige Rolle im Bereich des Machine Learnings. Für die Diagnose eines Patienten ist es unvorteilhaft, ihn auf 13000 Gene überprüfen zu lassen. Mit den kommenden Verfahren werden zum einen Top-50 und zum anderen Top-100 Feature Sets mit einer möglichst guten Performance gesucht.

Im weiteren Verlauf wird je ein Verfahren für die *Wrapper*- und die *Embedded*-Methode der Feature Selection vorgestellt. Wrapper-Methoden verwenden das zu trainierende Modell für die Auswahl der Features. Embedded Methoden nutzen lediglich Modelle, welche die Features bereits im Trainingsprozess nach gewissen Kriterien bewerten [vgl. GE03, S. 1166].

3.4.1 Extra Tree Classifier

Die Feature Selection über den *Extra Tree Classifier* (ETC) gehört zu den Embedded-Methoden. Vorerst ist jedoch das Verständnis der Funktionsweise eines Entscheidungsbaumes (*Decision Tree*) notwendig.

Ein Decision Tree teilt den Datensatz anhand der Features in Teildatensätze, sodass ein Baum entsteht [vgl. Kha08, S. 173]. Jedes Feature steht für einen Knoten im Baum, während die Kinder der Knoten den Split nach dem entsprechendem Feature repräsentieren (siehe Abbildung 12). Von der Wurzel bis zu den Blättern finden vorerst Teilungen nach den Features statt, die die größte Wichtigkeit, auch Importance, aufweisen. Die Kriterien zur Bestimmung der *Feature Importance* werden nicht näher erläutert. Die Teilungen finden solange statt, bis die verbleibenden Daten in den Blättern unter ein Label fallen.

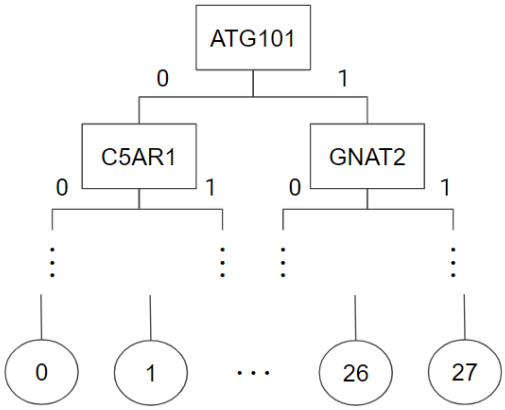


Abbildung 12: Exemplarische Darstellung eines Decision Trees.
In Anlehnung an [Tho17].

Ein *Random Forest* besteht aus mehreren Decision Trees [LW02]. Für die einzelnen Bäume im Random Forest werden Teildatensätze von zufällig ausgewählten Features verwendet. Für jedes dieser Features wird ein bestmöglicher Schwellenwert gesucht und anschließend wird nach dem besten Feature-Schwellenwert-Paar geteilt.

Der Extra Tree Classifier beruht auf dem Konzept der Extremely Randomized Trees und ist eine Modifikation des Random Forests. Hier werden die Schwellenwerte für jedes zufällig ausgewählte Feature eines Baumes zufällig ausgewählt. Dann wird nach dem besten Feature-Schwellenwert-Paar geteilt [GEW06]. Für den Extra Tree Classifier lässt sich die Anzahl an Bäumen übergeben. Aus [OPB12] geht hervor, dass 64 bis 128 Bäume in den meisten Fällen ideal sein sollen, da mehr Bäume die Berechnungszeit zwar erhöhen, aber keinen signifikanten Performance-Gewinn mehr liefern.

Für die Feature Selection ist die Vorhersage des ETC nicht von Bedeutung, stattdessen werden die Features mit den höchsten Feature Importances, welche über ETC berechnet wurden, bestimmt.

3.4.2 Recursive Feature Elimination

Ein weiteres Verfahren, welches in [Soh+17] in Kombination mit der SVM genutzt wurde ist die *Recursive Feature Elimination* (RFE). Hierbei handelt es sich um eine Wrapper-Methode, da die Feature Selection mit der zu trainierenden SVM bestimmt wird.

Für die RFE mit SVM werden wiederholt Modelle der SVM erstellt, während in jeder Iteration die am schlechtesten abschneidenden Features ausgelassen werden. Da die SVM über keine Feature Importance verfügt, werden die Gewichte für die Beurteilung der Relevanz einzelner Features verwendet. Nach jeder Iteration werden die Koeffizienten neu berechnet, wonach dann die zu entfernenden Features ausgewählt werden. Dieses Verfahren wird rekursiv auf die verbleibenden Daten angewandt bis die gewünschte Anzahl an Features erreicht ist [KJ19].

RFE nimmt zudem einen Parameter entgegen, welcher die Anzahl der zu entfernenden Features pro Iteration angibt. Der Wert für 0.05% pro Iterationsschritt wurde aus [Soh+17] übernommen.

3.4.3 RFE mit Kreuzvalidierung

Mit einer geringeren Auswahl an Features kann die Performance zunehmen, da zumeist nicht alle Features maßgeblich für die Klassifizierung sind. Insbesondere bei Genen ist eine hohe Redundanz der Daten nicht auszuschließen, da nicht direkt ersichtlich ist, welche Gene im Zusammenspiel mit anderen Genen wechselwirken.

Für die Suche nach einer optimalen Anzahl an Features wird RFE mit Kreuzvalidierung (RFECV) angewandt. RFE läuft wie zuvor mit einem Iterationsschritt von 0.05% in Kombination mit der SVM. Die über 5-fache Kreuzvalidierung berechnete Accuracy wird als Maß für die verwendete Anzahl an Features genommen.

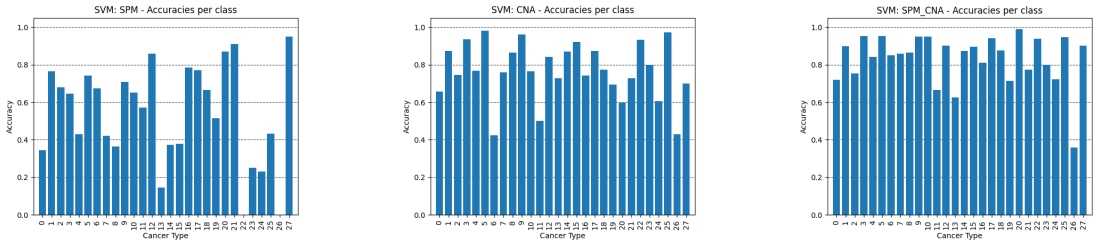
4 Auswertung

Es folgt die Evaluation der Machine-Learning-Algorithmen SVM und ANN, während die Techniken zur Feature Selection RFE und ETC gegenüberstehen. Zudem sind all diese Methoden auf die verschiedenen Matrizen SPM, CNA und SPM/CNA angewandt worden.

Für die finale Auswahl des Train- und Testsplits wurden zehn verschiedene Stratified Splits vorgenommen. Aus diesen Splits wird der Split für jede Matrix verwendet, welcher dem Mittel des gewichteten F1-Scores der zehn Splits am nächsten ist (siehe Anhang C).

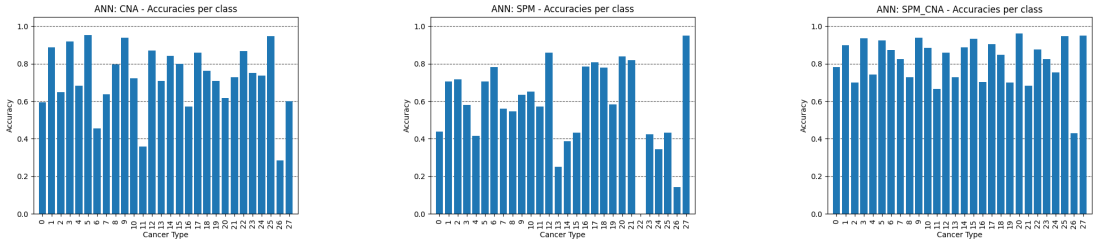
4.1 Evaluierung am ganzen Set

Die Accuracies der 28 Klassen wurden für jede der Matrizen für ANN und SVM geplottet.



(a) SPM-Matrix (b) CNA-Matrix (c) SPM/CNA-Matrix

Abbildung 13: Accuracies aller Klassen der SVM für die Matrizen SPM, CNA und SPM/CNA.



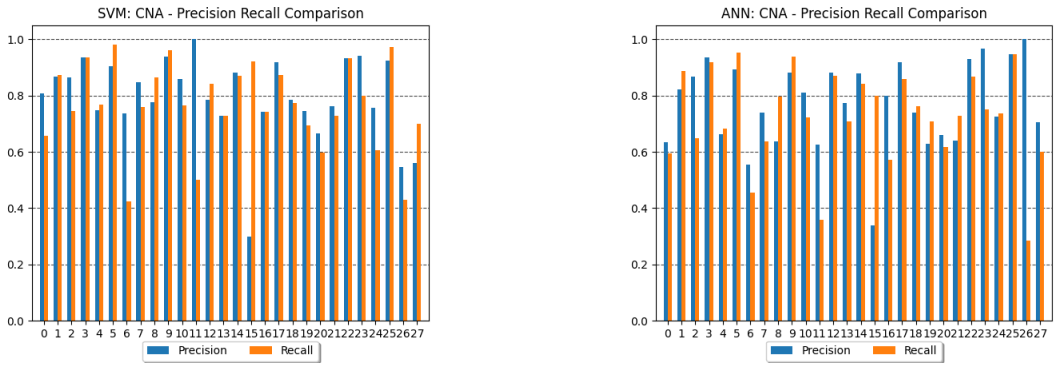
(a) SPM-Matrix (b) CNA-Matrix (c) SPM/CNA-Matrix

Abbildung 14: Accuracies aller Klassen der ANN für die Matrizen SPM, CNA und SPM/CNA.

In Abbildung 13a lässt sich erkennen, dass die Accuracy für die Labels $12, 20, 21$ und 27 über 80% beträgt. Die Accuracy der Labels 22 und 26 beträgt im gewählten Split der SPM-Matrix Null. Für die CNA-Matrix wird knapp die Hälfte der Labels mit einer Accuracy über 80% vorausgesagt (siehe Abbildung 13b), während dies für den Großteil der Labels bei der SPM/CNA-Matrix zutrifft. Für das Label 26 wird mit keiner der Matrizen eine

Accuracy von 50% erreicht. Die SVM hat die Gesamt-Accuracies von 60% für die SPM-Matrix, 80% für die CNA-Matrix und 87% für die SPM/CNA-Matrix erzielt.

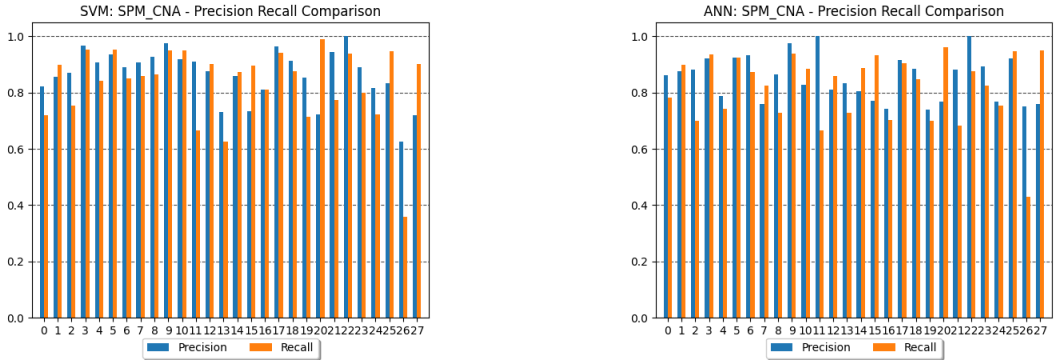
Die Accuracies der Labels der Matrizen für die ANN sind sich denen für die SVM sehr ähnlich (siehe Abbildung 14). Die ANN hat die Gesamt-Accuracies von 62% für die SPM-Matrix, 78% für die CNA-Matrix und 85% für die SPM/CNA-Matrix.



(a) SVM (b) ANN

Abbildung 15: Precision Recall Bar Plots der CNA-Matrix für SVM und ANN.

Die Precision Recall Bar Plots der CNA-Matrix zeigen für ANN und SVM einen hohen Recall und eine geringe Precision für Label 15 (siehe Abbildung 15). Demnach werden mehrere Labels in beiden Verfahren mit der CNA-Matrix fälschlicherweise als Label 15 klassifiziert. Die Labels 6 und 11 der CNA-Matrix für die SVM weisen eine weitaus höhere Precision als Recall auf. Für die ANN trifft selbiges für die Labels 11 und 26 zu.

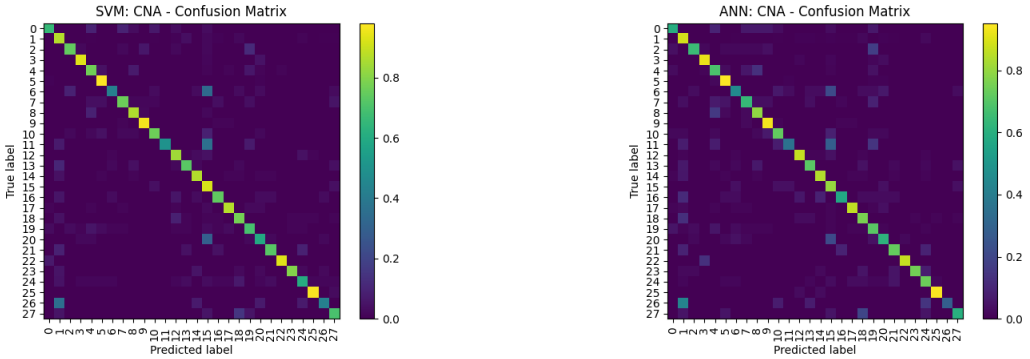


(a) SVM (b) ANN

Abbildung 16: Precision Recall Bar Plots der SPM/CNA-Matrix für SVM und ANN.

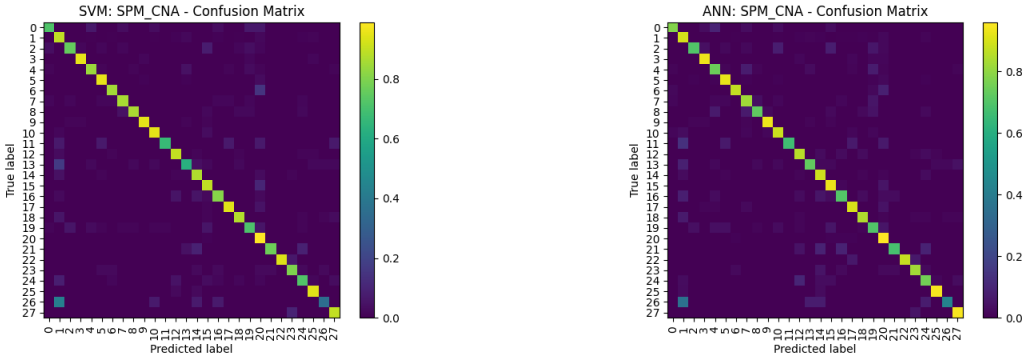
Im Gegensatz zu den Precision Recall Bar Plots für die CNA-Matrix sind Precision und Recall für die SPM/CNA-Matrix über ANN und SVM für die meisten Labels hoch und annähernd gleich (siehe Abbildung 16). Lediglich für die Labels 11 und 26 ist der Recall viel geringer als die Precision. Die Precision Recall Bar Plots der ANN und SVM für die SPM-Matrix sind in Anhang C. Da die F1-Scores einer jeden Klasse den Accuracies aus

Abbildung 13 und Abbildung 14 stark ähneln, sind diese ebenfalls in Anhang C vorzufinden.



(a) SVM (b) ANN

Abbildung 17: Confusion-Matrizen der CNA-Matrix für SVM und ANN.



(a) SVM (b) ANN

Abbildung 18: Confusion-Matrizen der SPM/CNA-Matrix für SVM und ANN.

Die Confusion-Matrizen der ANN und SVM sind für alle Matrizen sehr ähnlich. Insbesondere in Abbildung 17 lässt sich erkennen, dass die Verfahren ANN und SVM oft dieselben Labels fehlklassifizieren. So werden die Labels $6, 11$ und 20 für die CNA-Matrix oft als Label 15 vorausgesagt. Ferner wird Label 26 in beiden Verfahren für die CNA- und SPM/CNA-Matrix häufig als Label 1 klassifiziert (siehe Abbildung 17 und Abbildung 18). Für die SPM-Matrix finden sich die Confusion-Matrizen in Anhang C.

Für die SPM-Matrix lässt sich erkennen, dass ANN und SVM größtenteils dieselben Samples richtig klassifiziert haben (siehe Abbildung 19). ANN hat für die SPM-Matrix geringfügig mehr Samples richtig voraussagen können als SVM. Lagen sowohl ANN als auch SVM falsch mit der Prediction, so haben beide Verfahren zu ungefähr einem Drittel dieselbe falsche Prediction geliefert.

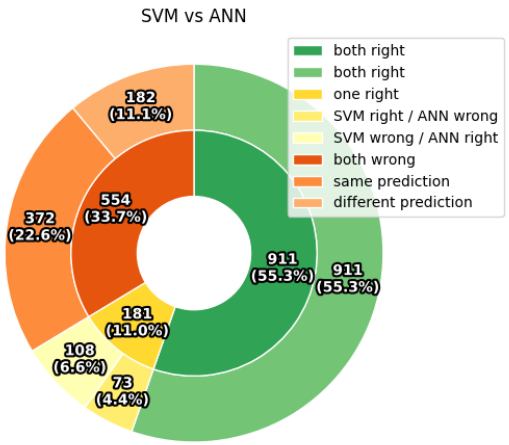
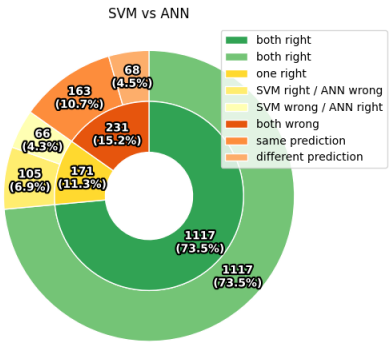
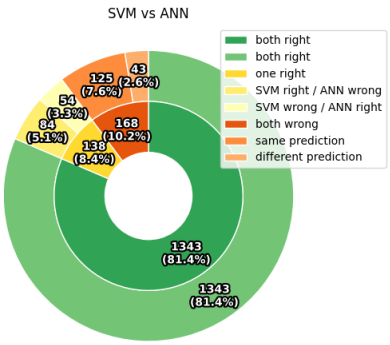


Abbildung 19: Anteile der falschen und richtigen Predictions zwischen SVM und ANN für die SPM-Matrix. Die prozentualen Anteile sind in Klammern angegeben, während die absoluten Werte darüberstehen. Die Legende gibt die Bedeutung des jeweiligen Anteils an. Der mittlere Kreis zeigt die gemeinsamen richtig oder falsch klassifizierten Anteile an.



(a) CNA-Matrix

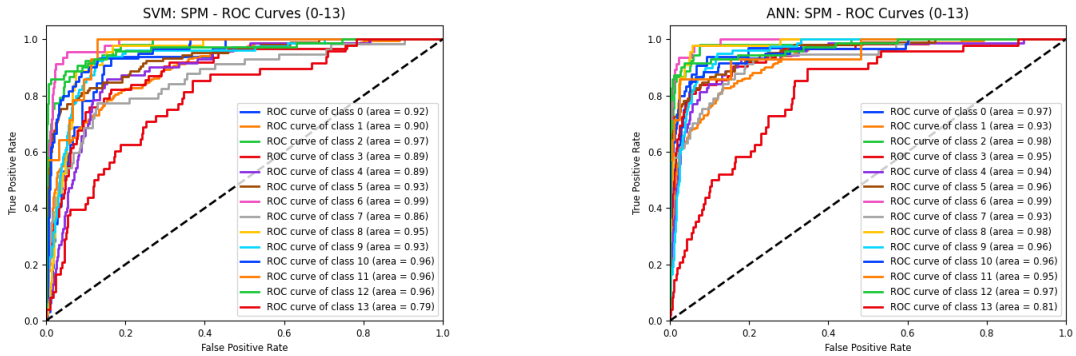


(b) SPM/CNA-Matrix

Abbildung 20: Anteile der falschen und richtigen Predictions zwischen ANN und SVM für die CNA- und SPM/CNA-Matrix.

Bei der CNA- und SPM/CNA-Matrix sind die relativen Predictions zwischen ANN und SVM sehr ähnlich (siehe Abbildung 20). Wieder haben ANN und SVM dieselben Samples zu einem Großteil richtig voraussagen können. Bei einer falschen Prediction hat hier jedoch SVM geringfügig mehr Samples richtig klassifiziert. In über zwei Drittel der Fälle haben ANN und SVM dieselbe Fehlklassifizierung aufgewiesen. Hier sind es knapp über 70% für die CNA-Matrix und nahezu 75% der Samples aus der SPM/CNA-Matrix.

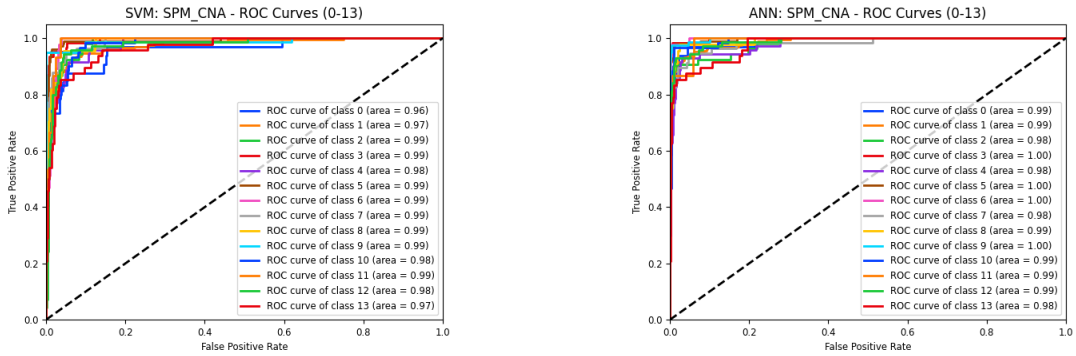
In [SR15] wird für ungleichgewichtige Datensätze dazu geraten, ROC-Curves mit besonderer Vorsicht zu deuten, weswegen eine möglichst allgemeine Bewertung stattfindet. Aus Gründen der Übersichtlichkeit wurden lediglich 14 der je 28 ROC-Curves pro Matrix auf einmal geplottet. Die Kurven sind nicht glatt wie in dem Beispiel in Abbildung 6b, da die vorauszusagenden Werte diskret sind.



(a) SVM (b) ANN

Abbildung 21: ROC-Curves und AUC-Scores für die ersten 14 Labels der SPM-Matrix für SVM und ANN.

In Abbildung 21 lässt sich erkennen, dass die ROC-Curves und die entsprechenden AUC-Scores der SPM-Matrix ähnlich für die SVM und ANN sind (die restlichen Kurven sind in Anhang C). Dennoch sind die AUC-Scores für die ANN höher als für die SVM.

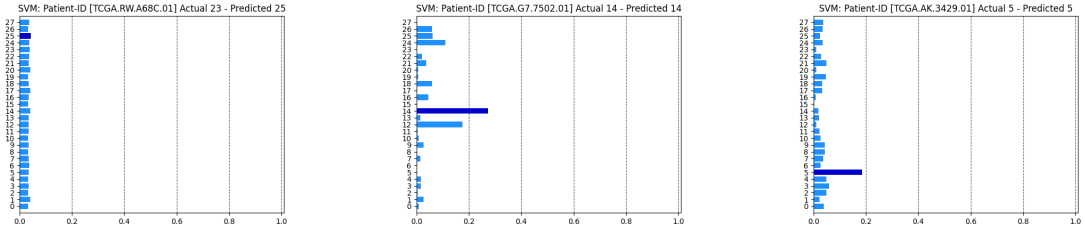


(a) SVM (b) ANN

Abbildung 22: ROC-Curves und AUC-Scores für die ersten 14 Labels der SPM/CNA-Matrix für SVM und ANN.

Im Gegensatz zu den ROC-Curves und AUC-Scores der SPM/CNA-Matrix sind die der SPM-Matrix wesentlich geringer (siehe Abbildung 22 und restliche Kurven in Anhang C). Die AUC-Scores der CNA-Matrix unterscheiden sich nur geringfügig von denen der SPM/CNA-Matrix für beide Verfahren (siehe Anhang C).

Für die Bestimmung mehrerer infrage kommender Krebsarten können die Wahrscheinlichkeiten einer jeden Klasse während einer einzelnen Prediction (*Prediction Probabilities*) für ANN und SVM geplottet werden. Die Prediction des Modells ist die Klasse mit der größten Wahrscheinlichkeit.

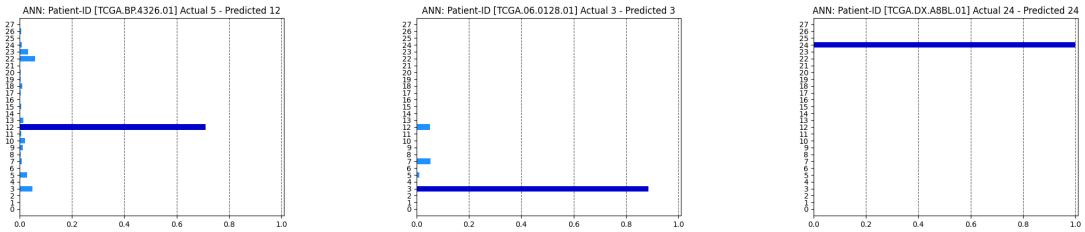


(a) SPM-Matrix (b) CNA-Matrix (c) SPM/CNA-Matrix

Abbildung 23: Prediction Probabilities von Samples aller drei Matrizen über SVM. Die Klasse mit der größten Wahrscheinlichkeit ist dunkelblau gekennzeichnet.

Für die SVM lässt sich bei der Voraussage der Krebsart über die SPM-Matrix erkennen, dass alle Prediction Probabilities sehr gering sind und sich kaum voneinander unterscheiden (siehe Abbildung 23a).

Dahingegen weisen die Prediction Probabilities der CNA- und SPM/CNA-Matrix große Unterschiede untereinander auf (siehe Abbildung 23b und Abbildung 23c). Die größte Wahrscheinlichkeit hebt sich besonders von den anderen Wahrscheinlichkeiten ab.



(a) SPM-Matrix (b) CNA-Matrix (c) SPM/CNA-Matrix

Abbildung 24: Prediction Probabilities von Samples aller drei Matrizen über ANN.

Die Prediction Probabilities der ANN weisen für die verschiedenen Matrizen kein solches Muster auf (siehe Abbildung 24). Zumeist kommen für die ANN ein bis drei verschiedene Klassen infrage, wobei sich die höchste Wahrscheinlichkeit besonders stark von den anderen Wahrscheinlichkeiten unterscheidet. Nicht selten wird lediglich eine Klasse mit einer 100%-igen Wahrscheinlichkeit vorausgesagt.

4.2 Top-50 und Top-100 Feature Sets

Zum Vergleich der Feature Selection Methoden RFE und ETC wurden die Gesamt-Accuracies beider Verfahren für jede Matrix bestimmt.

		ANN			SVM		
		SPM	CNA	SPM/CNA	SPM	CNA	SPM/CNA
Top-50	RFE	50%	75%	74%	47%	70%	72%
	ETC	49%	54%	69%	47%	51%	66%
Top-100	RFE	54%	79%	81%	52%	76%	80%
	ETC	54%	68%	75%	51%	65%	71%

Tabelle 4: Accuracies der Top-50 und Top-100 Feature Sets mittels RFE und ETC für jede Matrix über ANN und SVM.

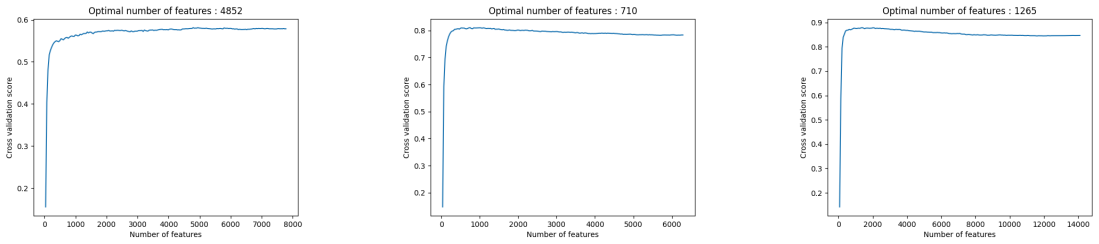
In Tabelle 4 lässt sich erkennen, dass die ANN für das Top-50 und Top-100 Feature Set höhere Genauigkeiten als die SVM erzielt hat. Für die SPM/CNA-Matrix hat die SVM gleichwertige Ergebnisse.

Für die Top-100 SPM-Matrix übertrifft die Accuracy beider Verfahren das Ergebnis aus [Soh+17], worin eine Accuracy von ungefähr 49.4% erreicht wurde.

In Anbetracht der SPM-Matrix sind nahezu keine Unterschiede zwischen RFE und ETC auszumachen, während ETC für die Matrizen CNA und SPM/CNA zu wesentlich schlechteren Ergebnissen als RFE führt.

4.3 Optimale Anzahl an Features

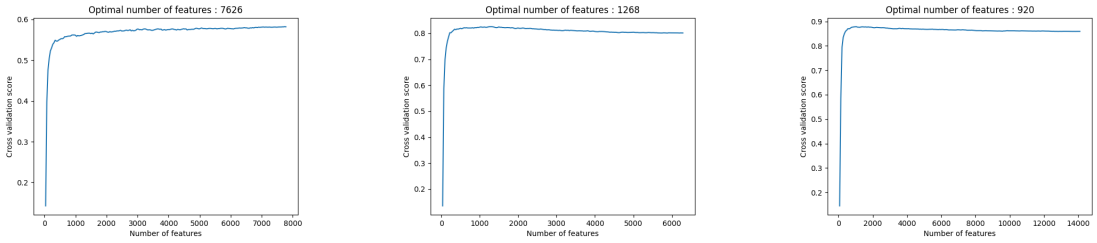
Die Durchführung von RFECV auf dem Trainingsdatensatz ergibt die in Abbildung 25 dargestellten Plots jeder Matrix, worin die Anzahl an Features gegen die aus der 5-fachen Kreuzvalidierung hervorgehenden Accuracy geplottet wird. Es zeigt sich für die SPM-Matrix eine Kurve, die mit einer höheren Anzahl an Features zunimmt. Für die CNA- und SPM/CNA-Matrix lässt sich ein Abfall der Performance nach einer bestimmten Anzahl an Features erkennen.



(a) SPM-Matrix (b) CNA-Matrix (c) SPM/CNA-Matrix

Abbildung 25: RFECV Plots der drei Matrizen über den Trainingsdatensatz. Die optimale Anzahl an Features ist für die SPM-Matrix 4852, 710 für die CNA-Matrix und 1265 für die SPM/CNA-Matrix.

Betrachtet man die Plots, die durch RFECV über dem ganzen Datensatz entstehen, so werden die Kurven für die CNA- und SPM/CNA-Matrix glatter und die Anzahl der optimalen Features verändert sich bei der CNA-Matrix auf 1268 und bei der SPM/CNA-Matrix auf 920 (siehe Abbildung 26). Dahingegen ist die daraus entstandene Kurve für die SPM-Matrix wie zuvor uneben und die Anzahl an optimalen Features steigt auf ein Maximum von 7626, was nahezu alle Features der SPM-Matrix ausmacht.



(a) SPM-Matrix. (b) CNA-Matrix. (c) SPM/CNA-Matrix

Abbildung 26: RFECV Plots der drei Matrizen über den ganzen Datensatz. Die optimale Anzahl an Features ist für die SPM-Matrix 7626, 1268 für die CNA-Matrix und 920 für die SPM/CNA-Matrix.

In Tabelle 5 lassen sich die Gesamt-Accuracies für die durch RFECV ausgewählten Features ablesen. ANN und SVM haben beide eine erhöhte Accuracy im Vergleich zu den Durchläufen, worin alle Features vorkamen. ETC liefert selbst für die CNA- und SPM/CNA-Matrix vergleichbare Ergebnisse zu RFE.

		ANN			SVM		
		SPM	CNA	SPM/CNA	SPM	CNA	SPM/CNA
Optimal (training set)	RFE	61%	83%	88%	59%	83%	88%
	ETC	61%	80%	84%	59%	79%	85%
Optimal (whole set)	RFE	61%	84%	90%	60%	86%	90%
	ETC	62%	80%	84%	60%	81%	84%

Tabelle 5: Accuracies der Verfahren aller Matrizen über Features der optimalen Anzahl des ganzen Datensatzes mittels RFE und ETC.

Für die Feature Selection nach der optimalen Anzahl wurden zum einen nur die Trainingsdaten verwendet und zum anderen der ganze Datensatz, so wie in [Soh+17]. Die Auswahl an Features nach dem ganzen Datensatz wurde aus Gründen der Vergleichbarkeit durchgeführt. Mit der letzteren Auswahl an Features wird die höchste Gesamt-Accuracy für die SPM/CNA-Matrix über SVM aus [Soh+17] übertroffen, welche 88.4% betrug. In Anhang C werden die gemittelten Ergebnisse ebendieser Auswahl an Features für zehn verschiedene Splits für ANN und SVM aufgeführt. Die Rate an ausgewählten Features für die SPM/CNA-Matrix ist unter Verwendung der Trainingsdaten und aller Daten nahezu identisch. Die Auswahl an Features für die SPM/CNA-Matrix besteht zu ungefähr 34% aus Genen der SPM-Matrix und zu 66% aus Genen der CNA-Matrix.

5 Fazit

Es folgt eine abschließende Zusammenfassung der verwendeten Methodik und bisherigen Ergebnissen gefolgt von dem Ausblick auf mögliche Erweiterungen.

5.1 Zusammenfassung

Für die lineare SVM wird der Squared Hinge Loss als Verlustfunktion mit einem Regularisierungsparameter $C = 0.0144$ verwendet. Die implementierte ANN ist ein 2-Layer-Neural-Network mit einer Hidden-Layer-Größe von 300. Für die Hidden-Layer wurde SELU als Aktivierungsfunktion verwendet und für die Output-Layer LogSoftMax. Die Verlustfunktion ist der Negative Log Likelihood Loss mit Adabound als Optimierer. Für die Feature Selection wurde SVM-basierte RFE und ETC verwendet und für die optimale Anzahl an Features SVM-basierte RFE mittels Kreuzvalidierung.

Insgesamt haben sowohl SVM als auch ANN gleichwertige Ergebnisse erzielen können. Werden alle Gene betrachtet, so hat die SVM bei der CNA und SPM/CNA-Matrix geringfügig besser als die ANN abgeschnitten. Für eine Teilmenge an Features übertreffen die Ergebnisse der ANN zumeist die der SVM.

Unter den Methoden der Feature Selection haben ANN und SVM für die Feature Selection über RFE bessere Resultate aufweisen können als über ETC. Lediglich für die Top-Features der SPM-Matrix kommen die Ergebnisse über ETC, denen mittels RFE gleich. Für die optimale Anzahl an Features gleichen die Ergebnisse über ETC, denen über RFE.

Unter Verwendung einer optimalen Anzahl an Features über das Trainingsset mittels SVM-basierter RFE haben sowohl ANN als auch SVM bessere Ergebnisse als bei den vollständigen Matrizen erzielen können. Die Hinzunahme des ganzen Datensatzes für RFECV lieferte wiederum noch bessere Ergebnisse, welche sich vor allem in der CNA- und SPM/CNA-Matrix bemerkbar machten. Das Verfahren der SVM konnte somit durch eine Feature Selection verbessert werden, während ANN gleichwertige Ergebnisse liefert.

Aus den Prediction Probabilities der SVM für die SPM-Matrix geht hervor, dass sich die SVM bei der Bestimmung des Krebstypen anhand dieser Matrix am unsichersten ist. Die eindeutigen Prediction Probabilities der ANN für die SPM-Matrix sind vermutlich auf die Aktivierungsfunktion in der Output-Layer zurückzuführen, da diese die Wahrscheinlichkeiten auf Basis der Exponentialfunktion skaliert. Der RFECV Plot ist für die SPM-Matrix uneben und es scheint als benötige man für eine möglichst gute Prediction alle Features der SPM-Matrix. Es liegt die Vermutung nahe, dass die SPM-Matrix weitaus weniger Informationen über den Krebstypen preisgibt als die CNA-Matrix. Dies wird zudem dadurch gestützt, dass ANN und SVM nahezu die gleichen Resultate über RFE und ETC bei der SPM-Matrix erzielt haben, wobei ETC allgemein schlechter als RFE abgeschnitten hat. Dennoch werden die Ergebnisse besser, wenn die SPM- und CNA-Matrix zusammen als eine SPM/CNA-Matrix verwendet werden. Zwei Drittel der endgültigen Auswahl an Features bestehen aus Einträgen der CNA-Matrix, obwohl die SPM-Matrix über mehr Features verfügt. Scheinbar gibt es Einträge in der SPM-Matrix, die nur in Zusammenhang mit den Einträgen aus der CNA-Matrix von größerer Bedeutung für den

Krebstypen sind.

In Bezug auf bisheriger Forschung mit Verweis auf [Soh+17] konnte bestätigt werden, dass die Verwendung beider Matrizen SPM und CNA bessere Resultate erzielt und, dass die Daten aus der CNA-Matrix von höherer Relevanz für die Krebsbestimmung sind. Zudem konnte die Accuracy der linearen SVM für die SPM/CNA-Matrix von 88.1% auf 90% erhöht werden. Gleichwertige Ergebnisse wurden über eine ANN realisiert.

5.2 Ausblick

Als Erweiterung könnte die Größe des Hidden-Layers und die Anzahl der Epochen für jede Matrix und jede Feature Selection einzeln optimiert werden. Ebenso wurde zur Bestimmung einer optimalen Anzahl an Features ausschließlich SVM-basierte RFE verwendet, worin die Ergebnisse der SVM als Performance-Kriterium gelten. Dennoch zeigte ANN ähnliche Ergebnisse. Eine Erweiterung wäre durch eine Feature Selection über ANNs, wie etwa in [VB02] beschrieben, gegeben. Zudem könnten mehrschichtige ANNs (*Deep Learning*) an das Problem der Klassifikation von Krebsarten herangeführt werden. Jedoch ist der Anteil der gleich-fehlklassifizierten Samples beider Verfahren in allen Matrizen ziemlich hoch. Label 15, welches in der CNA-Matrix einen hohen Recall aber eine geringe Precision aufweist, hat in der SPM/CNA-Matrix wiederum eine ähnlich hohe Precision im Vergleich zum Recall. Zudem werden knapp drei Viertel der fehlklassifizierten Samples unter Verwendung der SPM/CNA-Matrix von ANN und SVM gleich vorausgesagt. Dies lässt darauf schließen, dass bestimmte Einträge in der Matrix mehrdeutig sind und nicht klar von den Modellen unterschieden werden können. Verhältnismäßig werden die meisten Samples mit dem Label 26 als Label 1 klassifiziert, jedoch handelt es sich bei diesem Label um die kleinste Klasse im gesamten Datensatz. Es ist nicht gewiss, inwiefern eine weitere Optimierung des Verfahrens die Accuracy erhöhen würde.

Es bleibt zu erwähnen, dass es kein universelles Modell für alle Probleme gibt (No-Free-Lunch-Theorem) [Wol02]. Es ist demnach empfehlenswert, weitere Modelle zur Lösung des Problems der Klassifizierung von Krebsarten heranzuführen.

A SVM Tuning

Verlustfunktion	SPM	CNA	SPM/CNA
Hinge	54%	80%	86%
Squared Hinge	58%	81%	86%

Tabelle 6: Gewichtete F1-Scores der SVM aller Matrizen für die Verlustfunktionen Hinge Loss und Squared Hinge Loss. Für jede Matrix hat der Squared Hinge Loss bezüglich des gewichteten F1-Scores denselben oder einen höheren Wert als der Hinge Loss.

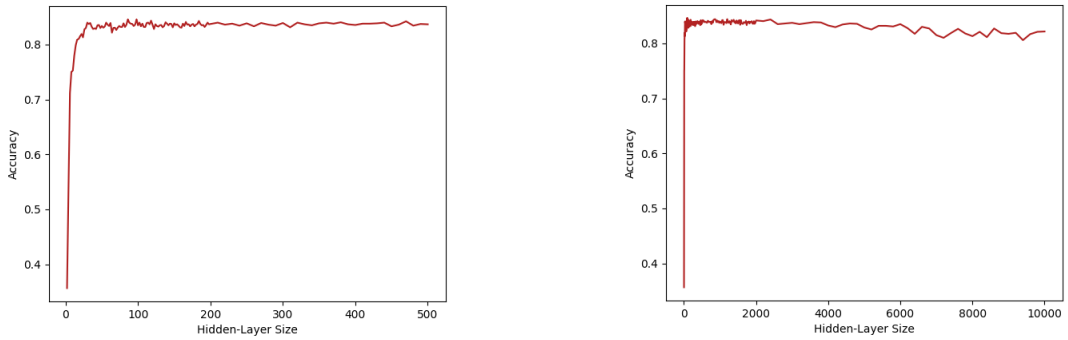
Optimierte Metrik	C-Wert	F1-Score
AUC-Score	0.2314	85%
Accuracy	0.0144	86%

Tabelle 7: Gewichtete F1-Scores der SVM für die SPM/CNA-Matrix mit den ermittelten C-Werten aus optunity. Der gewichtete F1-Score für den über die Optimierung der Accuracy bestimmte C-Wert ist höher.

B ANN Tuning

Anzahl an Neuronen	Schritt
2-200	2
210-500	10
575-2000	20
2200-10000	200

Tabelle 8: Verwendete Iterationsschritte bei der Bestimmung der Größe des Hidden-Layers aufgrund der höher werdenden Berechnungszeit für ein größeres Hidden-Layer.



(a) Im Bereich von 2-500

(b) Im Bereich von 500-10000

Abbildung 27: Graphisch dargestellte Accuracies für unterschiedlich großes Hidden-Layer in Bereichen von 2-500 und 500-10000. Für $100 - 2200$ Neuronen im Hidden-Layer bleibt die Accuracy gleich, während eine höhere Größe des Hidden-Layers ($4000+$) zu einem Abfall der Accuracy führt. Es wurde eine Größe von 300 gewählt.

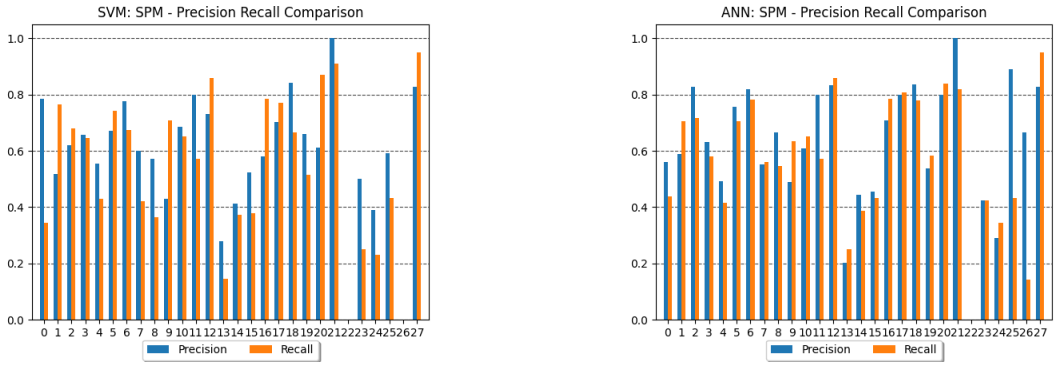
Aktivierungsfunktion	Accuracy
SELU	85%
ReLU	84%
Leaky ReLU	84%
Softmax	66%
LogSoftmax	72%
Sigmoid	82%
LogSigmoid	83%
tanh	84%

Tabelle 9: Accuracies der ANN für die SPM/CNA-Matrix nach 12 Epochen mit den Aktivierungsfunktionen SELU, ReLU, Leaky ReLU, Softmax, LogSoftmax, Sigmoid, LogSigmoid und tanh.

C Ergänzung zur Evaluation

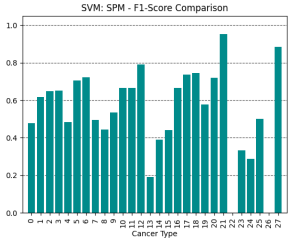
Splits 0-9	Weighted F1-Score	Splitindex mit nächstem F1-Score	
SPM-Matrix	59.7	0	59.73
CNA-Matrix	79.4	7	79.33
SPM/CNA-Matrix	84.9	0	84.98

Tabelle 10: Die gewichteten F1-Scores aller zehn Stratified Splits wurden über die Verfahren ANN und SVM gemittelt. Diese F1-Scores einer jeden Matrix waren Richtwert für die Wahl eines möglichst repräsentativen Splits. Von den zehn verfügbaren Splits pro Matrix wurden die drei Splits mit dem Index $0, 7$ und 0 für die SPM-, CNA- und SPM/CNA-Matrix verwendet.

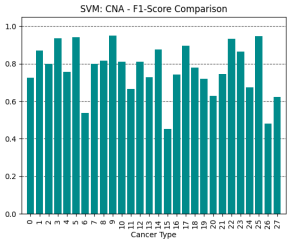


(a) (b)

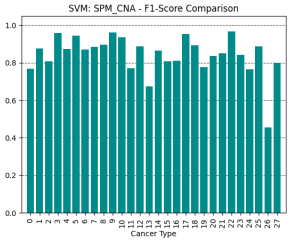
Abbildung 28: Precision Recall Bar Plots der SPM-Matrix für SVM (a) und ANN (b). Es zeigt sich die Unausgeglichenheit der Precision und des Recalls. ANN und SVM haben weitestgehend ähnliche Resultate erzielt.



(a) SPM-Matrix

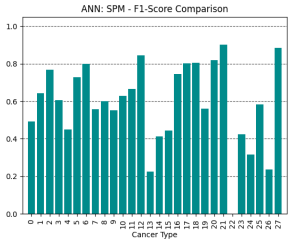


(b) CNA-Matrix

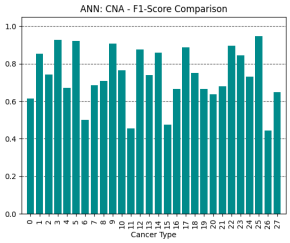


(c) SPM/CNA-Matrix

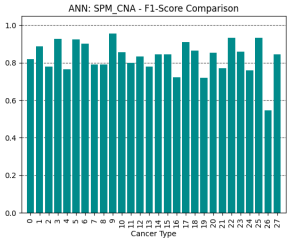
Abbildung 29: Hier sind die F1-Scores einer jeden Klasse für die Matrizen SPM, CNA und SPM/CNA über die SVM als Bar Plot dargestellt.



(a) SPM-Matrix



(b) CNA-Matrix

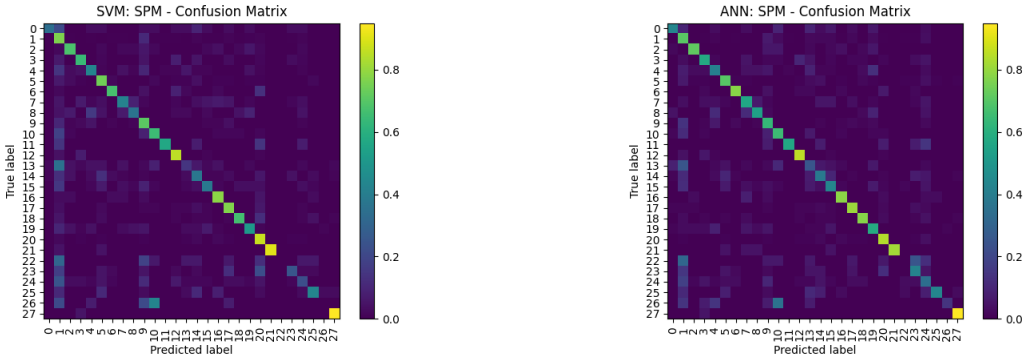


(c) SPM/CNA-Matrix

Abbildung 30: F1-Scores jeder Klasse für alle Matrizen über ANN.

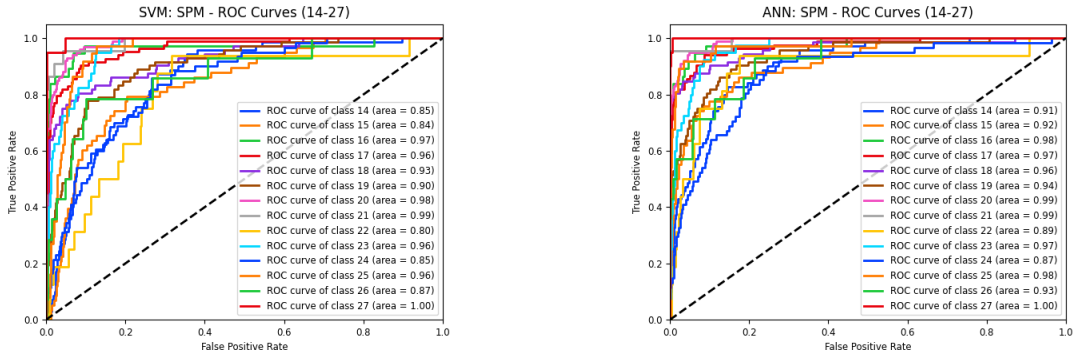
Average scores	ANN	SVM
Accuracy	0.897	0.902
Precision	0.899	0.905
Recall	0.897	0.902
F1-Score	0.897	0.902

Tabelle 11: Gemittelte Scores über zehn Splits nach der optimalen Feature Selection mittels des ganzen Datensatzes der SPM/CNA-Matrix für ANN und SVM. Bei den Werten handelt es sich um über die Splits gemittelte Scores, welche den jeweils gewichteten Score für Precision, Recall und F1 aufwiesen.



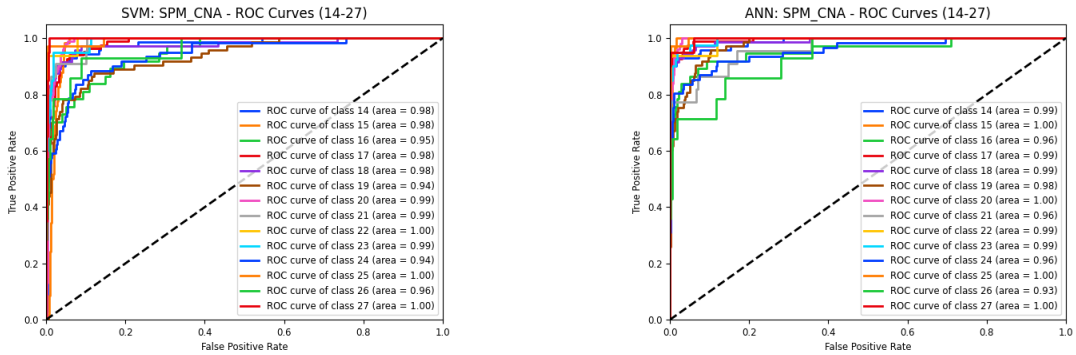
(a) (b)

Abbildung 31: Confusion-Matrizen der SVM (a) und ANN (b) für die SPM-Matrix. Es zeigen sich weitere Ähnlichkeiten bezüglich der Predictions von ANN und SVM.



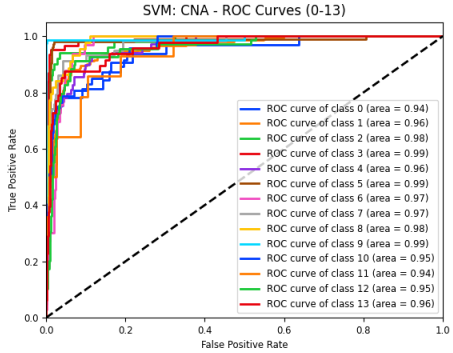
(a) (b)

Abbildung 32: Weitere ROC-Curves und AUC-Scores der SVM (a) und ANN (b) für die SPM-Matrix. Die Kurven sehen sich für beide Verfahren sehr ähnlich.

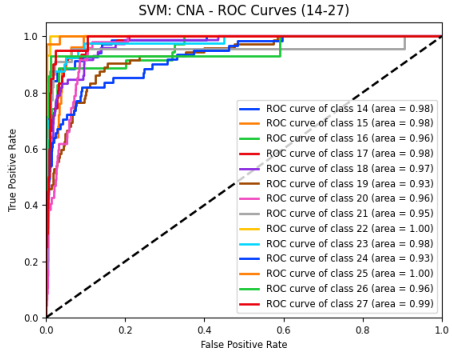


(a) SVM (b) ANN

Abbildung 33: Weitere ROC-Curves und AUC-Scores der SVM (a) und ANN (b) für die SPM/CNA-Matrix.

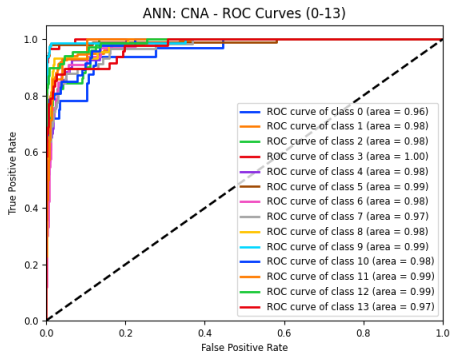


(a)

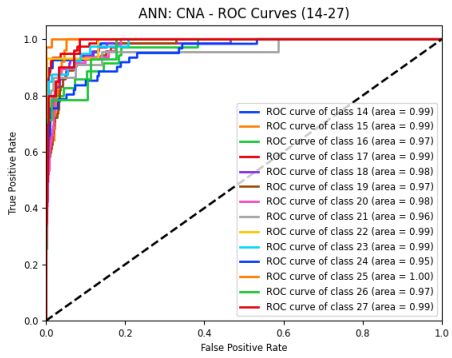


(b)

Abbildung 34: ROC-Curves und AUC-Scores der SVM für die CNA-Matrix. In (a) sind die ersten 14 Labels abgebildet und in (b) die letzten 14 Labels.



(a) Kurven 0-13



(b) Kurven 14-27

Abbildung 35: ROC-Curves und AUC-Scores der ANN für die CNA-Matrix. In (a) sind sie für die ersten 14 Labels abgebildet und in (b) die letzten 14 Labels.

Literaturverzeichnis

[Alp04] Ethem Alpaydin. *Introduction to Machine Learning, ser. Adaptive computation and machine learning*. 2004.

[Alp20] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[And95] James A Anderson. *An introduction to neural networks*. MIT press, 1995.

[Art19] Arthur Arnx. *Operations done by a neuron*. 2019. URL: https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf (besucht am 30. 06. 2020).

[Ber+10] Rameen Beroukhim, Craig H Mermel, Dale Porter, Guo Wei, Soumya Raychaudhuri, Jerry Donovan, Jordi Barretina, Jesse S Boehm, Jennifer Dobson, Mitsuyoshi Urashima et al. „The landscape of somatic copy-number alteration across human cancers“. In: *Nature* 463.7283 (2010), S. 899–905.

[Bro+89] Ilja N Bronstein, Konstantin A Semendjajew, Gerhard Musiol und Heiner Mühlig. „Taschenbuch der Mathematik, 24“. In: *Auflage, Thun u. Frankfurt/Main: Harri Deutsch* (1989).

[Cla+14] Marc Claesen, Jaak Simm, Dusan Popovic und BD Moor. „Hyperparameter tuning in Python using Optunity“. In: *Proceedings of the International Workshop on Technical Computing for Machine Learning and Mathematical Engineering*. Bd. 1. 2014, S. 3.

[DHS12] Richard O Duda, Peter E Hart und David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[Fan+08] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang und Chih-Jen Lin. „LIBLINEAR: A Library for Large Linear Classification“. In: *Journal of Machine Learning Research* 9 (2008), S. 1871–1874.

[Faw06] Tom Fawcett. „An introduction to ROC analysis“. In: *Pattern recognition letters* 27.8 (2006), S. 861–874.

[GE03] Isabelle Guyon und André Elisseeff. „An introduction to variable and feature selection“. In: *Journal of machine learning research* 3.Mar (2003), S. 1157–1182.

[GEW06] Pierre Geurts, Damien Ernst und Louis Wehenkel. „Extremely randomized trees“. In: *Machine learning* 63.1 (2006), S. 3–42.

[Gon19] Gonzalo Ferreiro Volpi. *Train and test split*. 2019. URL: https://towardsdatascience.com/6-amateur-mistakes-ive-made-working-with-train-test-splits-916fabb421bb (besucht am 22. 06. 2020).

[Goo+16] Ian Goodfellow, Yoshua Bengio, Aaron Courville und Yoshua Bengio. *Deep learning. vol. 1*. 2016.

[HCL+03] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin et al. *A practical guide to support vector classification*. 2003.

[Her18] John A Hertz. *Introduction to the theory of neural computation*. CRC Press, 2018.

[HS15] Mohammad Hossin und MN Sulaiman. „A review on evaluation metrics for data classification evaluations“. In: *International Journal of Data Mining & Knowledge Management Process* 5.2 (2015), S. 1.

[Jat16] Jatin Nanda. *1-vs All SVM classifier*. 2016. URL: https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj4/html/jnanda3/index.html (besucht am 06. 06. 2020).

[KB14] Diederik P Kingma und Jimmy Ba. „Adam: A method for stochastic optimization“. In: *arXiv preprint arXiv:1412.6980* (2014).

[Kha08] Shahzad Khan. „Ethem Alpaydin. Introduction to Machine Learning (Adaptive Computation and Machine Learning Series). The MIT Press, 2004.“ In: *Natural Language Engineering* 14.1 (2008), S. 133–137.

[KJ19] Max Kuhn und Kjell Johnson. *Feature engineering and selection: A practical approach for predictive models*. CRC Press, 2019.

[Kla+17] Günter Klambauer, Thomas Unterthiner, Andreas Mayr und Sepp Hochreiter. „Self-normalizing neural networks“. In: *Advances in neural information processing systems*. 2017, S. 971–980.

[Koo18] Koo Ping Shung. *Confusion Matrix*. 2018. URL: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 (besucht am 08. 06. 2020).

[Kro08] Anders Krogh. „What are artificial neural networks?“ In: *Nature Biotechnology* 26.2 (Feb. 2008), S. 195–197.

[LeC+12] Yann A LeCun, Léon Bottou, Genevieve B Orr und Klaus-Robert Müller. „Efficient backprop“. In: *Neural networks: Tricks of the trade*. Springer, 2012, S. 9–48.

[Lu+19] Lu Lu, Yeonjong Shin, Yanhui Su und George Em Karniadakis. „Dying relu and initialization: Theory and numerical examples“. In: *arXiv preprint arXiv:1903.06733* (2019).

[Luo+19] Liangchen Luo, Yuanhao Xiong, Yan Liu und Xu Sun. „Adaptive gradient methods with dynamic bound of learning rate“. In: *arXiv preprint arXiv:1902.09843* (2019).

[LW02] Andy Liaw und Matthew Wiener. „Classification and Regression by randomForest“. In: *R News* 2.3 (2002), S. 18–22.

[Mic20] Michael Chan. *Classical Neural Network*. 2020. URL: https://towardsdatascience.com/classical-neural-network-what-really-are-nodes-and-layers-ec51c6122e09 (besucht am 30. 06. 2020).

[Nie17] Michael Nielsen. „Improving the way neural networks learn“. In: *Neural Networks and Deep Learning* (2017).

[Nob06] William S Noble. „What is a support vector machine?“ In: *Nature biotechnology* 24.12 (2006), S. 1565–1567.

[OPB12] Thais Mayumi Oshiro, Pedro Santoro Perez und José Augusto Baranauskas. „How many trees in a random forest?“ In: *International workshop on machine learning and data mining in pattern recognition*. Springer. 2012, S. 154–168.

[Pas+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai und Soumith Chintala. „PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Hrsg. von H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox und R. Garnett. Curran Associates, Inc., 2019, S. 8024–8035.

[Ped+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot und E. Duchesnay. „Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830.

[Rac19] Rachel Draelos. *Measuring Performance: AUC (AUROC)*. 2019. URL: https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/ (besucht am 28. 06. 2020).

[Rad17] Pavlo M Radiuk. „Impact of training set batch size on the performance of convolutional neural networks for diverse datasets". In: *Information Technology and Management Science* 20.1 (2017), S. 20–24.

[Rom20] Roman Jumatov. *Classification of cancer types through tumour DNA using support vector machines and artificial neural networks*. 2020. URL: https://gitlab.cs.uni-duesseldorf.de/tran/classification-of-cancer-types-through-tumour-dna-using-support-vector-machines-and-artificial-neural-networks (besucht am 10. 08. 2020).

[Rud16] Sebastian Ruder. „An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[RZL17] Prajit Ramachandran, Barret Zoph und Quoc V Le. „Searching for activation functions". In: *arXiv preprint arXiv:1710.05941* (2017).

[Sar18] Sarang Narkhede. *AUC - ROC Curve*. 2018. URL: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5 (besucht am 28. 06. 2020).

[Soh+17] Kee Pang Soh, Ewa Szczurek, Thomas Sakoparnig und Niko Beerenwinkel. „Predicting cancer type from tumour DNA signatures". In: *Genome medicine* 9.1 (2017), S. 104.

[SR15] Takaya Saito und Marc Rehmsmeier. „The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets". In: *PloS one* 10.3 (2015), e0118432.

[Tho17] Thomas Plapinger. *What is a Decision Tree?* 2017. URL: https://towardsdatascience.com/what-is-a-decision-tree-22975f00f3e1 (besucht am 04. 07. 2020).

[Tor] Torch Contributors. *LOGSOFTMAX*. URL: https://pytorch.org/docs/stable/generated/torch.nn.LogSoftmax.html (besucht am 29. 06. 2020).

[VB02] Antanas Verikas und Marija Bacauskiene. „Feature selection with neural networks“. In: *Pattern recognition letters* 23.11 (2002), S. 1323–1335.

[Wol02] David H Wolpert. „The supervised learning no-free-lunch theorems“. In: *Soft computing and industry*. Springer, 2002, S. 25–42.

[Zup94] Jure Zupan. „Introduction to artificial neural network (ANN) methods: what they are and how to use them“. In: *Acta Chimica Slovenica* 41 (1994), S. 327–327.