technische universität
dortmund

fakultät für informatik

Masterarbeit

Implementierung von hybriden Methoden zur Instrumentenerkennung in verrauschten Musikdaten

Benedikt Adrian

22. Mai 2020

Gutachter:

Dr. Igor Vatolkin

M. Sc. Jurij Kuzmic

Fakultät für Informatik
Lehrstuhl für Algorithm Engineering (LS11)
Technische Universität Dortmund

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Motivation . . . 2
1.2 Verwandte Arbeiten . . . 2
1.2.1 Audioerkennung . . . 3
1.2.2 Musikerkennung . . . 3
1.2.3 Instrumentenerkennung . . . 4
1.2.4 Bezug zur Arbeit . . . 5
1.3 Aufbau der Arbeit . . . 5

2 Grundlagen der Audiosignalverarbeitung **7**

2.1 Audiosignale . . . 7
2.2 Repräsentationen . . . 8
2.2.1 Spektrale Merkmale . . . 8
2.2.2 Weitere Merkmale . . . 11
2.3 Normalisierung . . . 12
2.3.1 Z-Wert-Normalisierung . . . 12
2.3.2 Min-Max-Normalisierung . . . 13
2.3.3 Pro-Kanal-Energienormalisierung . . . 13
2.4 Datenaugmentierungen . . . 14
2.4.1 Allgemeine Augmentierungen . . . 14
2.4.2 Audioaugmentierungen . . . 16

3 Grundlagen der Klassifikation **21**

3.1 Einführung . . . 21
3.1.1 Muster . . . 22
3.1.2 Training . . . 22
3.2 Künstliche neuronale Netze . . . 23
3.2.1 Aufbau . . . 24
3.2.2 Schichttypen . . . 24
3.2.3 Optimierung . . . 30

3.3 Random-Forest-Klassifikatoren . . . 32
3.3.1 Entscheidungsbäume . . . 32
3.3.2 Bagging . . . 34
3.3.3 Aufbau und Eigenschaften . . . 35
3.4 Stützvektormaschinen . . . 35
3.4.1 Einführung . . . 35
3.4.2 Aufbau . . . 36
3.4.3 Schlupfvariablen . . . 37
3.4.4 Kernelfunktionen . . . 37
3.4.5 Multiklassenproblem . . . 38
3.5 Hybride Methoden und Transferlernen . . . 39
3.5.1 Hybride Methoden . . . 39
3.5.2 Transferlernen . . . 41

4 Evaluierung **43**
4.1 Versuchsaufbau . . . 43
4.1.1 Teststruktur . . . 43
4.1.2 Merkmalsextraktion . . . 44
4.1.3 Verwendete Datensätze . . . 44
4.1.4 Kreuzvalidierung und Aufteilung . . . 45
4.1.5 Generierte Augmentierungen . . . 46
4.1.6 Bewertungskriterien . . . 47
4.1.7 Klassifikatormodelle . . . 48
4.2 Experimente . . . 50
4.2.1 Vergleich von hybriden und nicht-hybriden Methoden . . . 50
4.2.2 Vergleich von Augmentierungskategorien . . . 52
4.2.3 Generalisierungstest . . . 56
4.2.4 Kombinierung der Datensätze . . . 58
4.2.5 Vergleich von Merkmalsextraktionsschichten der hybriden Methoden . . . 60
4.2.6 Vergleich auf verrauschten Daten des MIDI-Datensatzes . . . 61
4.2.7 Weitere relevante Ergebnisse . . . 64

5 Fazit und Ausblick **67**
5.1 Fazit . . . 67
5.2 Ausblick . . . 69
5.2.1 Teststrukturen und Datensätze . . . 69

5.2.2 Weitere Schichten und Methoden künstlicher neuronaler Netze 69
5.2.3 Augmentierungs- und Merkmalsselektion 71
5.2.4 Hybride Kombinationsmöglichkeiten 72
5.2.5 Weiterverwendung der Instrumentenerkennung 72

A Anhang **73**
A.1 Verwendete Bibliotheken 73
A.2 Ablauf der Merkmalsberechnung 74
A.3 Greedy-Algorithmus zur repräsentativen Kreuzvalidierungsaufteilung 75
A.4 Architektur des künstlichen neuronalen Netzes 76
A.5 Audio Degradation Toolbox - Raumhalleffekt-Datei 77
A.6 Vergleiche von Hyperparametern 78
A.6.1 Batch-Normalisierung 78
A.6.2 NN-SVM C-Parameter 78
A.6.3 Melspektrogramme 78
A.7 Modellvergleiche auf einzelnen Instrumenten 79
A.7.1 MIDI-Datensatz 79
A.7.2 Akkord-Datensatz 81
A.8 P-Werte beim Vergleich der Klassifikatormodelle 83
A.8.1 MIDI-Datensatz 83
A.8.2 Akkord-Datensatz 85
A.9 Tabellarische Vergleiche der Aumentierungskategorien 87
A.10 Generalisierungstestergebnisse mit verschiedenen Augmentierungskategorien 90
A.11 Vergleich des kombinierten Datensatzes auf einzelnen Instrumenten . 91
A.12 Vergleich von Merkmalsextraktionsschichten auf einzelnen Instrumenten 94
A.13 Weitere Ergebnisse 96
A.13.1 Vergleich auf verrauschten Daten des Akkord-Datensatzes . . 96
A.13.2 Trainingsdauer auf dem Akkord-Datensatz 97
A.14 Visualisierter Ausblick über Attention und Autoencoder 98
A.14.1 Attention 98
A.14.2 Autoencoder 98

Abbildungsverzeichnis **99**

Abkürzungsverzeichnis **101**

Literaturverzeichnis **103**

1 Einleitung

Auf ca. 50.000 Jahre wird das momentan älteste Instrument der Welt geschätzt [92]. Seither hat sich die Entwicklung von Instrumenten und Musik in der Geschichte der Menschheit immer weiter fortgesetzt. Es entstanden neue Instrumente, sowie eine weiterführende Erschließung der Musiktheorie mit einhergehender Notation von musikalischen Tönen und Kompositionen. Mit der Entwicklung von Computern und Synthesizern begann Mitte des 20. Jahrhunderts auch der Werdegang der digitalen Musikgenerierung und -analyse. Seither wurden verschiedene Repräsentationen und Klassifikatoren zur automatisierten Analyse von musikalischen Stücken entworfen und eingesetzt.

Innovation	Jahr	Fortschritt
Linear Predictive Coding	1969	Automatische, einfache Sprachkompression
Dynamische Zeitnormierung	1970er	Reduzierung der Suchkomplexität bei gleichzeitiger temporaler Flexibilität
Hidden Markov Modelle	1975	Temporale und spektrale Variationen können statistisch erfasst werden
Mel-Frequenz-Cepstrum-Koeffizienten	1980	Verbesserte auditiv-basierte Sprachkompression
Sprachmodelle	1980er	Eingebundene Sprachredundanz verbessert Erkennungsraten
Künstliche neuronale Netze	1980er	Exzellenter statischer nichtlinearer Klassifikator
Kernelbasierte Klassifikatoren	1998	Besseres charakteristisches Training
Dynamische bayessche Netze	1998	Allgemeinere statistische Netze
Tiefe neuronale Faltungsnetze	2012	Komprimierte und translationsinvariante Netze

Tabelle 1.1: Historische Fortschritte der Spracherkennung. Die Jahreszahlen geben in manchen Fällen nur eine Näherung an, da die Akzeptanz und Anwendung neuer Technologien erst über die Zeit stattfindet (nach [72, Tabelle 2]).

In Tabelle 1.1 ist eine kurze historische Zusammenfassung von einigen Innovationen der thematisch nahen Spracherkennungssysteme zu sehen, welche zum Teil auch in der Erkennung von Musikdaten zum Einsatz kommen.
Mit der Entwicklung von tiefen neuronalen Faltungsnetzen (siehe Abschnitt 3.2) und den seither eingeführten Methoden hat die Forschung im letzten Jahrzehnt vor allem in der Bilderkennung [48], aber unter anderem auch in der Erkennung von Audiodaten nochmals deutliche Fortschritte hervorgebracht [43, K. 5.5].

1.1 Motivation

Instrumentenerkennung kann zum einen zur automatischen Annotierung von Musikstücken und somit auch als Grundlage für Empfehlungssysteme verwendet werden. Zum anderen können durch Informationen aus der Instrumentenerkennung auch andere Eigenschaften wie das Genre [78] besser klassifiziert werden. Außerdem können darauf aufbauend Klassifikatoren eingesetzt werden, welche auf einzelne Instrumente spezialisiert sind [41].

Zusammen mit den tiefen neuronalen Faltungsnetzen, widmet sich diese Arbeit dem Themengebiet der hybriden Methoden und somit der Kombination von Klassifikatoren. Während sich über einzelne Anwendungen von Klassifikatoren bereits viele Arbeiten zur Musik- und Instrumentenerkennung finden (siehe Abschnitt 1.2), so ist insbesondere der in dieser Arbeit verwendete hybride Ansatz des Transferlernens zum Zeitpunkt der Erstellung dieser Arbeit in der Literatur nicht zu finden und wird dementsprechend ausführlich untersucht.
Ein weiterer wichtiger Aspekt der Arbeit ist die Verarbeitung von Informationen auf Basis verrauschter Musikdaten. Es werden aufgrund der besonderen Eigenschaften von verrauschten Daten auch entsprechende Methoden in die Designentscheidungen bei der Konstruktion des Erkennungssystems miteinbezogen.

1.2 Verwandte Arbeiten

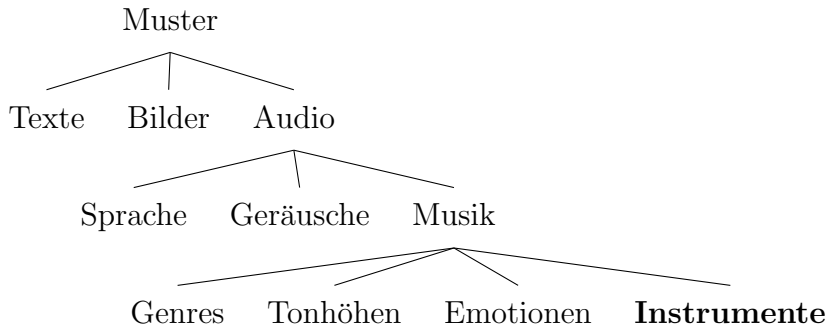


Abbildung 1.1: Einordnung des Themas dieser Arbeit in den allgemeinen Kontext der automatisierten Erkennung von Mustern.

Wie in Abbildung 1.1 zu sehen, gehört die Instrumentenerkennung als Teilgebiet zur Musikerkennung und zum allgemeinen Gebiet der Audioerkennung. In den folgenden Abschnitten werden Arbeiten der beiden thematisch verwandten Oberkategorien

vorgestellt, sowie dem Kontext dieser Arbeit zugeordnet. Abschließend werden auch explizit verwandte Arbeiten aus dem Bereich der Instrumentenerkennung eingeführt.

1.2.1 Audioerkennung

Ein Themengebiet, welches der Musikerkennung thematisch nah angesiedelt ist, ist die Spracherkennung. Bekannte Spracherkennungssysteme wie „Deep Speech“ [36][3] wenden in ihren Systemen ebenfalls künstliche neuronale Netze an. Während Artikel wie [80] vor allem die Architektur und Parameter von Spracherkennungssystemen optimieren, so werden auch wie in [105] Vergleiche zwischen künstlichen Spracherkennungssystemen und menschlichen Leistungen bei der Erkennung von Sprache angestellt. Durch die Erkennung von verrauschten Sprachdaten mit neuronalen Faltungsnetzen sind auch in [76] Bezüge zum Thema dieser Arbeit zu finden.
Auch in der thematisch ebenfalls ähnlichen Erkennung von Geräuschen werden künstliche neuronale Netze, zusammen mit dem auch in dieser Arbeit verwendeten Ansatz der Datenaugmentierungen, angewendet [81]. Die in dieser Arbeit verwendete Form der Normalisierung (siehe Abschnitt 2.3.3) ist ebenfalls in Ausarbeitungen zur Geräuscherkennung wiederzufinden [55][108].

1.2.2 Musikerkennung

Das Themenfeld Musikerkennung umfasst neben der behandelten Instrumentenerkennung auch musikalische Eigenschaften wie Genre, Tonhöhe und Emotionen.

Genre

Zur Erkennung des Genres werden in [17] zunächst mithilfe von Faltungsschichten (siehe Abschnitt 3.2.2) lokale Merkmale extrahiert und anschließend mit rekurrenten Schichten (siehe Abschnitt 3.2.2) zusammengefasst. In [20] werden ebenfalls Faltungsschichten zur Genreerkennung angewendet. Wie auch in der vorliegenden Arbeit wird dort der Merkmalsraum über Spektrogramme gebildet. Die Klassifikationsleistung wird anschließend mit den Ergebnissen einer Stützvektormaschine (siehe Abschnitt 3.4) auf ausgewählten Merkmalen verglichen. Eine weiterführende Verwendung der vorliegenden Arbeit kann wie in [78] erfolgen, sodass die erkannten Instrumente als Merkmale herangezogen werden, um anschließend das Genre von vorliegenden Stücken besser zu klassifizieren.

Tonhöhe und Emotionen

In [41] wird ebenfalls mit künstlichen neuronalen Netzen eine Erkennung der Tonhöhe von Gitarrenklängen durchgeführt. Die in [17] zur Genreerkennung verwendete Kombination aus Faltungs- und rekurrenten Schichten wird auch in [56] zur Erkennung von musikalischen Emotionen verwendet.

1.2.3 Instrumentenerkennung

Das in dieser Arbeit untersuchte Thema der Instrumentenerkennung ist auch in Vorarbeiten bereits zahlreich behandelt worden.
In [35] wurde die Architektur eines künstlichen neuronalen Netzes zur Instrumentenerkennung entwickelt. Sowohl die Architektur, als auch die verwendeten Mel-Spektrogramm-Merkmale (siehe Abschnitt 2.2.1) werden in dieser Arbeit ähnlich verwendet. Eine weitere Abschlussarbeit [84], welche die in [35] entwickelte Herangehensweise anwendet, ist ebenfalls dem Themengebiet der Instrumentenerkennung zuzuordnen. Dort wird neben der Instrumentenerkennung zusätzlich eine Überlagerungsreduzierung von einzelnen Instrumenten implementiert. Auch in [86] wird auf Basis von [35] mit Mel-Spektrogrammen und neuronalen Faltungsnetzen eine mehrfache Instrumentenerkennung durchgeführt. [34] ist eine weitere Untersuchung mit Referenz zu [35], welche zusätzlich die, auch in dieser Arbeit in anderer Form verwendete, Methodik des Transferlernens (siehe Abschnitt 3.5.2) anwendet. In [71] werden zudem Faltungsnetze auf CQT-transformierten Spektrogrammen zur Erkennung einzeln spielender Instrumente angewendet und anschließend mit weiteren Klassifikatoren verglichen.
Neben neuronalen Faltungsnetzen wird auch die Stützvektormaschine im Bereich der Instrumentenerkennung in [25] eingesetzt. Neben der Erkennung von mehreren Instrumenten wird dort auch ein Vergleich verschiedener Merkmale angestellt.
Der in dieser Arbeit verwendete Random-Forest-Klassifikator (siehe Abschnitt 3.3) kann ebenfalls zur Instrumentenerkennung eingesetzt werden. In [95] und [96] wird mithilfe des Random-Forest-Klassifikators eine Merkmalsselektion für einzelne Instrumente durchgeführt. Weiterhin wird in [95] die Generalisierung auf westlichen und ethnischen Instrumenten getestet. Der in [96] verwendete Datensatz, welcher gemischte Akkorde von Instrumenten ethnischer und westlicher Art enthält, wird unter anderem auch in der vorliegenden Arbeit zur Evaluierung herangezogen (siehe Abschnitt 4.1.3).

1.2.4 Bezug zur Arbeit

Bei der Analyse von Audiodaten kommen auch hybride Ansätze [2] zur Anwendung, welche künstliche neuronale Netze mit weiteren Klassifikatoren kombinieren, um die Klassifikationsleistung zu verbessern. Des Weiteren werden Methoden wie Datenaugmentierungen in weiteren Ausarbeitungen zur Erkennung von Geräuschen [81] und Sprache [21][46] verwendet. Auch die in dieser Arbeit verwendete Form der Normalisierung findet in der robusten Erkennung von Geräuschen Anwendung [55][108].

1.3 Aufbau der Arbeit

Das einführende Kapitel 2 behandelt die Vorverarbeitung von Audiodaten. Darin wird zunächst in die Audiosignale, sowie deren Repräsentationen in den Abschnitten 2.1 und 2.2 eingeführt. Die nachfolgenden Abschnitte 2.3 und 2.4 behandeln abschließend die Normalisierung von Audiomerkmalen, sowie Datenaugmentierungen zur künstlichen Erweiterung von Datensätzen.
In Kapitel 3 wird das Themenfeld der Klassifikatoren behandelt. Nach einer kurzen Einführung in Abschnitt 3.1 werden zunächst künstliche neuronale Netze in Abschnitt 3.2 vorgestellt. Es wird außerdem in den folgenden zwei Abschnitten 3.3 und 3.4 in die Grundlagen des Random-Forest-Klassifikators und der Stützvektormaschine eingeführt. Der thematische Abschluss der Klassifikatorgrundlagen wird in Abschnitt 3.5 mit den titelgebenden hybriden Methoden, sowie der thematisch nahen Methodik des Transferlernens gezogen.
Die in dieser Arbeit durchgeführte Evaluierung wird in Kapitel 4 beschrieben. Es wird zunächst in Abschnitt 4.1 der Versuchsaufbau dargestellt und folgend die Experimente in Abschnitt 4.2 ausgewertet und diskutiert.
Abschließend wird in Kapitel 5 das Fazit der Arbeit auf Basis der Testergebnisse gezogen, sowie ein Ausblick über fortführende Themen gegeben.

2 Grundlagen der Audiosignalverarbeitung

2.1 Audiosignale

Audiosignale bestehen aus kontinuierlichen Luftdruckschwingungen, welche beispielsweise durch Mikrofone aufgefangen werden können. In vielen Aspekten, wie der in Abschnitt 2.2.1 vorgestellten Frequenzzerlegung, wird sich bei der digitalen Audiosignalverarbeitung an dem natürlichen Vorbild des menschlichen Ohres orientiert, welches in Abbildung 2.1 skizziert zu sehen ist.

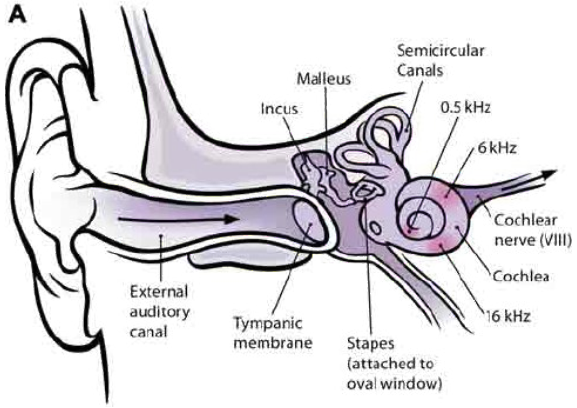


Abbildung 2.1: Der skizzierte Aufbau des menschlichen Ohres [16].

Zur digitalen Verarbeitung muss das anliegende kontinuierliche Audiosignal zunächst diskretisiert werden. Zur Diskretisierung kommen Verfahren wie die Pulse-Code-Modulation zum Einsatz, welche das kontinuierliche Eingabesignal mit konstanter Abtastrate, also einem vordefinierten zeitlichen Versatz, approximiert [52]. Um die Information eines kontinuierlichen Signals bei der Abtastung zu erhalten, muss die Abtastrate nach dem Nyquist-Shannon-Theorem mindestens doppelt so hoch gewählt werden, wie der höchste vorkommende Frequenzanteil des zu diskretisierenden Eingabesignals. Werden die Grenzen des Abtasttheorems nicht eingehalten kann es,

wie in Abbildung 2.2 zu sehen, zu unerwünschten Resultaten wie dem Alias-Effekt kommen.

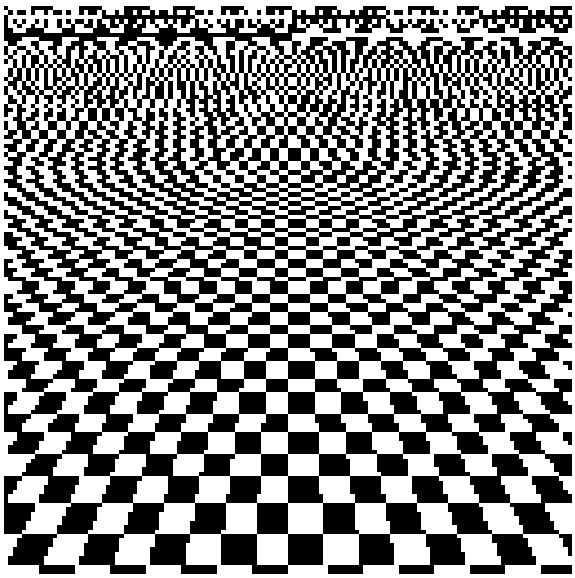


Abbildung 2.2: Eine Schachbretttextur, mit zunehmenden Auswirkungen des Alias-Effekts von unten nach oben [101].

Zusätzlich zur Abtastung folgt bei der Pulse-Code-Modulation eine Quantisierung der abgetasteten Werte auf einen diskreten und endlichen Wert. Das Ergebnis der Diskretisierung lässt sich entweder direkt weiter verarbeiten oder als Datenbasis in einer Audiodatei abspeichern.

2.2 Repräsentationen

Auch wenn die aktuelle Forschung von tiefen neuronalen Netzen (siehe Abschnitt 3.2) Modelle wie *WaveNet* [69] erfolgreich auf Audiosignalen ohne Vorverarbeitung anwenden, so wird in der Regel beim Aufbau eines Mustererkennungssystems auf entwickelte Audiorepräsentationen zurückgegriffen, welche die Audiosignale in eine kompaktere und strukturierte Repräsentation überführen, um ein effizienteres Lernen zu ermöglichen.

2.2.1 Spektrale Merkmale

Die Basis von spektralen Merkmalen stellen Spektrogramme dar, welche im folgenden Abschnitt eingeführt werden. Die weitere Verarbeitung der Spektrogramme kann mit einer Transformation und Skalierung erfolgen (siehe Abschnitt 2.2.1) oder als

Basis für weitere Merkmale dienen, welche eine geringere Dimensionalität und somit eine höhere Informationsdichte im Vergleich zu Spektrogrammen aufweisen. Da diese weiteren Merkmale in dieser Arbeit nicht verwendet werden, für fortführende Arbeiten aber dennoch von Relevanz sein können, werden diese in Abschnitt 2.2.2 kurz vorgestellt.

Spektrogramme

Bei der Berechnung von Spektrogrammen wird das Audiosignal in eine Repräsentation transformiert, welche Datenpunkte auf einer Zeit-, Frequenz- und Magnitudenachse darstellt und folglich der transformierte Raum 3 Dimensionen aufweist. Das Spektrogramm kann, wie in Abbildung 2.3 zu sehen, in einem 2-dimensionalen Koordinatensystem mit einer farblichen Magnitude dargestellt werden. Die mathematische Grundlage bei der Erstellung eines Spektrogramms bildet dabei die diskrete Form der Fourier-Transformation, welche Signale in ein Frequenzspektrum abbildet.

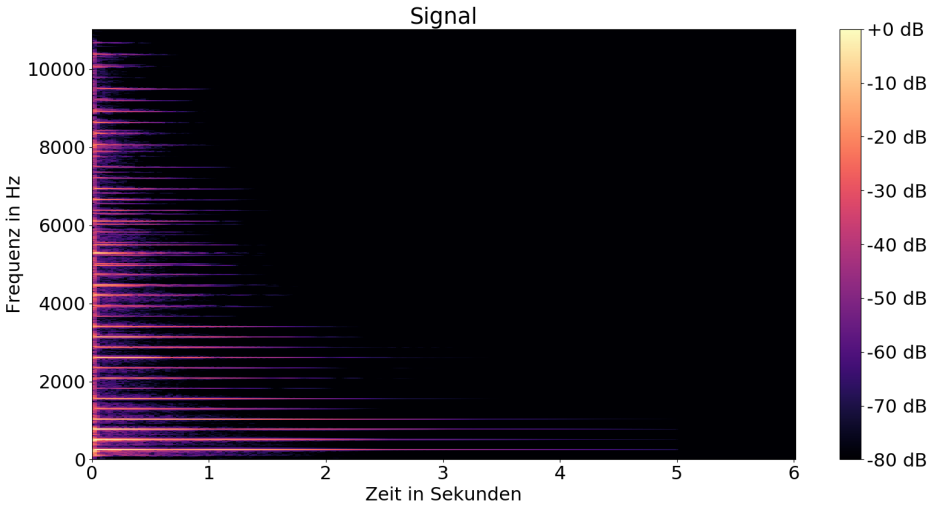


Abbildung 2.3: Das 6-sekündige Spektrogramm eines Gitarrenanschlags.

Da die stationäre (diskrete) Fourier-Transformation keine Informationen über zeitliche Veränderungen im Frequenzspektrum enthält, wird im Bereich der Audiodatenanalyse häufig die sogenannte Kurzzeit-Fourier-Transformation (STFT) angewendet [65, S. 1090]. Bei der STFT wird ein Fenster mit einer vorher definierten Fenster- und Schrittgröße über das Signal geschoben. Es ergibt sich bei einer Fensterfunktion

w mit Fenstergröße N und Schrittgröße $N/2$ und zusätzlicher Anzahl T an zeitlichen Containern und K Frequenzcontainern die STFT wie folgt:

$$\begin{aligned} t &\in [0 : T-1] \\ k &\in [0 : K] \\ X(t,k) &= \sum_{n=0}^{N-1} w(n)x(n + t \cdot N/2)e^{-j2\pi kn/N} \end{aligned} \tag{2.1}$$

Wobei der entsprechende Frequenzwert von Container k bei einer Abtastrate von f_s Hz definiert ist als:

$$f(k) = \frac{k}{N} \cdot f_s \tag{2.2}$$

Die in dieser Arbeit verwendete Abtastrate beträgt 16.000 Hz. Das berechnete Magnituden-Spektrogramm kann zudem mit 2 potenziert werden, um ein Leistungsspektrogramm zu erhalten [30, K. 1].

Mel-Spektrogramme

Die vorliegende Arbeit verwendet Mel-Spektrogramme (MS) als Merkmalsrepräsentation. Die Grundlage für Mel-Spektrogramme ist erneut auf die natürliche Audioverarbeitung des Menschen und insbesondere die psychoakustisch wahrgenommene Tonhöhe zurückzuführen. Eingeführt wurde die Mel-Skala bereits 1937 als Ergebnis einer Forschung, welche das Phänomen untersuchte, dass Menschen hohe Töne schlechter unterscheiden konnten und insbesondere die Tonhöhenwahrnehmung nicht linear verläuft [89]. Die Mel-Skala kann mit der folgenden Formel approximiert werden, wobei f die Frequenz eines Tones in Hz angibt [70, S. 150]:

$$m = 2595 \cdot log_{10}(1 + \frac{f}{700}) \tag{2.3}$$

Die Anwendung eines Mel-Filters auf ein Spektrogramm transformiert somit die Daten logarithmisch. Analog zur STFT findet außerdem eine Einordnung in eine vorher definierte Anzahl an Frequenzcontainern statt. Die unterschiedlichen Skalierungen sind in Abbildung 2.4 visualisiert.

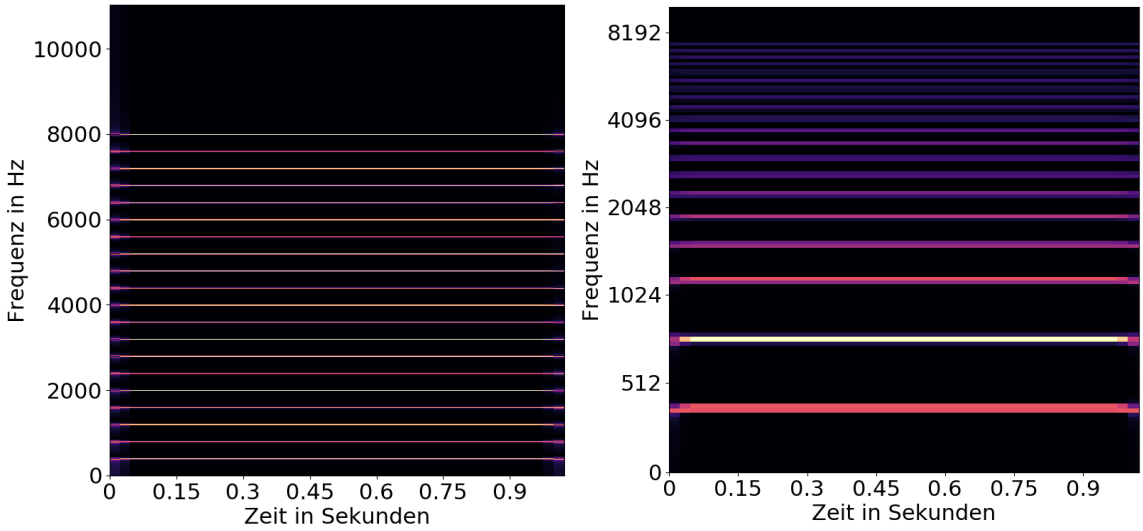


Abbildung 2.4: Die unterschiedliche Skalierung im Spektrogramm (links) und Mel-Spektrogramm (rechts) von 20 Sinustönen mit jeweils 400 Hz Unterschied.

2.2.2 Weitere Merkmale

Neben den Spektrogrammen gibt es auch weitere Merkmale, welche aus spektralen und rhythmischen Eigenschaften des vorliegenden Audioausschnitts berechnet werden können.

Mel-Frequenz-Cepstrum-Koeffizienten

Die Mel-Frequenz-Cepstrum-Koeffizienten (MFCC) werden nach den folgenden Schritten berechnet:

- Berechnung des Mel-Leistungsspektrogramms und gleichzeitige Dimensionsreduktion durch Abbildung auf Mel-Frequenzbänder (siehe Abschnitt 2.2.1),
- Logarithmierung des Mel-Leistungsspektrogramms,
- Dekorellation beispielsweise durch eine diskrete Kosinustransformation [79].

MFCCs werden insbesondere in der Sprach-, Musik- und Instrumentenerkennung verwendet [66][96].

Mittelwertbildung

Zur Dimensionsreduktion kann eine Mittelwertbildung beispielsweise in Form des quadratischen Mittels (*Root Mean Square* (RMS)) eingesetzt werden, welches für

jeden zeitlichen Schritt die mittlere Energie berechnet. Ein weiteres Merkmal ist der spektrale Zentroid, welcher auf dem normalisierten Magnituden-Spektrogramm S mit Frequenzbändern $freq$ für jedes Zeitelement t wie folgt definiert ist [45, K. 5]:

$$zentroid[t] = \sum_k S[k,t] \cdot \frac{freq[k]}{\sum_j S[j,t]} \tag{2.4}$$

Transferlernen

Neben speziell entwickelten Merkmalsrepräsentationen kann auch auf eine Merkmalsberechnung mithilfe künstlicher neuronaler Netze und des Prinzips des Transferlernens angewendet werden. Da diese Methodik auch in der vorliegenden Arbeit angewendet wird, findet sich in Abschnitt 3.5.2 eine genauere Ausführung der Thematik.

2.3 Normalisierung

Ein wichtiger Schritt bei der Vorverarbeitung von Daten, welche als Merkmale für einen Klassifikator verwendet werden, ist die Normalisierung. So kann beispielsweise der Trainingsprozess künstlicher neuronaler Netze mit normalisierten Daten deutlich effizienter ablaufen [42]. Da verschiedene Datensätze mitunter verschiedene Skalierungen oder Versätze haben können, kann eine normalisierende Vorverarbeitung zusätzlich einen positiven Effekt auf die Generalisierung eines Modells haben [40]. Im Folgenden werden verschiedene Normalisierungsmethoden näher vorgestellt.

2.3.1 Z-Wert-Normalisierung

Bei der Z-Wert-Normalisierung wird zunächst für jedes Merkmal das Mittel μ_i und die Standardabweichung δ_i berechnet und folgend ergibt sich der Normalisierungsterm zu:

$$x' = \frac{x_i - \mu_i}{\delta_i} \tag{2.5}$$

Die resultierenden Daten haben einen Mittelwert von 0 und die Einheitsvarianz als Eigenschaften. Vor allem die negativen Effekte von Ausreißern werden durch diese Art der Normalisierung reduziert [42, S. 3].

2.3.2 Min-Max-Normalisierung

Für die Min-Max-Normalisierung werden alle Merkmale linear anhand ihres Minimums und Maximums auf einem definierten Intervall, üblicherweise [0,1] oder [-1,1], skaliert. Nach Entfernung von Merkmalen mit konstanten Werten ($x_{max} - x_{min} = 0$) ergibt sich der Normalisierungsterm wie folgt:

$$x' = (x_{max} - x_{min}) \times \frac{x_i - x_{min}}{x_{max} - x_{min}} + x_{min} \tag{2.6}$$

2.3.3 Pro-Kanal-Energienormalisierung

Der in dieser Arbeit verwendete Normalisierungsansatz ist die Pro-Kanal-Energienormalisierung (*Per-Channel Energy Normalization*, PCEN). Die PCEN ist eine speziell für Audiodaten entwickelte Normalisierungstechnik, welche eine automatische Verstärkungsregelung (*Automatic Gain Control*, AGC) und Dynamikkompression (*Dynamic Range Compression*, DRC) der Audiosignale implementiert. Unterschiede der durch logarithmische Normalisierung und PCEN berechneten Transformation sind in Abbildung 2.5 visualisiert.

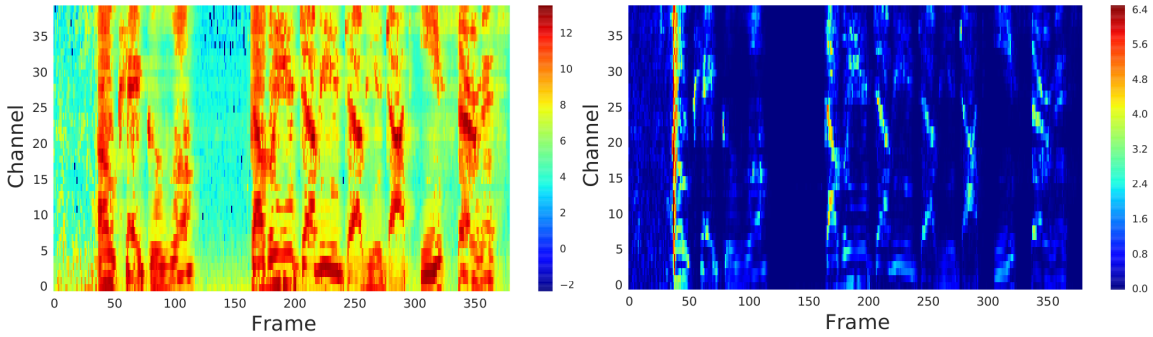


Abbildung 2.5: Unterschiedliche Visualisierung eines log-skalierten Mel-Spektrogramms (links) und PCEN-skalierten Mel-Spektrogramms (rechts) [98].

Während die DRC vor allem die Varianz der Vordergrundlautstärke verringert, wirkt sich die AGC positiv auf die Reduzierung des stationären Hintergrundrauschens aus [54]. Folglich erreichen Modelle, welche unter anderem die PCEN-Skalierung verwenden, in verrauschten Audiodaten eine konsistent bessere Klassifikationsleistung als Modelle, welche ihre Merkmale aus log-skalierten Mel-Spektrogrammen ziehen, sowohl in der Musikerkennung [55, S. 25] als auch in der Geräuscherkennung [76].
Der PCEN-Term beinhaltet die Parameter $\alpha \in [0, 1]$, $\epsilon \ll 0, 1$, r und δ. Sei E das eingegebene Mel-Spektrogramm und M eine, durch einen Filter mit unendlicher Impulsantwort und Glättungskoeffizienten s definierte, geglättete Version von

E, so ergeben sich für einen Zeitpunkt t und eine Mel-Frequenz f die folgenden Definitionen von M und folglich $PCEN$:

$$\begin{aligned} M(t,f) &= (1-s)M(t-1,f) + sE(t,f) \\ PCEN(t,f) &= \left(\frac{E(t,f)}{(\epsilon + M(t,f))^{\alpha}} + \delta\right)^{r} - \delta^{r} \end{aligned} \tag{2.7}$$

Auch wenn die PCEN im Vergleich zu anderen Normalisierungsmethoden viele teils frequenzabhängige Parameter aufweist, so sind diese interpretierbar und auf spezielle Charakteristiken der Daten einstellbar, beispielsweise: ruhig/aktiv (ϵ), stationär/temporär (s) oder leise/laut (δ) [54, K. 5]. Des Weiteren sind die in PCEN verwendeten Parameter ableitbar und können somit folglich, wie in [98] vorgeschlagen, als trainierbare Schicht eines künstlichen neuronalen Netzes (siehe Abschnitt 3.2) implementiert werden. Die Funktionalität einer PCEN-Schicht ist im Rahmen dieser Masterarbeit nicht implementiert worden, wird aber im Ausblick in Abschnitt 5.2.2 diskutiert.

2.4 Datenaugmentierungen

Datenaugmentierungen werden verwendet, um die Menge der Trainingsdaten künstlich zu erweitern und so eine bessere Generalisierung des Modells zu erhalten (siehe Klassifikatorkapitel 3). Die bereits vorliegenden Daten werden mithilfe geeigneter Transformationen verändert und als eigenständige Beispiele weiter verwendet. Bei jeglichen Datenaugmentierungen ist es wichtig zu beachten, dass sie weiterhin eine sinnvolle Repräsentation des Beispiels darstellen.

2.4.1 Allgemeine Augmentierungen

Einige mögliche Augmentierungen auf Bilddaten sind [85, S. 7-16]:

- Spiegelung: Das Bild wird horizontal gespiegelt (siehe Abbildung 2.6),
- Farbraumveränderung: Die Farbwerte des Bildes werden verändert, beispielsweise durch Erhöhung von Kontrast, Transformation in den Graubildraum oder Helligkeitsveränderung,
- Rotation: Das Bild wird in einem vorgegebenen Winkel gedreht,
- Rauschen: Das Bild wird mit einem Rauschsignal überlagert (siehe Abschnitt 2.4.2),

- Zufälliges Löschen: Ähnlich zum Rauschen und inspiriert von der in Abschnitt 3.2.2 vorgestellten Technik des *Dropouts* werden zufällig Pixel des Eingabebildes entfernt.

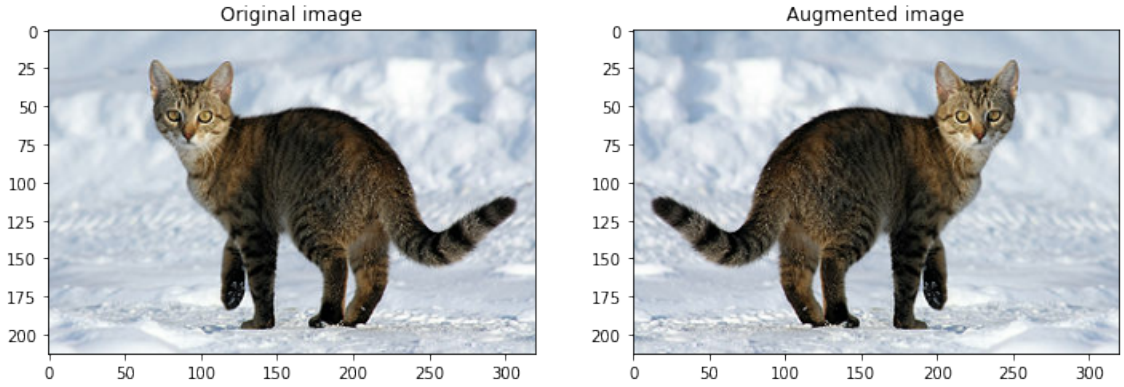


Abbildung 2.6: Erweiterung der Trainingsdaten durch Spiegelung (rechts) des Originalbildes (links) [91].

Des Weiteren sind künstliche Erweiterungen mit speziellen Transformationen möglich, welche darauf abzielen die Optimierungskriterien des Klassifikators zu nutzen, um daraus marginale Änderungen des Bildes herzuleiten. Das wie in Abbildung 2.7 gezeigte, transformierte Bild wird dadurch mit hoher Konfidenz fehlklassifiziert, erhält aber, für einen Menschen klar zu sehen, weiterhin die originale Bildinformation und kann so als neues Trainingsbeispiel genutzt werden [32]. Zur Generierung dieser künstlichen Augmentierungen kommen die im Ausblick in Abschnitt 5.2.2 thematisierten *Generative Adversarial Networks* zum Einsatz.

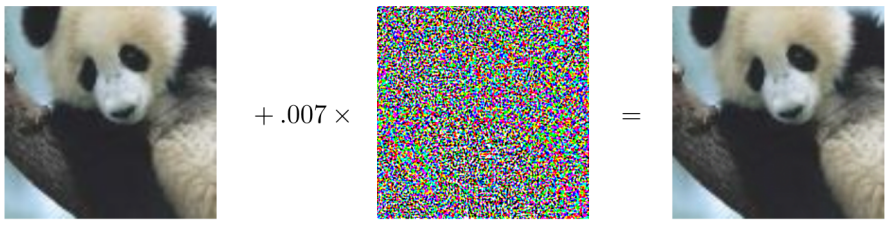


Abbildung 2.7: Das Originalbild (links) wird mit einem geringen Anteil an Rauschinformationen (mittig) überlagert und folglich wird das transformierte Bild (rechts) mit hoher Konfidenz fehlklassifiziert [32, S. 3].

Wie in Abbildung 2.8 zu sehen kann durch künstliche Datenaugmentierungen die Klassifikationsleistung, gemessen an dem Validierungsfehler, verbessert werden.

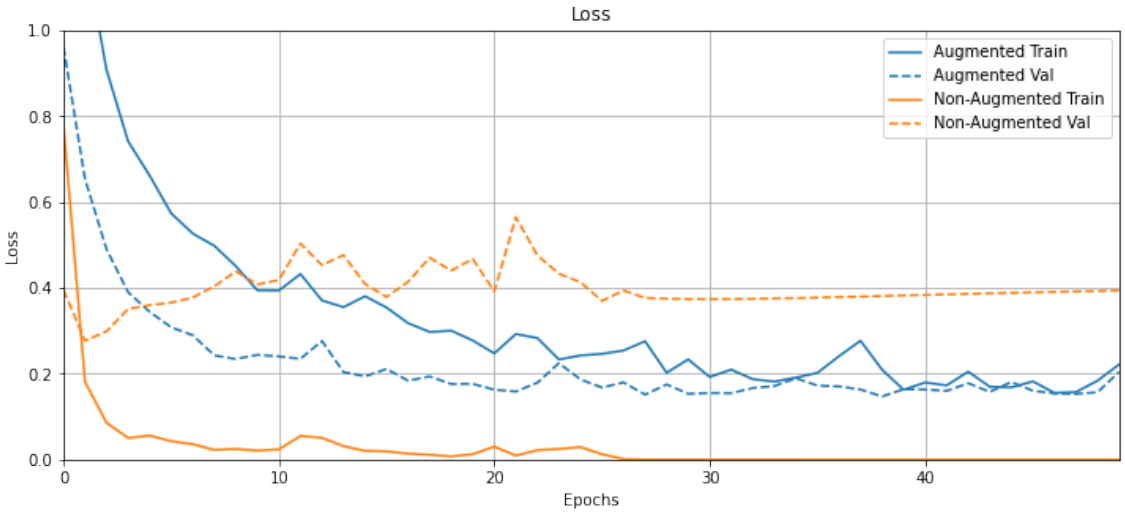


Abbildung 2.8: Vergleich von Trainingsleistung (durchgezogen) und Validierungsleistung (gestrichelt) auf Daten mit (blau) und ohne (gelb) Augmentierungen [91].

2.4.2 Audioaugmentierungen

Während die im vorherigen Abschnitt beschriebenen Datenaugmentierungen allgemein für Bilder gültig sind, so werden in dieser Arbeit PCEN-skalierte Mel-Spektrogramme als Merkmale verwendet. Diese können zwar ebenfalls als Bild aufgefasst und dargestellt werden, allerdings existieren auch einige wichtige Unterschiede, weshalb eine direkte Anwendung der Bildaugmentierungen auf Spektrogramme nicht sinnvoll ist. Zum einen haben die Achsen eines Spektrogramms unterschiedliche Bedeutungen und insbesondere Spiegelungen und Rotationen verändern die elementaren Bestandteile des Beispiels wie den Frequenzverlauf oder die einzelnen Schwingungen. Deshalb werden in den folgenden zwei Abschnitten spezielle Augmentierungsarten für Audiodaten vorgestellt, welche die Audiodaten vor der Berechnung der Spektrogramme transformieren.

Verzerrungen

Verzerrungen des Eingabesignals haben verschiedene Auswirkungen auf die Eigenschaften. Im Folgenden sind die Verzerrungsarten kurz ausgeführt.

Amplitude:

Bei einer Skalierung mit einem Wert α auf der Amplitudenachse wird das eingegebene Signal für $\alpha > 1$ lauter und für $\alpha < 1$ leiser.

Zeitachse:

Skaliert man die Zeitachse mit einem Faktor β so wird das Signal für $\beta < 1$ schneller

und erhält durch die einhergehende Skalierung der Frequenzen eine höhere Tonlage. Dem gegenüberstehend wird das Signal für $\beta > 1$ langsamer und tonal tiefer.

Tonhöhe:

Um die Tonhöhe eines Audiosignals zu verändern wird zunächst das Spektrogramm des Signals mittels Fourier-Transformation berechnet (siehe Abschnitt 2.2.1). Anschließend wird eine Translation des Spektrogramms auf der Frequenzachse ausgeführt. Wird das Nyquist-Shannon-Theorem (siehe Abschnitt 2.1) beachtet, lässt sich aus dem Spektrogramm das originale, aber tonal veränderte Signal rekonstruieren.

Rauschen und Kompression

Neben Verzerrungen ist das Hinzufügen von Rauschanteilen eine weitere Möglichkeit sinnvolle Datenaugmentierungen eines Audiosignals zu erstellen. Von Rauschen spricht man, wann immer ein Informationssignal durch ein weiteres (Rausch-)Signal überlagert wird, welches in der Regel die korrekte Auswertung von Informationen aus dem Gesamtsignal erschwert. Gemessen wird der Anteil des Informationsanteils in dem sogenannten Signal-Rausch-Verhältnis (*Signal-Noise-Ratio*, SNR) und ist wie folgt definiert:

$$SNR = \frac{Nutzleistung}{Rauschleistung} \tag{2.8}$$

In Analogie zu Lichtspektren werden Rauscharten je nach enthaltenen Frequenzanteilen verschiedenen „Farben“ zugeordnet. Einige Rauschfarben mit ihrer jeweiligen Proportionalität zur spektralen Leistungsdichte sind in Tabelle 2.1 zu sehen.

Farbe	Proportionalität zur spektralen Leistungsdichte
Violett	f^2
Blau	f
Weiß	1
Pink	$1/f$
Rot	$1/f^2$

Tabelle 2.1

Die drei in dieser Masterarbeit verwendeten Augmentierungen, welche der Kategorie „Rauschen und Kompression“ zugeordnet sind, werden im Folgenden aufgeführt.

Weißes Rauschen:

Wie in Tabelle 2.1 zu sehen hat weißes Rauschen eine konstante und frequenzunabhängige Intensität und somit auch ein konstantes Leistungsdichtespektrum [57].

In Abbildung 2.9 ist ein mit weißem Rauschen verändertes Audiosignal visualisiert.

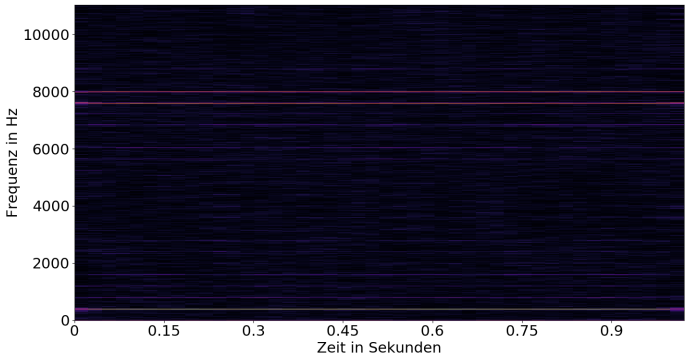


Abbildung 2.9: Der bereits in Abbildung 2.5 verwendete Audioausschnitt, überlagert mit weißem Rauschen mit $SNR = 20$.

Dynamikkompression:
Bei der Dynamikkompression werden die verschiedenen Lautstärkepegel eines Audiosignals mit einem „Kompressor“ zeitlich angeglichen und somit der Dynamikumfang des Signals komprimiert. Parameter der Dynamikkompression sind:

- Schwellenwert: Der Grenzwert an welchem der Kompressor beginnt das Signal zu bearbeiten.
- Ein-/Ausgangsverhältnis: Das Verhältnis in welchem (den Schwellenwert überschreitende Signale) heruntergepegelt werden.
- Ausgangsverstärker: Um Absenkungen des Gesamtpegels entgegenzuwirken wird das ausgehende Signal unabhängig vom Schwellenwert verstärkt.
- Attack/Release: Die Attack- und Release-Zeiten geben an, wie schnell im Kompressor das in Punkt 2 aufgeführte Verhältnis bei Überschreitung des Schwellenwertes voll angewendet wird und wie lange nach Unterschreitung es weiter anhalten soll. Je länger die zeitlichen Werte gewählt werden, desto träger und weicher reagiert der Kompressor.

Einige der Parameter und das Signalverhältnis der Dynamikkompression sind grafisch in Abbildung 2.10 zu sehen. Die Dynamikkompression wird unter anderem in [81] benutzt, um augmentierte Eingabedaten zu erstellen.

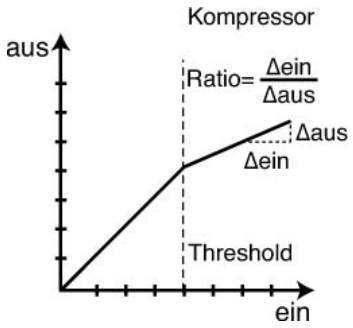


Abbildung 2.10: Graph zur Visualisierung von Signalverhältnis von Ein- und Ausgabesignal des Kompressors, sowie Beschreibungen der Parameter Schwellwert (*Threshold*) und Verhältnis (*Ratio*) [29].

Faltungshall:

Um Live-Aufnahmen in großen Konzerthallen zu simulieren kann auf Filter zurückgegriffen werden, welche mittels einer vordefinierten Impulsantwort das Audiosignal verändern. Die Impulsantwort eines Raumes ist der Raumklang nach einem kurzzeitigen und breitbandigen Schallereignis. Hat man die Impulsantwort eines Raumes gemessen, so können andere Audioausschnitte mit einer klanglichen Veränderung erzeugt werden, die dem originalen Raumklang sehr ähnlich kommen [29, S. 288]. Technisch wird die Faltung mit einer Multiplikation der Impulsantwort mit dem Eingangssignal im Frequenzraum realisiert, zu sehen in Abbildung 2.11.

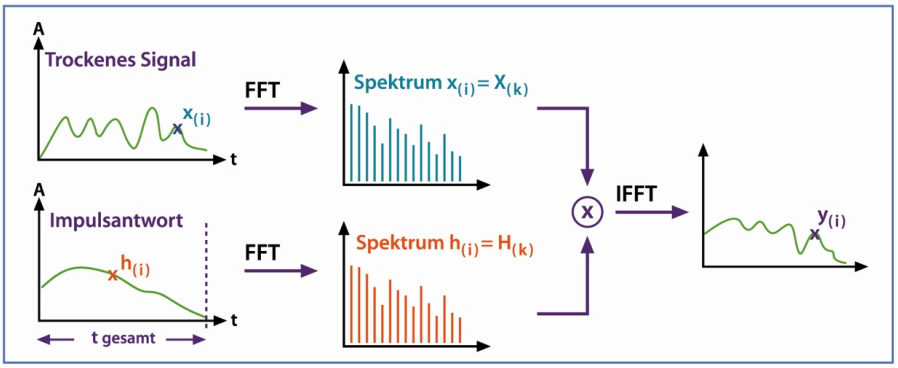


Abbildung 2.11: Berechnung des Faltungshalls mittels Transformation des Eingangssignals und der Impulsantwort in den Frequenzraum (FFT), folgende Multiplikation und abschließende Rücktransformation in den Zeitraum mittels inverser Fourier-Transformation (IFFT) [104].

3 Grundlagen der Klassifikation

Dieses Kapitel behandelt die Klassifikatormodelle, welche zum Auswerten der vorliegenden Daten verwendet werden. Das Hauptaugenmerk wird mit Abschnitt 3.2 auf künstliche neuronale Netze (NN) gelegt. Neben Grundlagen und der generellen Struktur werden auch die verschiedenen Schichtentypen vorgestellt. In den darauf folgenden Abschnitten 3.3 und 3.4 werden die weiteren Modelle Random-Forest-Klassifikator (RF) und Stützvektormaschine (SVM) thematisiert. Der thematische Abschluss dieses Kapitels folgt in Abschnitt 3.5, welcher die titelgebenden hybriden Methoden behandelt und dabei unter anderem auf die bereits vorgestellten Methoden zurückgreift. Alle vorgestellten Klassifikatormodelle werden bereits in verwandten Arbeiten zur Audio-, Musik- und Instrumentenerkennung erfolgreich eingesetzt (siehe Abschnitt 1.2).

3.1 Einführung

Das Thema dieser Arbeit ist im Kontext des maschinellen Lernens und dort im Unterbereich des überwachten Lernens angesiedelt. Im Gegensatz zu klassischen Klassifikationssystemen, welche von Hand definiert werden und somit größtenteils auf Expertenwissen basieren, wird beim maschinellen Lernen auf eine automatisierte Erkennung von Strukturen und selbst-optimierende Modelle gesetzt. Die Grundlagen für überwachtes, maschinelles Lernen sind annotierte Datensätze und das statistische Modell, nach welchem die Daten klassifiziert werden sollen. Die annotierten Datensätze werden üblicherweise in Trainings-, Validierungs- und Testdaten aufgeteilt. Auf den Trainingsdaten wird das in Abschnitt 3.1.2 beschriebene Training des Modells durchgeführt. Die Validierungsdaten werden unter anderem für die Optimierung der ebenfalls in Abschnitt 3.1.2 vorgestellten Hyperparameter der Modelle verwendet. Ein weiterer Anwendungsfall der Validierungsdaten findet sich außerdem in Abschnitt 3.2.3. Die Testdaten werden nur zur abschließenden Evaluierung verwendet und sind nicht in den Optimierungsprozess eingebunden. Bei der Aufteilung der Datensätze ist es wichtig, dass die drei Untermengen repräsentativ und gleichzei-

tig disjunkt und unabhängig voneinander sind, damit Validierung und Evaluierung auf unbekannten Daten stattfindet und somit insbesondere bei der Evaluierung die Generalisierung des Modells verifiziert werden kann.

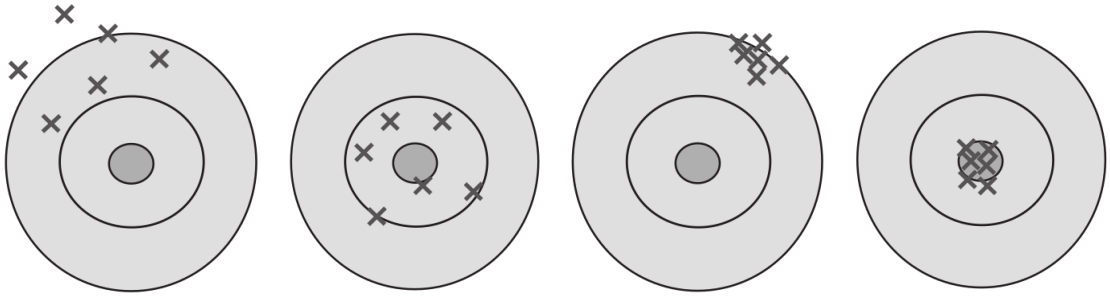
3.1.1 Muster

Als Muster werden strukturierte Daten bezeichnet, welche eine vordefinierte Anzahl von charakterisierenden Merkmalen enthalten. Eine grundlegende statistische Annahme ist, dass Muster mit ähnlichen Merkmalen auch ähnliche Klassifikationskategorien besitzen. Das passende Ähnlichkeitsmaß zu finden und somit eine Verarbeitung der konkreten Merkmale zu einer abstrakten Klassenentscheidung zu bilden ist der zentrale Bestandteil beim Bau eines Klassifikationssystems. Ein zusätzlicher Bestandteil ist die bereits in Kapitel 2 vorgestellte übliche Vorverarbeitung der Rohdaten zu Merkmalen.

3.1.2 Training

Bevor ein Klassifikatormodell beim überwachten Lernen Klassenentscheidungen treffen kann, ist bei den meisten Klassifikatoren zunächst ein sogenanntes Training erforderlich. Beim Training werden dem Modell beispielhafte Muster mit den zugehörigen korrekten Annotierungen übergeben. Die Parameter des Modells werden nun auf Basis der Ein- und Ausgaben der sogenannten Trainingsdaten angepasst. Das jeweilige Optimierungskriterium hängt dabei vom Modell ab und kann durch zusätzliche Hyperparameter eingestellt werden. Unter anderem bei der Wahl der Hyperparameter gilt es auch stets das „Verzerrung-Varianz-Dilemma“ zu beachten. Die Verzerrungskomponente gibt dabei an wie akkurat ein Modell im Mittel auf verschiedenen Trainingsdaten ist. Dem gegenüber steht die Varianz, welche ein Maß ist wie sensitiv das Modell auf kleine Änderungen in den eingegebenen Daten reagiert und somit ein Hinweis auf eine Überanpassung (*Overfitting*) ist [82, S.100]. Methoden, welche einer Überanpassung der Modelle entgegenwirken werden dem Begriff „Regularisierung“ zugeordnet. Eine Illustration von Verzerrung- und Varianzfehlern ist in Abbildung 3.1 zu sehen.

Das Training eines Modells wird beendet sobald ein vorgegebenes Optimierungskriterium erreicht wird. Ist auch der gesamte Optimierungsprozess abgeschlossen, so kann das Modell abschließend auf den Testdaten evaluiert werden.



Hohe Verzerrung Hohe Varianz | Niedrige Verzerrung Hohe Varianz | Hohe Verzerrung Niedrige Varianz | Niedrige Verzerrung Niedrige Varianz

Abbildung 3.1: Genauigkeiten eines Modells mit jeweils hoher und niedriger Verzerrung und Varianz (nach [82, S. 101]).

3.2 Künstliche neuronale Netze

Als (künstliche) neuronale Netze werden Klassifikatormodelle bezeichnet, deren Grundlagen die, den Neuronen im Gehirn natürlicher Lebewesen nachempfundenen, Perzeptronen bilden. Ein Vergleich zwischen natürlichem Neuron und künstlichem Perzeptron ist in Abbildung 3.2 skizziert.

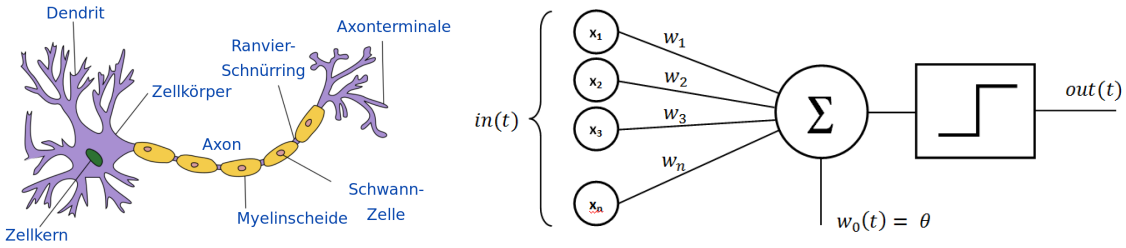


Abbildung 3.2: Der Aufbau eines natürlichen Neurons (links) [103] im Vergleich zu einem künstlichem Perzeptron (rechts) [102].

Ein Perzeptron besteht aus mehreren Eingängen (siehe x_i in rechter Abbildung 3.2), welche gewichtet aufsummiert und anschließend in einer sogenannten Aktivierungsfunktion nicht-linear transformiert werden. Einige Beispiele für Aktivierungsfunktionen sind die *Schwellwertfunktion*, *Sigmoidfunktion* und die *Rectified Linear Unit (ReLU)*, zu sehen in Abbildung 3.3. Die trainierbaren Parameter in einem NN befinden sich in der Regel in den Gewichten der Perzeptroneneingänge (siehe w_i in rechter Abbildung 3.2).

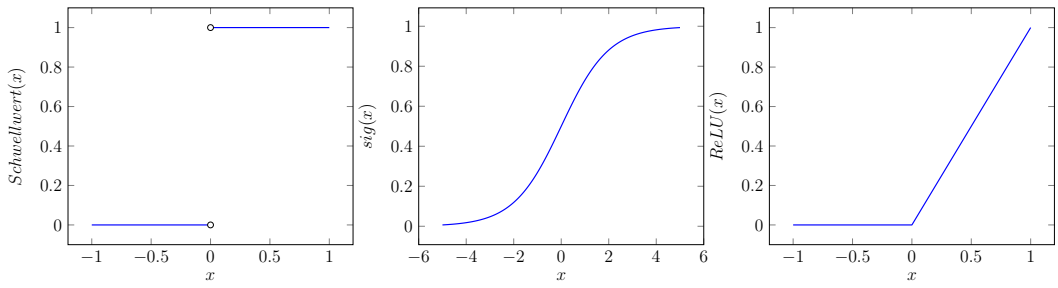


Abbildung 3.3: Schwellwert-, Sigmoid- und ReLU-Funktion

3.2.1 Aufbau

Da ein einzelnes Perzeptron bereits einfache Probleme, wie das XOR-Problem, nicht lösen kann [64], wird dieses mit weiteren Perzeptronen zu einem sogenannten künstlichen neuronalen Netz verbunden. Der in dieser Arbeit behandelte und allgemein gebräuchliste Typ von künstlichen neuronalen Netzen ist das *Feed-Forward*-Netz, in welchem Perzeptronen streng hintereinander und für effizientere Berechnungen in Schichten angeordnet werden. Eine Ausnahme von der streng von Ein- zu Ausgabe verlaufenden Propagation bilden die rekurrenten Schichten, welche zusammen mit den restlichen verwendeten Schichttypen im folgenden Abschnitt 3.2.2 näher beschrieben werden. Des Weiteren gibt [83] einen weiteren Überblick über das Themengebiet von tiefen neuronalen Netzen.

3.2.2 Schichttypen

Im Folgenden werden verschiedene Schichttypen eines NN eingeführt.

Voll-vernetzte Schichten

Die Basis eines NN bilden die voll-vernetzten Schichten. In dieser Art von Schicht ist jedes Perzeptron mit jedem Perzeptron der vorherigen Schicht mit einer gewichteten Kante verbunden und enthält zusätzlich einen Versatz-Wert (*Bias*), welcher eine eingabeunabhängige Translation der gewichteten Summe ermöglicht (siehe w_0 in rechter Abbildung 3.2). Bezeichnet L_i die Anzahl der Perzeptronen der Schicht i so berechnet sich die Anzahl der Trainingsparameter A einer voll-vernetzten Schicht i zu:

$$A_i = (L_{i-1} + 1) \cdot L_i \tag{3.1}$$

Die Realisierung dieser Schicht kann als Matrixmultiplikation der Eingänge mit den zugehörigen Gewichten realisiert werden. Somit entspricht die Ausgabe y eines voll-

vernetzten Perzeptrons mit Aktivierungsfunktion f, Gewichten W, Eingaben x und Bias W_0:

$$\begin{aligned} y &= f\left(\left(\sum_{i=1}^{n} W_i x_i\right) + W_0\right) \\ &= f\left(W^T[x, 1]\right) \end{aligned} \tag{3.2}$$

Faltungsschichten

Faltungsschichten ähneln neuronalen Verbindungen im menschlichen visuellen Cortex [99]. Neben der daraus folgenden Verwendung zur Bilderkennung, ist auch die Anwendung auf anderen Daten wie Spektrogrammen plausibel [74].
Im Unterschied zu voll-vernetzten Schichten transformieren Faltungsschichten die vorherige Schicht mithilfe eines sogenannten Filterkernels, welcher als Fenster über die Eingabedaten bewegt wird (siehe Abbildung 3.4). Die Transformation erfolgt mit einer Matrixmultiplikation des Kernels an jeder durch Hyperparameter definierten Stelle gleich. Somit ergibt sich im Vergleich zu den voll-vernetzten Schichten durch die Parameterteilung und somit einhergehende Translationsinvarianz eine Reduzierung der Parameter. Die Kernelmatrix entspricht dabei den optimierbaren Gewichtsparametern der Schicht. Die Hyperparameter einer Faltungsschicht sind im Folgenden aufgelistet und erläutert.

Kernelgröße
Die Kernelgröße K gibt die Größe des Fensters an, welches über die Daten geschoben wird. Je nach Struktur der vorliegenden Daten kann auch die Dimension des Kernels höher gewählt werden. Der gebräuchlichste Typ bei der Bildklassifizierung sind zweidimensionale Kernel. So enthält ein zweidimensionaler Kernel von Größe $K = 3 \times 3$ insgesamt 9 Gewichte pro Eingabekanal.

Schrittgröße
Die Schrittgröße S gibt an, in welchen Abständen der Kernel über die Daten bewegt wird. Eine Schrittgröße von 1 schiebt den Kernel über jede mögliche Stelle, während größere Werte den Versatz des Fensters einige Datenpunkte überspringen lässt. Je größer die Schrittgröße gewählt wird, desto kleiner wird die resultierende, transformierte Ausgabe. Bei mehrdimensionalen Filtern sind auch dimensionsunterschiedliche Schrittgrößen möglich.

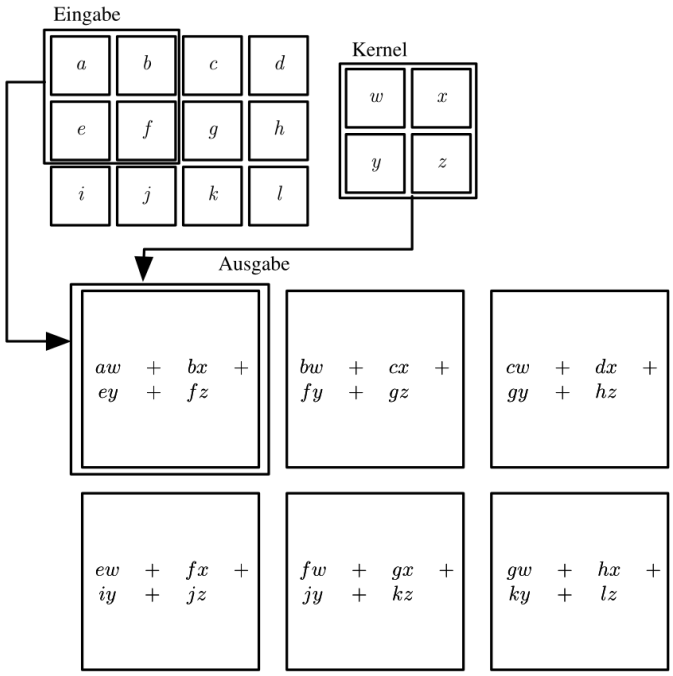


Abbildung 3.4: Ein 2x2-Filterkernel wird über die Eingabe bewegt und erzeugt dabei eine transformierte Ausgabe [31, Abbildung 9.1].

Zeropadding

Das Außmaß des *Zeropaddings* P gibt an, um wie viele Datenpunkte die Ränder der Eingabe mit Nullen ergänzt werden sollen. Diese Art der künstlichen Auffüllung kann von Nutzen sein, um die Dimensionalität der Eingabe bei der Transformation zu erhalten.

Kernelanzahl

Ein Kernel wird durch die bereits aufgeführten Hyperparameter definiert. Die Anzahl der Kernel wird durch den Hyperparameter L gewählt. Ist die Ausgabe eines Kernels von Dimensionalität $W \times H$ so ergibt sich mit L Kerneln die Ausgabedimension der Schicht als: $W \times H \times L$.

Parameter- und Ausgabegröße bei zweidimensionalen Daten

Werden die in der Bilderkennung üblichen zweidimensionalen Kernel verwendet und ist die Eingabe mit einer Größe von W_{in} Breite und H_{in} Höhe gegeben, so ergeben sich die folgenden Ausgabegrößen einer Faltungsschicht.

$$\begin{aligned} W_{out} &= \frac{1}{S}(W_{in} - K + 2P) + 1 \\ H_{out} &= \frac{1}{S}(H_{in} - K + 2P) + 1 \\ D_{out} &= L \end{aligned} \tag{3.3}$$

Die Anzahl der Parameter einer zweidimensionalen Faltungsschicht mit L Kerneln, Kernelgröße $K \times K$ und Eingabekanaltiefe von D_{in} entspricht:

$$(K \cdot K \cdot D_{in} + 1) \cdot L \tag{3.4}$$

Rekurrente Schichten

Der Einsatz von rekurrenten Schichten kann dann erfolgen, wenn ein zeitlicher Verlauf der Daten erfasst werden soll. Während die Propagation in voll-vernetzten Schichten und Faltungsschichten streng von Ein- nach Ausgabe folgt, so enthalten rekurrente Schichten auch Gewichte, welche Informationen aus zeitlich vorangegangen Iterationen miteinbeziehen. Die Rückkopplung (*Feedback*) kann dabei direkt oder indirekt erfolgen, in dem der eigene Ausgang oder der Ausgang eines nachfolgenden Perzeptrons als zusätzlicher Eingang verwendet wird.
Eine Erweiterung der rekurrenten Schichten ist die Einführung von *Long short-term Memory* (LSTM) Schichten [39]. Diese werden unter anderem in Spracherkennungssystemen verwendet [36][3] und ergänzen die direkte rekurrente Verbindung durch gewichtete Gatterverbindungen, welche den internen Informationsfluss regeln und als zusätzliche Parameter optimiert werden. Ein LSTM-Modul enthält neben der Informationseinheit c_t jeweils drei Gatterarten. Das Eingabegatter regelt den Informationseinfluss der Eingabe auf die Informationseinheit c_t. Das Ausgabegatter regelt den Informationsfluss aus c_t heraus zur Ausgabe. Das letzte Gatter dient zur Einstellung der Verflüchtigung von Informationen in c_t und kann biologisch als „Vergessen“ von Informationen gesehen werden. Die Gatter sollen eine bessere Anpassung auf Informationen ermöglichen, welche kürzere oder längere zeitliche Relationen besitzen. Da der einfache Fehlerrückführungsalgorithmus nicht auf rekurrente Schichten angewendet werden kann, werden rekurrente Schichten mit zeitabhängiger Fehlerrückführung (*Backpropagation through time*) trainiert [100]. Der Einsatz von rekurrenten Schichten ist in dieser Masterarbeit nicht implementiert, eine mögliche Anwendung in fortführenden Arbeiten wird aber im Ausblick in Abschnitt 5.2.2 diskutiert.

Pooling

Pooling-Schichten fassen mehrere Datenpunkte lokal zusammen. Hyperparameter der zusammenfassenden Schichten sind ähnlich wie bei Faltungsschichten die Fenstergröße, die Schrittgröße und das *Zeropadding*. Zusätzlich muss die Art der Zusammenfassung definiert werden, beispielsweise durch Maximierung oder Durchschnittsbildung. In tiefen neuronalen Netzen werden diese Schichten benutzt, um

eine Datenreduktion durchzuführen und gleichzeitig eine mit fortschreitender Tiefe abstrahierende Sicht auf die Merkmale zwischen Faltungsschichten zu erhalten, da sich die rezeptiven Felder, also das Betrachtungsfenster auf das Gesamtbild, mit der Fenstergröße der Poolingschicht multipliziert. Biologisch betrachtet sind maximierende Poolingschichten stark von der lateralen Hemmung inspiriert, welche ebenfalls im visuellen System des Menschen zu finden ist [27].

Dropout

Dropout-Schichten sind eine Form der Regularisierung [87]. Dabei werden, wie in Abbildung 3.5 rechts zu sehen, Ausfälle von Perzeptronen simuliert, indem zufällige Indizes der Eingabematrix auf 0 gesetzt und der Rest unverändert an die folgende Schicht weitergegeben wird. Ein Hyperparameter definiert wie viele Datenpunkte entfernt werden sollen. Den regularisierenden Effekt erreichen Dropout-Schichten, da die Ausfälle einem Training mehrerer NN mit geteilten Gewichten und somit einer Modellmittelung, vergleichbar mit dem in Abschnitt 3.3 vorgestellten Random-Forest-Klassifikator, entspricht. Insbesondere bei der Merkmalsberechnung von Faltungsschichten werden so Koadaptionen verringert und Merkmale bevorzugt, welche generalisierende Informationen enthalten [38].

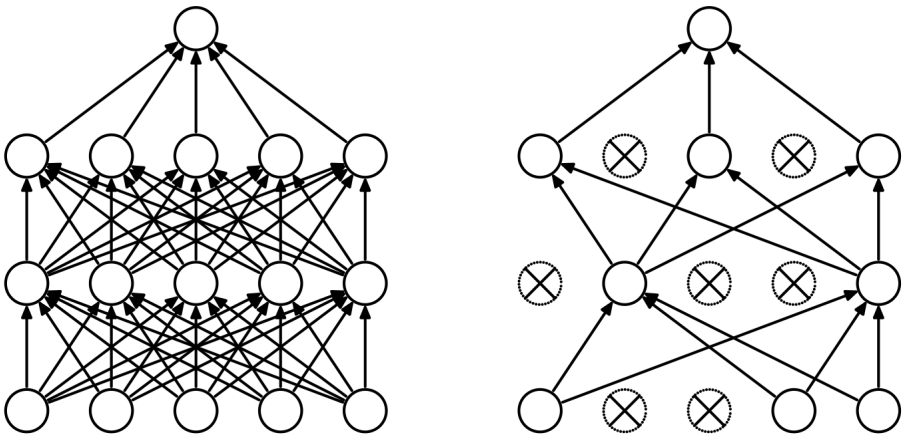


Abbildung 3.5: Schichten eines voll-vernetzten NN ohne (links) und mit (rechts) Dropout-Regularisierung [87, Abbildung 1].

Batch-Normalisierung

Bei der *Batch*-Normalisierung wird die Eingabe hinsichtlich Mittel und Varianz normalisiert. Die Grundidee hinter dieser Methode ist es, die übliche Normalisierung der Trainingsdaten (siehe Abschnitt 2.3) auch auf die internen Repräsentationen

des NN zu übertragen. Durch die angewandte Normalisierung wird unter anderem das Training beschleunigt und stabilisiert [40]. Des Weiteren hat die Normalisierung einen reduzierenden Effekt auf die Kovariaten-Verschiebung. Diese tritt auf, wenn sich bei einer gelernten Abbildung $X \to Y$ die Verteilung der unabhängigen Variable X ändert und so eine Anpassung der Abbildung vonnöten ist. Insbesondere bei tiefen neuronalen Netzen sind die hinteren Schichten stark von den vorherigen Schichten abhängig und Gewichtsänderungen in den ersten Schichten können zu stark variierenden Werten in den hinteren Schichten führen, was unter anderem zum Problem des explodierenden Gradienten führen kann [31, S. 356]. Normalisierung wirkt diesen Auswirkungen entgegen, sodass einzelne Schichten unabhängiger von davor liegenden Schichten optimieren können [40].
Da, wie in Abschnitt 3.2.3 genauer thematisiert, das Training von NN üblicherweise in Sätzen (*Batches*) erfolgt, wird auch die Normalisierung nur für die aktuell durch das Netz propagierte Batch berechnet.
Ist die Batchgröße auf n festgelegt und bezeichnet $B = \{x_1, \ldots, x_n\}$ die aktuelle Batch, so ergeben sich das Mittel μ und die Varianz δ^2 der Batch B zu:

$$\begin{aligned} \mu_B &= \frac{1}{n} \sum_{i=1}^{n} x_i \\ \delta_B^2 &= \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu_B)^2 \end{aligned} \tag{3.5}$$

Liegen mehrdimensionale Eingangsdaten mit Dimension d vor, so wird jede Dimension von $x = (x^{(1)}, \ldots, x^{(d)})$ seperat normalisiert und zusammen mit ϵ als numerischen Stabilisator ergibt sich darausfolgend der Normalisierungsterm:

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{\delta_B^{(k)^2} + \epsilon}} \tag{3.6}$$

Um die Repräsentationskraft des Netzwerks zu erhalten, wird zusätzlich eine Transformation in Form von Skalierung γ und Verschiebung β durchgeführt:

$$y_i^{(k)} = \gamma^{(k)} \hat{x}_i^{(k)} + \beta^{(k)} \tag{3.7}$$

Die optimalen Transformationsparameter $\gamma^{(k)}$ und $\beta^{(k)}$ werden während des Trainings für jede Dimension als zusätzliche Parameter gelernt.

3.2.3 Optimierung

Künstliche neuronale Netze werden mit dem Fehlerrückführungsalgorithmus (*Backpropagation*) optimiert. Dabei werden zunächst alle Gewichte zufällig initialisiert und anschließend die Trainingsdaten in das Netz gegeben und durch die Schichten bis zur Ausgabeschicht berechnet.

Fehlermaße

In der Ausgabeschicht erfolgt mithilfe eines geeigneten Fehlermaßes (*Loss*) die Berechnung der Kostenfunktion als Differenz zwischen korrekter und tatsächlicher Ausgabe. Beispiele für Fehlermaße sind der mittlere quadratische Fehler (*mean-squared-error*), die kategorische Kreuzentropie, sowie die in dieser Arbeit für das Multiklassenproblem verwendete binäre Kreuzentropie [6, S. 17][31, K. 6.2.1].

Gradientenabstieg

Der Trainingsprozess wird nach Berechnung der Kostenfunktion rückwärts durch das Netz durchgeführt, indem jeweils zu jedem Perzeptron der aktuellen Schicht die partiellen Ableitungen aus den lokalen Fehlern der nachfolgenden Schicht mit den entsprechenden Gewichten gebildet werden. Durch rekursive Anwendung der Kettenregel kann so zu jedem Gewicht im Netz der dort anliegende Fehler berechnet und das Gewicht in die negative Richtung des Fehlers verändert werden. Der Faktor, welcher angibt wie stark Gewichte bei jeder Iteration des Gradientenabstiegs in Richtung des negativen Fehlers verändert werden sollen, wird auch als Lernrate bezeichnet.

Problem des verschwindenden Gradienten

Die Benutzung der ReLU-Aktivierungsfunktion ist bei der Fehlerrückführung in tiefen neuronalen Netzen von wichtiger Bedeutung, da sie im Gegensatz zu beispielsweise der sigmoid-Funktion im geringeren Umfang zum Problem des verschwindenden Gradienten führt [31, S. 192]. Das Problem des verschwindenden Gradienten entsteht unter anderem durch die rekursive Anwendung der Kettenregel auf die jeweiligen Aktivierungsfunktionen. Kann beispielsweise die Ableitung der Aktivierungsfunktion $a(x)$ mit $m > a'(x)$ nach oben abgeschätzt werden, so ergibt sich für $m < 1$ das Problem, dass sich der lokal anliegende Fehler mit zunehmender Anzahl der iterierten Schichten sukzessive mindestens um den Faktor m verringert und somit gerade das Training der ersten Schichten nicht effizient erfolgen kann [31, K. 8.2.5].

Optimierer

Da die Berechnung der Netze für den kompletten Datensatz wenig praktikabel ist, wird das Training in sogenannten Sätzen (*Batches*) durchgeführt. Es gibt verschiedene Optimierer, welche den Gradientenabstieg mit unterschiedlichen Methoden durchführen. Die Basis bildet der einfache stochastische Gradientenabstieg (SGD), welcher die Richtung des Gradientenabstiegs anhand der einzelnen Batches berechnet. Erweitert wird der stochastische Gradientenabstieg durch Optimierer, welche unter anderem, wie in Abbildung 3.6 zu sehen, ein sogenanntes Momentum in die Berechnung mit einbeziehen, um eine bessere Konvergenz des Trainings zu erhalten. Zu den häufig verwendeten Optimierern gehören: SGD, *Root Mean Square Propagation Algorithm* (RMSProp) und der in dieser Arbeit verwendete „adaptive Momente“ (ADAM) Optimierer [31, K. 8.5.4][44]. RMSProp und ADAM implementieren unter anderem zusätzlich eine adaptive Änderung der Lernrate.

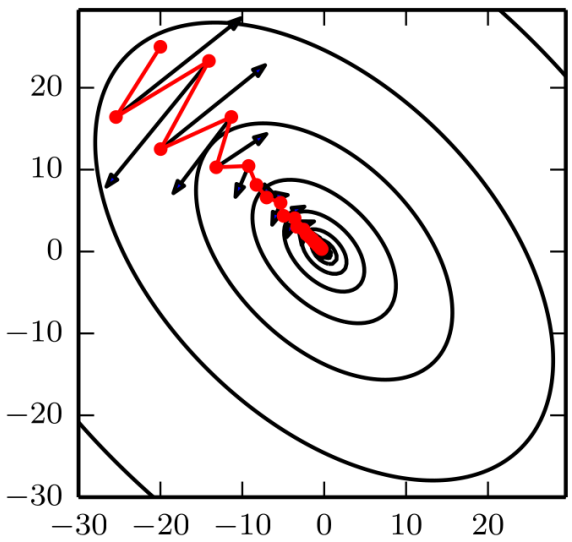


Abbildung 3.6: Gradientenabstieg mit eigentlicher Richtung des Gradienten (schwarz) und Schrittrichtung mit Momentum (rot) [31, Abbildung 8.5].

Regularisierung

Da tiefe neuronale Netze aufgrund ihrer hohen Parameteranzahl leicht zur Überanpassung tendieren [31, K. 7.8], gibt es zahlreiche Regularisierungstechniken, welche dem entgegenwirken. Eine Regularisierungsmethode ist die in Abschnitt 3.2.2 vorgestellte *Dropout*-Schicht. Weiterhin wirken auch die in Abschnitt 3.2.2 vorgestellten Faltungsschichten durch eine Reduzierung der Parameteranzahl im Vergleich zu vollvernetzten Schichten einer Überanpassung entgegen.

Eine weitere in dieser Arbeit eingesetzte Methode der Regularisierung ist das „frühe Abbrechen“ (*Early-Stopping*) [75]. Nach jedem vollständigen Durchlauf (Epoche) der in Batches eingeteilten Trainingsdaten, wird das Modell auf den, in Abschnitt 3.1 thematisierten, Validierungsdaten evaluiert. Wird auf den Validierungsdaten kein vorher definierter signifikanter Fortschritt mehr erzielt (siehe Abbildung 3.7), so wird das Training frühzeitig beendet.

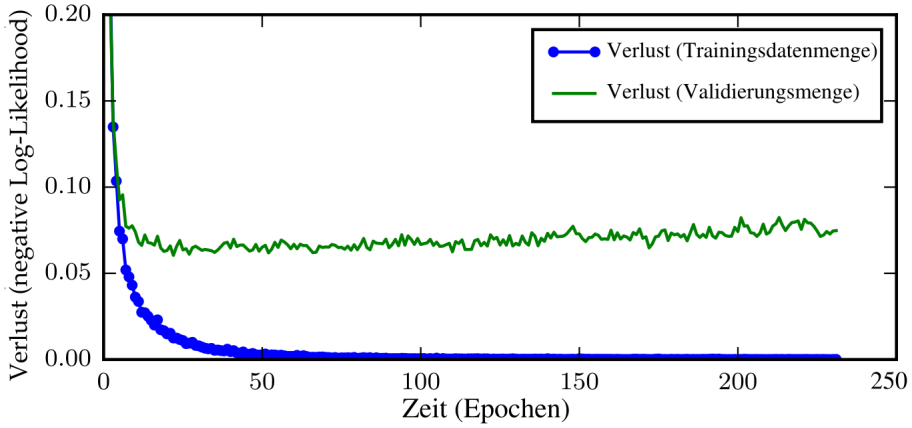


Abbildung 3.7: Trainingsfehler (blau) und Validierungsfehler (grün) bei der Optimierung eines NN [31, Abbildung 7.3].

3.3 Random-Forest-Klassifikatoren

In diesem Abschnitt wird der Random-Forest-Klassifikator (RF) vorgestellt. Es wird zunächst in die Entscheidungsbäume eingeführt, welche die Klassifikatorbasis des RF bilden. Folgend wird das zugrundeliegende Prinzip des *Baggings* näher beschrieben. Anschließend werden Aufbau, Eigenschaften und Parameter des Random-Forest-Klassifikators näher dargelegt.

3.3.1 Entscheidungsbäume

Die Basis des RF bilden die sogenannten Entscheidungsbäume. Ein Entscheidungsbaum wird konstruiert, indem die annotierten Trainingsdaten iterativ anhand ihrer Merkmale aufgeteilt werden, sodass jedes Blatt des Entscheidungsbaums entweder eine eindeutige Klassenentscheidung oder die jeweiligen Klassenentscheidungskonfidenzen enthält. Ein Vorteil der Entscheidungsbaumstruktur ist, dass sie auf kleineren Datensätzen interpretierbar ist und somit (Fehl-)Entscheidungen des Modells von einem Menschen im Gegensatz zu NN besser nachvollzogen werden können.

Wie in Abbildung 3.8 skizziert, gibt es in einem Entscheidungsbaum Knoten, Kanten und Blätter. Jeder Knoten (orange im Beispiel) entspricht einem einzelnen Merkmal in den Daten und jedes Blatt (grün/rot) einer Klassenentscheidung. Eine Kante (blau) repräsentiert einen Test auf dem im Elternknoten definierten Merkmal. Abhängig von der vorliegenden Form der Daten kann ein Test entweder (wie im Beispiel) kategorisch oder (wie im Rahmen dieser Masterarbeit) numerisch durchgeführt werden. Zur Klassifizierung von numerischen Daten, werden Methoden wie *Classification and Regression Trees* (CART) [14] angewendet, um Aufteilungen mithilfe von Intervallabgrenzungen zu realisieren. Im Folgenden wird auf die Kriterien zur Merkmals- und Aufteilungswahl näher eingegangen.

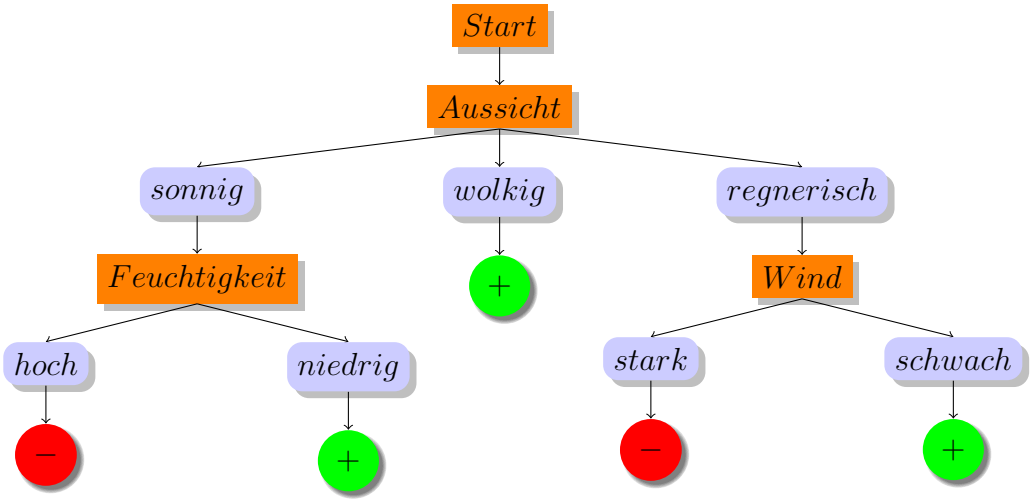


Abbildung 3.8: Beispielhafter Entscheidungsbaum zur wetterabhängigen Planung von Freizeitaktivitäten.

Aufteilungskriterien

Neben der Entscheidung ob Bäume binär oder mit mehreren Zweigen aufgebaut werden, ist die Wahl des Teilungskriteriums von zentralem Bestandteil. Ist T eine Teilmenge der Trainingsdaten, so lässt sich p_j als Angabe der relativen Häufigkeit der Klasse C_j in T wie folgt definieren:

$$p_j = \frac{|C_j \cap T|}{|T|} \tag{3.8}$$

Darauf aufbauend gibt es zur Messung der Qualität einer Aufteilung unter anderem den Gini-Koeffizienten und den *Information Gain*. Der Gini-Koeffizient wird zum Beispiel im bereits aufgeführten CART-Algorithmus [14] und somit auch in der in

dieser Masterarbeit verwendeten Bibliothek *scikit-learn* [73] angewendet. Die Qualität der Klassenverteilung in einer Teilmenge T ist gemäß Gini-Koeffizient definiert als:

$$Gini(T) = 1 - \sum_{j=1}^{|C|} p_j^2 \tag{3.9}$$

Das Maximum wird mit $1 - \frac{1}{|C|}$ bei einer Gleichverteilung der Klassen und das Minimum 0 bei einer Verteilung zu einer einzelnen Klasse erreicht.
Im Gegensatz zum Gini-Koeffizienten berechnet der *Information Gain* den Informationsgewinn mithilfe der Entropie. Dieser wurde unter anderem im Entscheidungsbaumalgorithmus *ID3* verwendet [77]. Die Entropie des *Information Gain* ist wie folgt definiert:

$$entropie(T) = -\sum_{j=1}^{|C|} p_j \cdot log_2 p_j \tag{3.10}$$

Das Maximum wird mit $log_2|C|$ bei einer Gleichverteilung und mit 0 ein Minimum bei einer kompletten Zuordnung zu einer Klasse erreicht.
Werden die zu teilenden Daten T anhand eines gegebenen Merkmals A kategorisch oder numerisch in m Teilmengen $\{T_1, T_2, \ldots, T_m\}$ aufgeteilt, so lässt sich der Informationsgewinn der Aufteilungen mit einem der beiden Koeffizienten als $koef$ wie folgt berechnen:

$$gewinn(T, A) = koef(T) - \sum_{i=1}^{m} \frac{|T_i|}{m} \cdot koef(T_i) \tag{3.11}$$

Mithilfe dieser Entscheidungskriterien werden die optimalen Merkmale und Aufteilungen iterativ gewählt und die Bäume absteigend aufgebaut. Um eine Überanpassung zu verhindern kann außerdem ein vorläufiges Abbruchkriterium, beispielsweise die Vorgabe einer maximalen Tiefe, oder ein nachträgliches Beschneiden (*Pruning*) der Bäume zu einer besseren Generalisierung führen [63].

3.3.2 Bagging

Das zugrundeliegende Prinzip des RF ist die *Bagging*-Methode (*Bootstrap aggregating*). Das Ziel beim Erstellen eines Klassifikators nach der *Bagging*-Methode ist es aus mehreren Klassifikatoren mit jeweils hoher Varianz und niedriger Verzerrung (siehe Abschnitt 3.1.2) einen neuen Klassifikator mit verringerter Varianz und gleichzeitig niedriger Verzerrung zu erhalten [12]. *Bagging* ist somit eine Form der

Modellmittelung und soll die Stabilität eines Modells verbessern. Zur Mittelung werden zunächst, mithilfe der originalen Trainingsdaten T, m neue Trainingsmengen T_i mit jeweils n Elementen generiert. Die Auswahl der einzelnen Trainingsdaten für T_i erfolgt auf Basis des *Bootstrap*-Verfahrens und folglich werden n Elemente aus T zufällig mit Zurücklegen gezogen [22]. Anschließend werden m Basis-Klassifikatoren auf den jeweiligen Untermengen trainiert. Die Klassenentscheidungen der Klassifikatoren werden ausgewertet und am Ende dem Problem entsprechend aggregiert.

3.3.3 Aufbau und Eigenschaften

Der RF wendet das Prinzip des *Baggings* mit Entscheidungsbäumen als Klassifikatorbasis an. Jeder Entscheidungsbaum wird somit nur auf einem Teil der Trainingsdaten trainiert. Weiterführend werden auch die Merkmale, welche zur Konstruktion eines einzelnen Entscheidungsbaums berücksichtigt werden, zufällig gewählt. Die Anzahl der jeweils zufällig zu wählenden Merkmale wird dabei anhand eines Hyperparameters oder einer Heuristik, wie beispielsweise der Quadratwurzel der Merkmalsgröße, festgelegt. Zu den weiteren Hyperparametern des Random-Forest-Klassifikators gehören unter anderem die Anzahl der Bäume, das Merkmalteilungskriterium (siehe Abschnitt 3.3.1), die maximale Baumtiefe, sowie analog zur Merkmalsanzahl, die Anzahl an Trainingsdaten pro Entscheidungsbaum.
Aufgrund des Gesetzes der großen Zahlen, neigt der RF bei einer größeren Baumanzahl, im Gegensatz zu beispielsweise NN, nicht zur Überanpassung [13, S. 25]. Somit wird die Anzahl der Bäume lediglich durch Rechenkapazitäten nach oben begrenzt.

3.4 Stützvektormaschinen

3.4.1 Einführung

Die Stützvektormaschine (*Support Vector Machine*) klassifiziert Beispiele, indem sie die vorliegenden annotierten Trainingsdaten durch eine Hyperebene und den damit aufgespannten Bereich zwischen zwei verschiedenen Klassenbeispielen, maximal voneinander trennt [19]. Die Hyperebene wird dabei lediglich von den Trainingsbeispielen definiert, welcher der Hyperebene im Merkmalsraum am nächsten liegen. Diese elementaren Beispiele werden auch Stützvektoren (*Support Vectors*) genannt und sind ausschlaggebend für den Namen des Klassifikators. Eine beispielhafte Trennung zweier Klassen mit einer Stützvektormaschine ist in Abbildung 3.9 zu sehen.

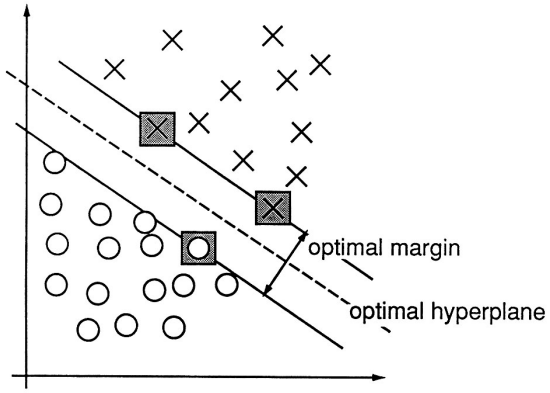


Abbildung 3.9: Eine Trennung im zweidimensionalen Raum mittels Hyperebene (gestrichelte Linie), Spanne (gezogene Linien) und Stützvektoren (graue Rechtecke) [19, S. 275].

3.4.2 Aufbau

Die SVM klassifiziert im Intervall [-1,1] und verwendet somit auch Trainingsdaten, welche je nach Klassenzugehörigkeit y_i des Beispiels i mit -1 oder 1 annotiert sind. Zusammen mit den Merkmalen x_i ergibt sich dann die Trainingsdatenmenge mit m Elementen zu:

$$\{(x_i, y_i)|i = 1, \ldots, m; y_i \in \{-1, 1\}\} \tag{3.12}$$

Die Ausrichtung der Hyperebene ist definiert durch den Normalenvektor w, der senkrecht zur aufgespannten Ebene steht. Zusätzlich wird der Versatz der Ebene durch den Bias b definiert, welcher aus dem negativen Abstand vom Schnitt der Hyperebene mit dem Koordinatensystem zum Ursprung entsteht. Für alle Punkte x, welche auf der Hyperebene liegen, gilt dann:

$$\langle w, x\rangle + b = 0 \tag{3.13}$$

Für alle anderen Punkte ergibt die Gleichung ein positives oder negatives Vorzeichen. Somit berechnet sich die Klassenzugehörigkeit $y_i \in \{-1, 1\}$ eines beliebigen Beispiels mit Merkmalen x_i zu:

$$y_i = sgn(\langle w, x_i\rangle + b) \tag{3.14}$$

Die Optimierung der Parameter w und b erfolgt, sodass der minimale Abstand der Trainingsdaten zur Hyperebene maximiert wird und somit eine möglichst breite Trennung der Merkmalsräume erreicht wird (*Large-Margin-Classification*). Es wird dabei die Hyperebene gewählt, welche die minimale, quadratische Norm $||w||_2^2$ aufweist. Somit ergibt sich das folgende Optimierungsproblem, welches mithilfe der

Lagrange-Optimierung gelöst werden kann:

Minimiere $\frac{1}{2}||w||_2^2$ unter Einhaltung der durch die Trainingsbeispiele definierten Nebenbedingungen:

$$y_i(\langle w, x_i\rangle + b) \geq 1 \quad \forall 1 \leq i \leq m \tag{3.15}$$

Da die Trainingsbeispiele lediglich mit einer trennenden Hyperebene klassifiziert werden, kann die Stützvektormaschine so nur das linear-lösbare Zweiklassenproblem lernen. Im allgemeinen Fall sind jedoch komplexe Probleme aufgrund von Ausreißern und generellen Überlappungen der Klassen selten linear lösbar. Deshalb existieren einige Erweiterungen der Stützvektormaschine, welche im Folgenden vorgestellt werden.

3.4.3 Schlupfvariablen

Schlupfvariablen erlauben geringe Verletzungen der Nebenbedingungen. Es wird somit für jede Nebenbedingung eine Schlupfvariable ξ_i eingeführt, welche den Faktor der Verletzung angibt. Das Optimierungsproblem wird mit den Schlupfvariablen erweitert und zusätzlich ein Hyperparameter C eingeführt, welcher als Faktor angibt, wie sehr Nebenbedingungsverletzungen bei der Optimierung ins Gewicht fallen. Mit $C = \infty$ erhält man wieder das Standardverhalten der SVM ohne Schlupfvariablen. Das Optimierungsproblem mit Schlupfvariablen ergibt sich zu:

Minimiere:

$$\frac{1}{2}||w||_2^2 + C\sum_{i=1}^{m}\xi_i \tag{3.16}$$

unter Einhaltung der durch die Trainingsbeispiele und ξ_i definierten Nebenbedingungen:

$$y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i \quad \forall 1 \leq i \leq m \tag{3.17}$$

3.4.4 Kernelfunktionen

Mithilfe von Schlupfvariablen lassen sich auch nicht linear-separierbare Daten voneinander trennen. Jedoch existieren im allgemeinen Fall Probleme, welche nicht optimal linear lösbar sind. Deshalb werden sogenannte Kernelfunktionen eingesetzt, um den Merkmalsraum in eine höhere Dimension abzubilden, in welchem sich die Beispiele (besser) linear trennen lassen. Ein Beispiel zur linearen Trennung des XOR-Problems ist in Abbildung 3.10 zu sehen.

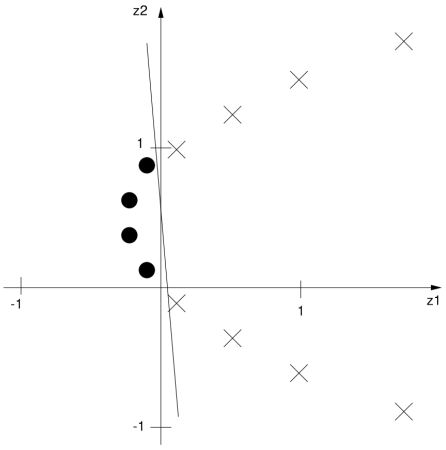


Abbildung 3.10: Das XOR-Zweiklassenproblem wird nach Transformation in 2D linear seperabel [28, S. 196].

Sei ϕ die Transformation, welche die Merkmale mit Dimension d_1 in einen Raum mit Dimension d_2 und $d_1 < d_2$ abbildet:

$$\phi : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}, x \mapsto \phi(x) \tag{3.18}$$

Da für das Optimierungsproblem lediglich Skalarprodukte berechnet werden, kann statt einer höher-dimensionalen Transformation und expliziter Skalarproduktberechnung im höher-dimensionalen Raum zweier Beispiele x_i und x_j: $\langle \phi(x_i), \phi(x_j) \rangle$ implizit auch direkt berechnet werden, in dem eine sogenannte Kernelfunktion k verwendet wird, welche wie folgt definiert ist:

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \tag{3.19}$$

Beispiele für Kernelfunktionen sind polynomielle Kernel von Grad n:

$$k(x_i, x_j) = (x_i^T x_j + c)^n \tag{3.20}$$

oder die auch in dieser Arbeit verwendete radiale Basisfunktion mit zusätzlichem Parameter δ:

$$k(x_i, x_j) = exp\left(-\frac{||x_i - x_j||^2}{2\delta^2}\right) \tag{3.21}$$

3.4.5 Multiklassenproblem

Da im Laufe dieser Arbeit in einem Audiosegment mehr als zwei Instrumente klassifiziert werden, wird die Limitierung der SVM auf Zweiklassenprobleme mittels

Problemtransformationen aufgehoben. Die Transformation erzeugt aus dem Multiklassenproblem mehrere binäre Entscheidungsprobleme mithilfe der *One-vs.-rest*-Strategie. Es werden so für jede Klasse k die Trainingsdaten einzeln transformiert und nur positiv annotiert, wenn das zugehörige Beispiel zur betrachteten Klasse gehört. Es müssen somit insgesamt k einzelne Klassifikatoren trainiert werden. Neben der in dieser Arbeit verwendeten *One-vs.-rest*-Strategie gibt es auch noch weitere wie die *One-vs.-one*-Strategie, welche zu jeder möglichen Klassenkombination einen Klassifikator trainiert und so eine Trennung berechnet. Da die Anzahl der zu trainierenden Klassifikatoren allerdings quadratisch ($k(k-1)/2$) wächst ist dieser Ansatz sehr rechenintensiv.
Die heuristische Transformation von einem Multiklassenproblem zu einem binären Problem kann jedoch dazu führen, dass negative Klassenentscheidungen bevorzugt werden, da durch die Binarisierung die Anzahl an negativen Trainingsbeispielen in der Regel höher ausfällt als die positiven [9, S. 338].

3.5 Hybride Methoden und Transferlernen

Diese Arbeit widmet sich dem Themengebiet der Instrumentenerkennung mit hybriden Methoden. Im Folgenden wird zunächst in die Thematik der hybriden Methoden eingeführt und anschließend auch auf die Adaptierung von vorberechneten Modellen, dem sogenannten „Transferlernen“, eingegangen.

3.5.1 Hybride Methoden

Der Begriff der hybriden Methoden umfasst Herangehensweisen, welche Modellstrukturen verschiedenster Klassifikatormodelle zu einem neuen Modell kombinieren. Durch die Hybridität des neuen Klassifikators können die einzelnen Vorteile der Basis-Klassifikatoren genutzt werden, welche gegebenfalls auf einzelne Aspekte der Klassifikation spezialisiert sind.
Die Kombination der Modelle kann auf verschiedene Weisen angewendet werden. So ist eine grundlegende Neukonzeption eines Klassifikators möglich, zum Beispiel in Form des *Neural Random Forests* [8], welcher die Baumstrukturen des Random-Forest-Klassifikators (siehe Abschnitt 3.3) durch künstliche neuronale Netze ersetzt. Eine Neukonzeption eines Entscheidungsbaumes zu einem künstlichen neuronalen Netz ist in Abbildung 3.11 zu sehen. Dabei wird die Struktur und die Merkmalsraumaufteilung des Entscheidungsbaumes mithilfe der Perzeptronen eines NN rea-

lisiert. Merkmals-Perzeptronen (Kreise) werden dabei nur mit Blätter-Perzeptronen (Rechtecke) verbunden, wenn ein Pfad vom Merkmal zum jeweiligen Blatt führt.

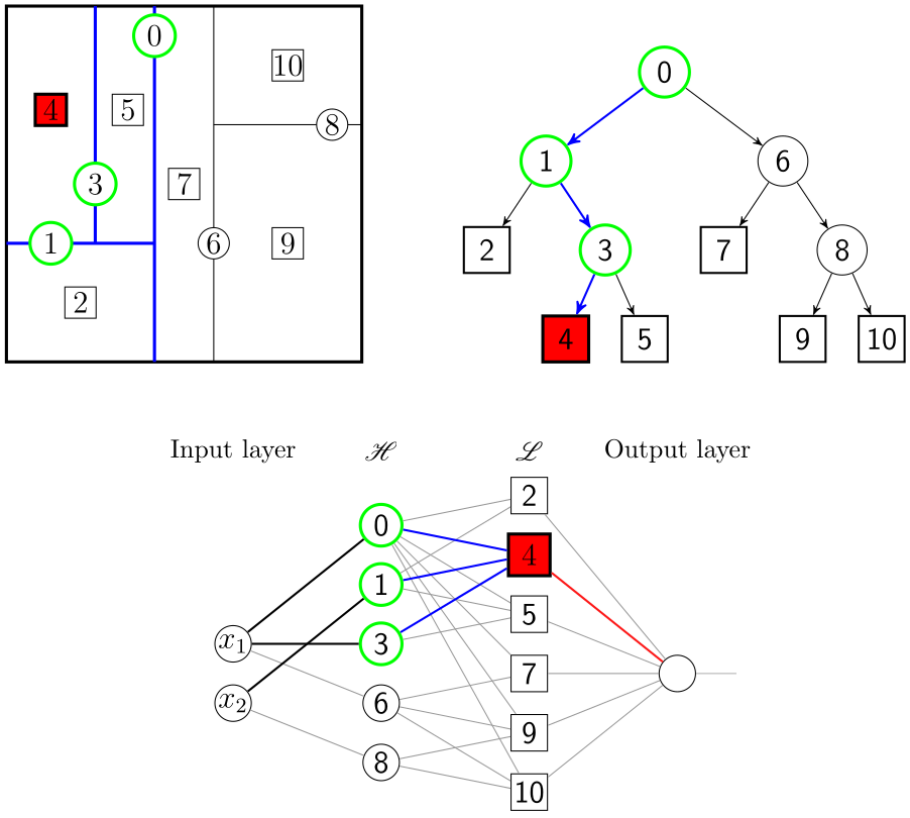


Abbildung 3.11: Ein künstliches neuronales Netz (unten), hergeleitet aus einem Entscheidungsbaum (oben-rechts), dessen Aufteilung des Merkmalsraums oben-links zu sehen ist [8].

Eine weitere Möglichkeit hybride Methoden anzuwenden, ist der Einsatz von Ensemble-Klassifikatoren, beispielsweise durch Anwendung der *Bagging*-Methode (siehe Abschnitt 3.3.2) mit verschiedenen Klassifikatormodellen. Wie in Abbildung 3.12 zu sehen, ist gerade in den letzten Jahren ein klar steigender Trend von Ausarbeitungen zu beobachten, welche hybride oder Ensemble-Methoden anwenden.

Für eine ausführlichere Auflistung von entwickelten hybriden und Emsemble-Methoden sei an dieser Stelle auf [5, K. 2.1 und 2.2] verwiesen.
Weiterführend zu den konzeptionellen und Emsemble-Methoden gibt es den in dieser Arbeit angewendeten Ansatz des Transferlernens, welcher im folgenden Abschnitt eingeführt wird.

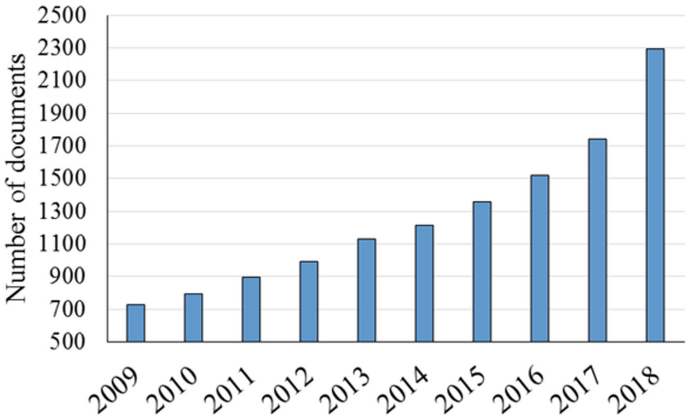


Abbildung 3.12: Der Anstieg an Ausarbeitungen zu hybriden und Ensemble-Methoden in den vergangenen Jahren [5, S. 217].

3.5.2 Transferlernen

Da die Informationsverarbeitung in NN in Schichten erfolgt, kann die Ausgabe der einzelnen Schichten als Merkmalsrepräsentation mit verschiedenen Abstraktionsgraden aufgefasst werden [90]. So neigen die ersten Schichten neuronaler Faltungsnetze häufig dazu, Parameter zu lernen, welche einer Kantenerkennung wie beispielsweise den Gabor-Filtern ähneln [107][31, S. 412]. Aufgrund der erst in den hinteren Schichten zunehmenden Spezialisierung auf ein bestimmtes Problem können so Netze, welche auf einem (meist größeren) Datensatz trainiert wurden, als Basis für ein anderes Problem verwendet werden. Insbesondere bei Datensätzen mit wenig Trainingsbeispielen kann so die Klassifikationsleistung deutlich gesteigert werden [67]. Die in dieser Arbeit angewandte Kategorie des Transferlernens wird auch als Netzwerk-basierter Ansatz bezeichnet. Eine Skizzierung des Ansatzes für die Anwendung auf verschiedenen Domänen ist in Abbildung 3.13 zu sehen.

Beim Netzwerk-basierten Transferlernen werden Schichten eines vortrainierten Netzes verwendet, um anschließend eine Feinanpassung weiterer Klassifikatoren vorzunehmen. Üblicherweise findet die weitere Feinanpassung ebenfalls mit einem NN statt, sodass mitunter auch die vollständige Architektur des vortrainierten Netzes übernommen wird und lediglich die hinteren Schichten neu trainiert werden. Im Laufe dieser Arbeit werden stattdessen vortrainierte Netze als Merkmalsberechnung für die weiteren Klassifikatoren SVM und RF verwendet, was zu einer hybriden Herangehensweise führt. Eine zusätzliche Unterscheidung zum herkömmlichen Transferlernen ist, dass keine zusätzliche Domäne eingeführt wird. Das Training des NN-Modells und die Feinanpassung der weiteren Klassifikatoren erfolgt auf derselben Datengrundlage.

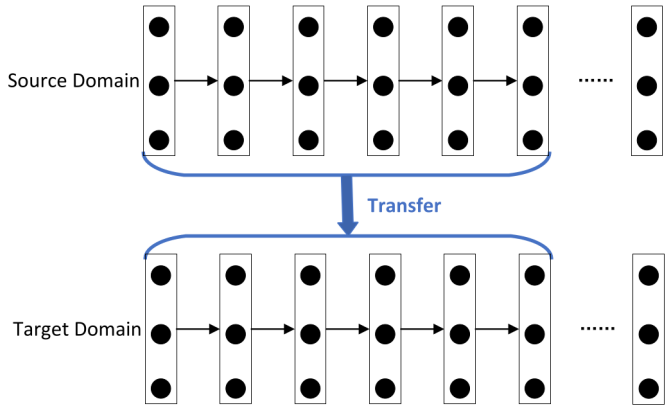


Abbildung 3.13: Skizzierung des Netzwerk-basierten Transferlernens, bei welchem ein NN auf einer Domäne trainiert (*Source Domain*) und anschließend auf einer anderen Domäne angewendet wird (*Target Domain*) [90].

Für die hybride Herangehensweise in dieser Arbeit wird das trainierte NN an einer vordefinierten Schicht abgeschnitten und so die interne Merkmalsrepräsentation des Netzes in dieser Schicht für die Trainingsdaten berechnet. Die transformierten Merkmale dienen nun als Grundlage für die Klassifikatormodelle SVM und RF, welche auf dem transformierten Merkmalsraum trainieren und anschließend auf den ebenfalls durch das NN transformierten Testdaten evaluiert werden.

4 Evaluierung

4.1 Versuchsaufbau

Im Folgenden wird der Versuchsaufbau mit den verwendeten Instrumenten, Merkmalen, Datensätzen, Augmentierungen, Bewertungskriterien und Klassifikatormodellen beschrieben. Die verwendeten Bibliotheken können zudem in Anhang A.1 eingesehen werden.

4.1.1 Teststruktur

Bei den Klassifikationstests dieser Arbeit wird das Multiklassenproblem gelöst. Sollen n Instrumente klassifiziert werden, so ist die Ausgabe der Modelle ein n-dimensionaler Vektor, welcher die Konfidenz des Klassifikators angibt, dass zu den eingegebenen Daten die jeweiligen Instrumentenaktivierungen vorliegen.
Je nach Klassifikator wird die Klassenentscheidung nach einem Grenzwert getroffen. Für die Sigmoidfunktion der künstlichen neuronalen Netze entspricht der übliche Grenzwert $0,5$ und für die Stützvektormaschine, welche im Intervall $[-1, 1]$ klassifiziert, beträgt der Grenzwert 0. Darüber hinaus können auch beliebige weitere, im Intervall der Funktion liegende, Grenzwerte gewählt werden. Je geringer der Grenzwert, desto mehr positive Klassenentscheidungen trifft der Klassifikator und vice versa. Während in dieser Arbeit der Ansatz nicht weiter verfolgt wird, kann auch auf Basis einer optimierenden Wahl des Grenzwertes die Klassifikationsleistung verbessert werden [53].
In dieser Arbeit werden die folgenden 6 Instrumente klassifiziert:

Instrument	Kategorie
Bratsche	Streichinstrument
Sitar	Zupfinstrument
Violine	Streichinstrument
Trompete	Blasinstrument
Flöte	Blasinstrument
Piano	Schlaginstrument

4.1.2 Merkmalsextraktion

Als Merkmale werden die jeweils PCEN-normalisierten Magnituden- und Leistungs-Mel-Spektrogramme des Audioausschnitts verwendet. Wie im Anhang in Tabelle A.4 zu sehen, erreichen NN-Modelle trainiert auf der Kombination aus beiden Spektrogrammen im Schnitt eine leicht bessere Klassifikationsleistung als auf einfachen Spektrogrammen. Die Audiolänge der extrahierten Beispiele beträgt eine Sekunde. Die Abtastrate der Audiodateien beträgt 16.000 Hz und bei der Berechnung der Spektrogramme wird die Fenstergröße der STFT auf 2048 und die Sprungweite (*Hop-Length*) auf 1024 gesetzt. Zusätzlich werden die beiden Spektrogramme mit der in Abschnitt 2.3.3 vorgestellten PCEN-Methode normalisiert. Die genaue Merkmalsextraktion kann in Anhang A.2 eingesehen werden. Die Merkmalsraumgröße für den einsekündigen Audioausschnitt beträgt (16, 128, 2).

4.1.3 Verwendete Datensätze

Die folgenden Datensätze werden zum Trainieren und Evaluieren der Klassifikatormodelle benutzt. Zusätzlich zu den 6 Instrumenten, werden auch Beispiele ohne Aktivierungen dieser Instrumente unter dem Bezeichner „Sonstige“ miteinbezogen.

Akkorde ethnischer und westlicher Instrumente

Beschreibung

Der Akkord-Datensatz (ADS) enthält 6-sekündige Akkorde aus jeweils 3 Instrumenten, welche gleichzeitig verschiedene Töne anspielen. Die Instrumente sind westlicher und ethnischer Art [96]. Für das Training und die Evaluierung wird jeweils die erste Sekunde der Audiodaten als Beispiel extrahiert.

Instrumente	Dateien	Beispiele
37	6000	6000

Tabelle 4.1: Die Gesamtanzahl der Instrumente, Dateien und extrahierten Trainingsbeispiele im Akkord-Datensatz.

Instrument	Bratsche	Sitar	Violine	Trompete	Flöte	Piano	Sonstige
Absolut	979	377	880	906	825	1026	1973
Relativ	0,1632	0,0628	0,1467	0,151	0,1375	0,171	0,3288

Tabelle 4.2: Die absolute Anzahl der Instrumentenaktivierungen, sowie das relative Verhältnis zur Gesamtanzahl der Beispiele im Akkord-Datensatz.

Künstlicher MIDI-Datensatz

Der künstliche MIDI-Datensatz (MDS) besteht aus Stücken, welche aus MIDI-Dateien künstlich generiert wurden. Die einzelnen Stücke sind von unterschiedlicher Länge und enthalten insgesamt 11 Instrumente. Für diese Arbeit werden die zwei enthaltenen Pianoarten *ElectricPiano* und *Piano* zu einem Instrument zusammengefasst. Mithilfe der MIDI-Annotierungen können Beispiele bei jeder Notenaktivierung eines Instruments exakt erstellt werden. Enthält ein Beispiel aufgrund des einsekündigen Eingabefensters mehrere unterschiedliche Aktivierungen, so werden die entsprechenden Aktivierungen mit einem logischen Oder zusammengefasst.

Instrumente	Dateien	Beispiele
10	201	75457

Tabelle 4.3: Die Gesamtanzahl der Instrumente, Dateien und Trainingsbeispiele im MIDI-Datensatz.

Instrument	Bratsche	Sitar	Violine	Trompete	Flöte	Piano	Sonstige
Absolut	11463	22714	12171	11783	11950	43707	6771
Relativ	0,1512	0,2995	0,1605	0,1554	0,1576	0,5764	0,0892

Tabelle 4.4: Die absolute Anzahl der Instrumentenaktivierungen, sowie das relative Verhältnis zur Gesamtanzahl der Beispiele im MIDI-Datensatz.

4.1.4 Kreuzvalidierung und Aufteilung

Die beiden Datensätze werden in 5 Untermengen aufgeteilt, über welche mittels einer verschachtelten Kreuzvalidierung evaluiert wird. Zunächst wird mit einer 5-fachen Kreuzvalidierung der Datensatz in Test- und Experimente-Menge im Verhältnis 1:4 aufgeteilt. Die Experimente-Menge wird dann zusätzlich mit einer 4-fachen Kreuzvalidierung im Verhältnis 1:3 in Validierungs- und Trainingsdaten geteilt. Es ergibt sich so eine Gesamtzahl von 20 Iterationen.
Die vom jeweiligen Datensatz vorliegenden Dateien werden als Ganzes einer der 5 Untermengen zugewiesen, um insbesondere bei dem in Abschnitt 4.1.3 vorgestellten MIDI-Datensatz eine Überanpassung an die einzelnen Stücke zu vermeiden.
Das vorliegende Problem eine optimale (repräsentative) Aufteilung zu erhalten, bei der die Abweichung der Anzahl an Instrumenten (und Negativbeispielen) zwischen den Untermengen minimiert wird, entspricht dem nicht-binären multi-dimensionalen Partitionierungsproblem. Da bereits das binäre ein-dimensionale Partitionierungsproblem NP-vollständig ist [62], wird die optimale Aufteilung in dieser Arbeit durch

einen heuristischen Greedy-Algorithmus approximiert.
Bei diesem wird iterativ der Untermenge eine neue Datei zugewiesen, welche zur Iteration $n-1$ in Bezug auf Instrumentenaktivierungen und Negativbeispielen relativ am geringsten repräsentiert ist. Die Implementierung in Python kann in Anhang A.3 nachvollzogen werden. Die Aufteilung des MIDI-Datensatzes ist in Grafik 4.1 zu sehen. Weitere Indikatoren der Repräsentativität wie Lautstärke und Tonhöhe sind als Erweiterungen für fortführende Arbeiten ebenfalls möglich.

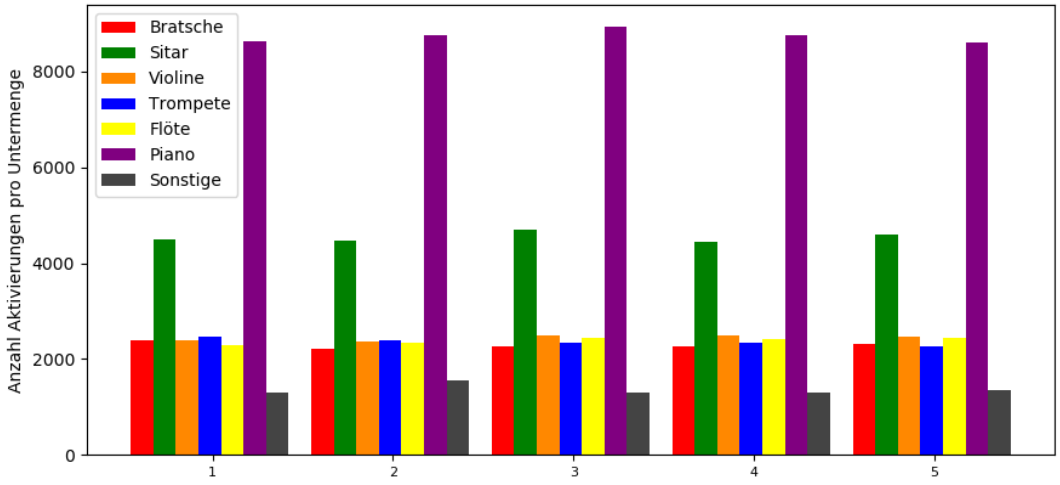


Abbildung 4.1: Anzahl der Instrumentenaktivierungen der 5 Kreuzvalidierungsmengen beim MIDI-Datensatz.

4.1.5 Generierte Augmentierungen

Für die Erstellung von Augmentierungsdaten wird eine Python-Version [37] der Bibliothek *Audio Degradation Toolbox (ADT)* [60] verwendet. Von den Audiodaten werden vor der Merkmalsextraktion insgesamt 5 verschiedene Augmentierungen erstellt. Somit entsteht eine Vergrößerung der Daten um den Faktor 6. Die gewählten Augmentierungen sind außerdem den zwei Kategorien „Lautstärke“ (AKV) und „Rauschen und Kompression“ (AKN, im Folgenden auch nur als Kategorie „Rauschen“ bezeichnet) zugeordnet und sind in Tabelle 4.5 aufgelistet.
Es werden die in Tabelle 4.6 gelisteten Kombinationen von Augmentierungskategorien untersucht.

Augmentierung	Kategorie	ADT Parameter/Preset
Lauter	AKV	`name:gain, volume:20`
Leiser	AKV	`name:gain, volume:-20`
Rauschen (weiß)	AKN	`name:noise, color:white`
Dynamikkompression (DRC)	AKN	`name:dynamic_range_compression`
Raumhall	AKN	`live_recording.json` (Anhang A.5)

Tabelle 4.5: Die Parameter und Kategorien der generierten Augmentierungen.

Kategorie	Grunddaten	Lautstärke	Rauschen und Kompression
Keine	✓		
Lautstärke	✓	✓	
Rauschen	✓		✓
Alle	✓	✓	✓

Tabelle 4.6: Die untersuchten Kombinationen der Augmentierungskategorien.

4.1.6 Bewertungskriterien

Als ein Bewertungskriterium (im Folgenden auch „Maß“) der Modelle auf den einzelnen Instrumenten wird der auch in [95] und [96] verwendete balancierte Klassifizierungsfehler (BKF) herangezogen. Dieser setzt sich aus dem Mittel der Falsch-Negativen-Rate (FNR) und der Falsch-Positiven-Rate (FPR) zusammen. Definiert man die klassifizierten Beispieluntermengen wie folgt:

- TN: korrekt klassifizierte Negativbeispiele,
- TP: korrekt klassifizierte Positivbeispiele,
- FP: falsch klassifizierte Negativbeispiele,
- FN: falsch klassifizierte Positivbeispiele,

so ergibt sich der balancierte Klassifizierungsfehler e zu:

$$e = \frac{1}{2} \cdot \left(\frac{FN}{FN + TP} + \frac{FP}{FP + TN} \right) \tag{4.1}$$

Als weiteres Bewertungskriterium wird das F-Maß verwendet [15], welches sich als harmonisches Mittel aus Präzision (*Precision*) und Sensitivität (*Recall*) als $f1$ wie folgt ergibt:

$$f1 = \frac{2TP}{2TP + FP + FN} \tag{4.2}$$

Training und Evaluierung werden für jedes Modell gemäß der Kreuzvalidierung ausgeführt. Anschließend werden die Kriterien für jedes einzelne Instrument ausgewertet und zusätzlich die durchschnittliche Klassifikationsleistung berechnet.
Beim Bewerten der Ergebnisse ist darauf zu achten, dass ein hoher F-Maß-Wert und ein niedriger BKF-Wert eine gute Klassifikationsleistung angibt.

4.1.7 Klassifikatormodelle

In den folgenden Abschnitten werden Aufbau und Parameter der verwendeten Klassifikatormodelle beschrieben.

Künstliches neuronales Netz

Die Architektur für das künstliche neuronale Netz ist zum größten Teil von der Architektur aus [35] übernommen. Erweiterungen für diese Arbeit sind das Hinzufügen von Batch-Normalisierung, sowie Anpassungen von Poolingschichten an die kleinere Eingabegröße. In der im Anhang gelisteten Tabelle A.2 ist zu sehen, dass das Hinzufügen von Batch-Normalisierung die Klassifikationsleistung der NN-Modelle signifikant steigert. Die genaue Architektur des NN ist in Anhang A.4 zu sehen und enthält eine Gesamtzahl von 1.444.742 Trainingsparametern.
Das Training erfolgt mit einer Batchgröße von 64. Als Kostenfunktion wird die binäre Kreuzentropie und als Optimierer für den Gradientenabstieg ein ADAM-Optimierer [31, K. 8.5.4][44] mit Lernrate $0,001$ verwendet. Das Training wird beendet, wenn auf den Validierungsdaten innerhalb von 5 Epochen keine Verbesserung des balancierten Klassifizierungsfehlers (siehe Abschnitt 4.1.6) mehr stattfindet. Durch diesen Ansatz des frühen Abbrechens (siehe Abschnitt 3.2.3) soll sowohl die Überanpassung, als auch die Trainingsdauer verringert werden. Die Gewichte des bis dahin besten Netzes werden als finales Netz übernommen.

Stützvektormaschine

Die Stützvektormaschine wird mit der *One-vs-rest*-Strategie (siehe Abschnitt 3.4.5) und den in Tabelle 4.7 aufgelisteten Parametern trainiert.
Die Wahl des C-Parameters auf den Wert $0,1$ folgt aus Vortests nach dem BKF, zu sehen im Anhang in Tabelle A.3. Das Training der SVM wird mit der Bibliothek Scikit-learn und den enthaltenen Klassen `SVC` und `OneVsRestClassifier` realisiert.

Parameter	Kurzbeschreibung	Wert
`C`	Schlupfvariablenfaktor	`0,1`
`kernel`	Kernelfunktion	`'rbf'`
`gamma`	Kernelkoeffizient	`'scale'`
`class_weight`	Klassengewichtung	`'balanced'`

Tabelle 4.7: Die Parameter der Stützvektormaschine.

Random-Forest-Klassifikator

Der Random-Forest-Klassifikator wird ebenfalls mit Scikit-learn und der Klasse `RandomForestClassifier` mit den in Tabelle 4.8 gelisteten Parametern trainiert.

Parameter	Kurzbeschreibung	Wert
`n_estimators`	Baumanzahl	`1000`
`criterion`	Baumaufteilungskriterium	`'gini'`
`max_depth`	Maximale Baumtiefe	`None` (keine Begrenzung)
`max_features`	Max. Merkmalsanzahl bei Teilung	`'auto'`($\widehat{=}\sqrt{\sharp Merkmale}$)

Tabelle 4.8: Die Parameter des Random-Forest-Klassifikators.

Hybride Methoden

Zum Erstellen der hybriden Methoden wird das NN trainiert, nach der letzten Faltungsschicht (siehe Schicht 23 in Anhang A.4) abgeschnitten und die Ausgabe zu einem Merkmalsraum von $2 \cdot 4 \cdot 256 = 2048$ Dimensionen vektorisiert.
Auf dem transformierten Merkmalsvektor werden die Modelle SVM und RF mit den beschriebenen Parametern trainiert. Als Vergleichswert werden auch die Modelle SVM und RF auf dem zeitlichen Durchschnitt der Mel-Spektrogramme trainiert. Dabei wird die Eingabedimension von (16, 128, 2) auf 256 reduziert. Es ergeben sich somit fünf Klassifikatormodelle, welche in Tabelle 4.9 zusammengefasst sind.

Name	Abkürzung	Merkmale
Random-Forest-Klassifikator	RF	Zeitlicher Durchschnitt der MS
Stützvektormaschine	SVM	Zeitlicher Durchschnitt der MS
Künstliches neuronales Netz	NN	Mel-Spektrogramme
NN-RF-Hybrid	NN-RF	NN-Transformierte Merkmale
NN-SVM-Hybrid	NN-SVM	NN-Transformierte Merkmale

Tabelle 4.9: Liste der evaluierten Klassifikatormodelle.

4.2 Experimente

Im Folgenden werden die Experimente dieser Arbeit durchgeführt. Die aufgestellten Hypothesen werden außerdem mithilfe des P-Tests und einem Signifikanzniveau von $0,05$ statistisch überprüft.
Für eine kompaktere Darstellung werden insbesondere in den Tabellen meist nur die im Abkürzungsverzeichnis zu findenden Abkürzungen verwendet.

4.2.1 Vergleich von hybriden und nicht-hybriden Methoden

DS	Maß	Modell	RF	SVM	NN	NN-RF	NN-SVM
Akkord-Datensatz	BKF	RF	0,4775	**+0,1697**	**+0,2344**	**+0,1982**	**+0,3056**
		SVM	**-0,1697**	0,3077	**+0,0647**	**+0,0285**	**+0,1359**
		NN	**-0,2344**	**-0,0647**	0,2430	**-0,0362**	**+0,0712**
		NN-RF	**-0,1982**	**-0,0285**	**+0,0362**	0,2792	**+0,1074**
		NN-SVM	**-0,3056**	**-0,1359**	**-0,0712**	**-0,1074**	0,1718
	F-Maß	RF	0,0750	**-0,2978**	**-0,5031**	**-0,4763**	**-0,5584**
		SVM	**+0,2978**	0,3728	**-0,2053**	**-0,1785**	**-0,2606**
		NN	**+0,5031**	**+0,2053**	0,5781	**+0,0268**	**-0,0553**
		NN-RF	**+0,4763**	**+0,1785**	**-0,0268**	0,5513	**-0,0821**
		NN-SVM	**+0,5584**	**+0,2606**	**+0,0553**	**+0,0821**	0,6334
MIDI-Datensatz	BKF	RF	0,4474	**+0,1934**	**+0,3135**	**+0,3060**	**+0,3335**
		SVM	**-0,1934**	0,2540	**+0,1201**	**+0,1126**	**+0,1401**
		NN	**-0,3135**	**-0,1201**	0,1339	-0,0075	**+0,0200**
		NN-RF	**-0,3060**	**-0,1126**	+0,0075	0,1414	**+0,0275**
		NN-SVM	**-0,3335**	**-0,1401**	**-0,0200**	**-0,0275**	0,1139
	F-Maß	RF	0,2453	**-0,3542**	**-0,5814**	**-0,5933**	**-0,5985**
		SVM	**+0,3542**	0,5995	**-0,2273**	**-0,2391**	**-0,2444**
		NN	**+0,5814**	**+0,2273**	0,8267	-0,0119	**-0,0171**
		NN-RF	**+0,5933**	**+0,2391**	+0,0119	0,8386	-0,0053
		NN-SVM	**+0,5985**	**+0,2444**	**+0,0171**	+0,0053	0,8439

Tabelle 4.10: Vergleich der Modelle ausgewertet für die durchschnittliche Klassifikationsleistung auf den Instrumenten.

Die fünf Modelle werden gemäß der Kreuzvalidierung trainiert und anhand der Bewertungskriterien evaluiert. In der Tabelle 4.10 sind die Gegenüberstellungen der Klassifikatormodelle nach den Datensätzen und Bewertungskriterien zu sehen. Die Ergebnisse stellen dabei den Durchschnitt der Klassifikationsleistung über die einzelnen Instrumente dar, Vergleiche auf den einzelnen Instrumenten können in Anhang A.7 eingesehen werden. In den jeweiligen Diagonalen der Tabellenabschnitte sind

die erzielten Werte des Klassifikators nach dem gegebenen Kriterium und Datensatz gelistet. In den anderen Tabellenelementen ist jeweils die Differenz des Klassifikatormodells aus der Zeile mit der Spalte abgebildet. Ist die Differenz statistisch signifikant, so wird der Wert fett ausgeschrieben. Die genauen Werte des P-Tests beim Vergleich der Klassifikatormodelle sind zusätzlich in Anhang A.8 gelistet.

Im Folgenden werden die entsprechenden Hypothesen aufgestellt und ausgewertet.

Vergleich von hybriden und nicht-hybriden Methoden

Die Null-Hypothese ist hier wie folgt gegeben: Hybride Methoden erreichen ohne Augmentierungen den gleichen Klassifizierungsfehler wie nicht-hybride Ansätze.
Für den hybriden Ansatz NN-RF kann die Null-Hypothese im Vergleich zu den Basis-Klassifikatoren RF und SVM für jeden Datensatz und jedes Bewertungskriterium verworfen werden. Aus der Tabelle 4.10 ist abzulesen, dass der NN-RF Ansatz bessere Ergebnisse erzielt. Vergleicht man NN-RF mit NN so kann die Null-Hypothese für den MIDI-Datensatz nicht verworfen werden, da die Unterschiede sich sowohl für das F-Maß, als auch den balancierten Klassifizierungsfehler nicht signifikant unterscheiden. Für den Akkord-Datensatz lässt sich die Null-Hypothese verwerfen, da der hybride Ansatz NN-RF nach beiden Maßen schlechter klassifiziert als das NN.
Der hybride Ansatz NN-SVM erzielt gegenüber jeder nicht-hybriden Methode ein signifikant besseres Klassifizierungsergebnis, sowohl nach beiden Bewertungskriterien, als auch auf beiden Datensätzen.

Vergleich von hybriden Methoden

Die Null-Hypothese ist hier wie folgt gegeben: Die hybride Methode NN-RF erreicht den gleichen Klassifizierungsfehler wie die hybride NN-SVM Methode.
Aus der Tabelle 4.10 lässt sich ablesen, dass die Methode NN-SVM auf dem Akkord-Datensatz nach beiden Bewertungskriterien signifikant besser als der NN-RF Ansatz ist. Auch für den MIDI-Datensatz kann die Null-Hypothese nach dem balancierten Klassifizierungsfehler zugunsten der NN-SVM Methode verworfen werden. Werden die Ergebnisse nach dem F-Maß auf dem MIDI-Datensatz gemessen, so lässt sich kein signifikanter Unterschied zwischen NN-RF und NN-SVM feststellen.

4.2.2 Vergleich von Augmentierungskategorien

In den Abbildungen 4.2 ist ein grafischer Vergleich der Klassifikatormodelle NN, NN-RF und NN-SVM mit den in Abschnitt 4.1.5 definierten Augmentierungskategorien zu sehen.

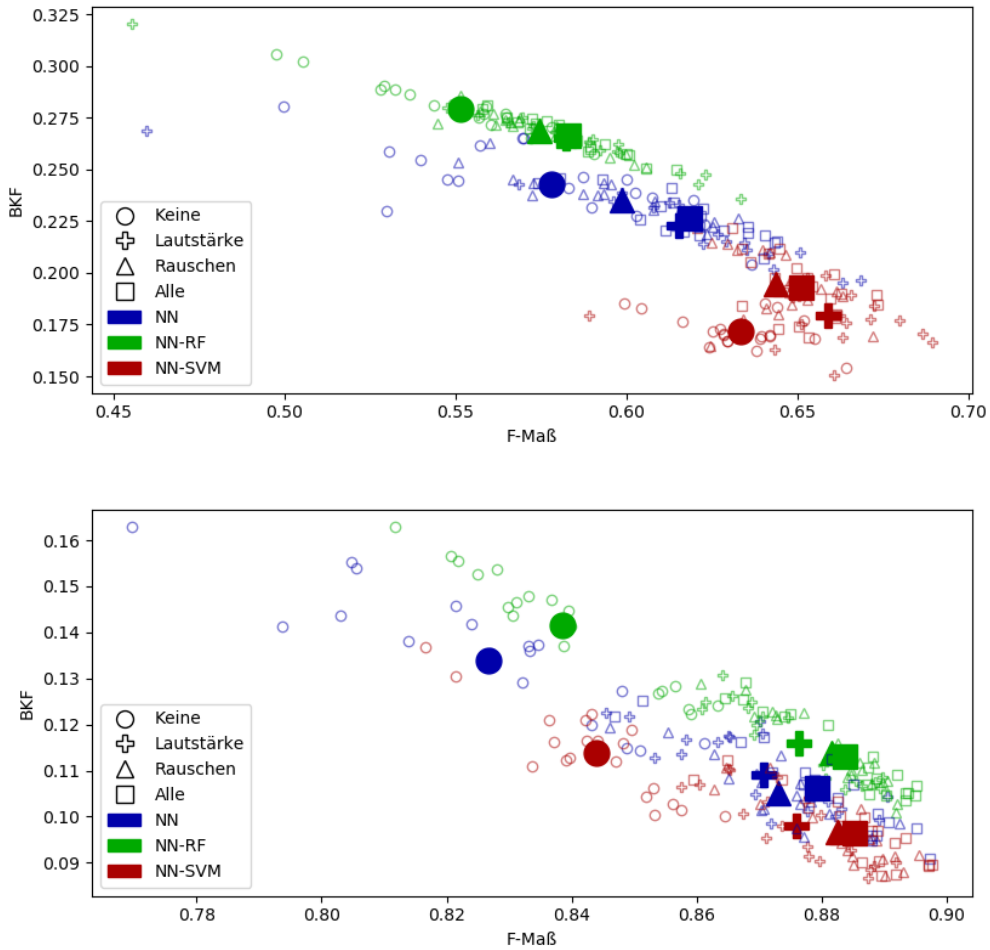


Abbildung 4.2: Grafischer Vergleich der Augmentierungskategorien (Symbole) für die farblich gekennzeichneten Modelle NN, NN-RF und NN-SVM auf dem Akkord-Datensatz (oben) und dem MIDI-Datensatz (unten) mit den einzelnen Testergebnissen der Kreuzvalidierung und dem jeweiligen Durchschnitt (gefüllte Symbole).

Des Weiteren findet sich in den Tabellen 4.11 und 4.12 eine komplette Gegenüberstellung sämtlicher Klassifikatormodelle und Augmentierungskategorien. Die farblichen Visualisierungen entsprechen dabei den folgenden Bewertungen des Modells der Zeile im Vergleich mit dem Modell der Spalte, gemessen an der durchschnittlichen Klassifikationsleistung auf den einzelnen Instrumenten:

- Das Modell klassifiziert nach beiden Maßen statistisch signifikant besser.
- Das Modell klassifiziert nach einem Maß statistisch signifikant besser.
- Das Modell klassifiziert nach beiden Maßen statistisch signifikant schlechter.
- Das Modell klassifiziert nach einem Maß statistisch signifikant schlechter.
- Das Modell klassifiziert nach beiden Maßen nicht signifikant unterschiedlich.
- Das Modell klassifiziert nach einem Maß signifikant besser und nach dem anderen Maß signifikant schlechter.

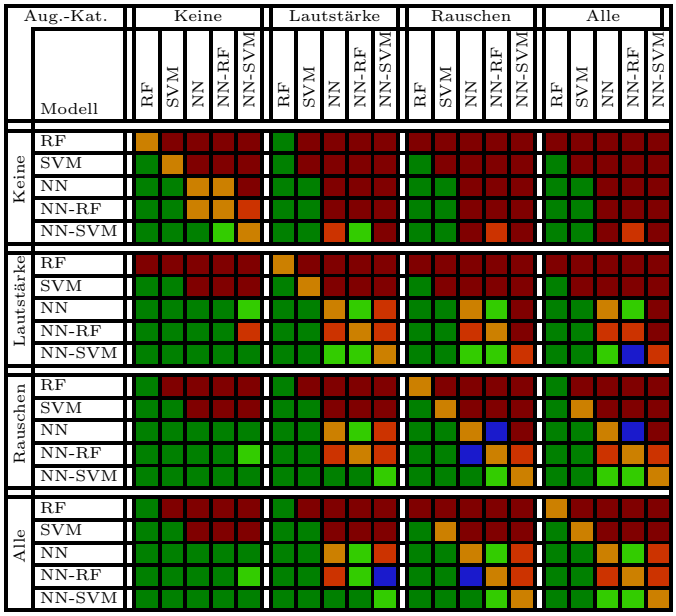


Tabelle 4.11: Gegenüberstellung aller Modelle trainiert auf verschiedenen Augmentierungskategorien auf dem MIDI-Datensatz.

Im Folgenden werden für die Klassifikatormodelle NN, NN-RF und NN-SVM die Augmentierungskategorien nach den in Anhang A.9 gelisteten Ergebnissen ausgewertet. In den dortigen Diagonalen der jeweiligen Tabellenabschnitte befindet sich die Klassifikationsleistung nach dem gegebenen Maß, Modell und Datensatz. Die restlichen Tabellenelemente stellen jeweils analog zu den Klassifikatormodellvergleichen aus Abschnitt 4.2.1 sowohl farblich als auch numerisch die Differenz zwischen der Zeile und der Spalte dar.
Es wird für die drei Modelle die folgende Null-Hypothese überprüft:
Wird das gegebene Klassifikatormodell auf verschiedenen Augmentierungskategorien trainiert, so ist die Klassifikationsleistung gleich.

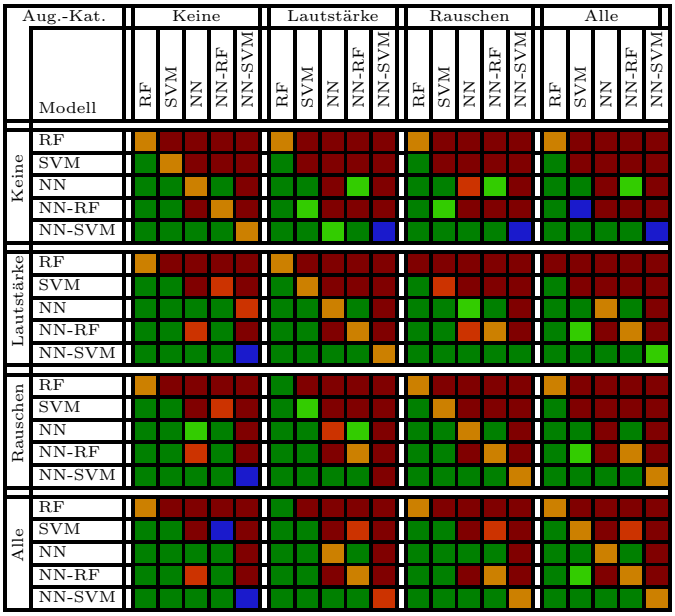


Tabelle 4.12: Gegenüberstellung aller Modelle trainiert auf verschiedenen Augmentierungskategorien auf dem Akkord-Datensatz.

NN

MIDI-Datensatz: Wie in Tabelle A.15 zu sehen, ist für das NN jegliche Art der Augmentierung signifikant besser, als ein Training auf unaugmentierten Daten. Der Unterschied zwischen den restlichen, einzelnen Augmentierungsstufen ist allerdings statistisch nicht signifikant.
Akkord-Datensatz: Aus der Tabelle kann ebenfalls abgelesen werden, dass NN-Modelle auf dem Akkord-Datensatz signifikant bessere Klassifikationsleistungen erzielen, wenn Lautstärke-augmentierte Daten ins Training einbezogen werden. Augmentierungen der Kategorie „Rauschen“ bringen lediglich nach dem F-Maß signifikante Verbesserungen und sind insbesondere im direkten Vergleich zum Training mit Augmentierungen der Kategorie „Lautstärke“ nach dem BKF signifikant schlechter.

NN-RF

MIDI-Datensatz: Der NN-RF Ansatz erzielt wie in Tabelle A.16 zu sehen, ebenfalls signifikante Verbesserungen, wenn augmentierte Daten den unaugmentierten gegenübergestellt werden. Ein Unterschied zu den unhybriden NN-Modellen ist, dass das Training auf allen Augmentierungskategorien nach dem F-Maß signifikant bessere Ergebnisse erzielt, als ein Training auf Lautstärke-Augmentierungen.
Akkord-Datensatz: Aus der Tabelle kann auch für die NN-RF-Variante abgelesen werden, dass zwar jegliche Art der Augmentierung signifikant besser ist, als ein Trai-

ning auf unaugmentierten Daten, der Unterschied zwischen den restlichen, einzelnen Augmentierungsstufen allerdings statistisch nicht signifikant ist.

NN-SVM

MIDI-Datensatz: Analog zu den Ergebnissen des NN-RF kann auch für den NN-SVM Ansatz der Schluss gezogen werden, dass beim MIDI-Datensatz Augmentierungen jeder Art die Klassifikationsleistung verbessern (siehe Tabelle A.17). Eine weitere Erkenntnis ist, dass die Augmentierungs-Kategorien, welche Rausch-Augmentierungen enthalten unter Betrachtung des F-Maßes signifikante Verbesserungen zeigen, im Vergleich zu Augmentierungen der Kategorie „Lautstärke".
Akkord-Datensatz: Während auf dem MIDI-Datensatz insbesondere Augmentierungskategorien mit verrauschten Daten, die Klassifikationsleistung konsistent und signifikant steigern, so sind die Ergebnisse auf dem Akkord-Datensatz für den NN-SVM Ansatz nicht eindeutig. Wird nach dem F-Maß gemessen, so ergeben sich die gleichen Schlussfolgerungen der vorangegangen Auswertungen, dass jegliche Art von Augmentierung die Klassifikationsleistung signifikant steigert. Wird allerdings nach dem BKF gemessen, so lässt sich eine signifikante Verschlechterung der Klassifikationsleistung für jede Augmentierungskategorie feststellen. Eine kurze Untersuchung für die unterschiedlich ausfallenden Maße, welche auch als blaue Markierungen in den Tabellen 4.11 und 4.12 zu finden sind, wird im Folgenden genauer thematisiert.

Maße / Aug.	Präzision	Sensitivität	F-Maß	FNR	FPR	BKF
Keine	0,5440	0,8056	0,6334	0,1944	0,1492	0,1718
Lautstärke	0,6079	0,7408	0,6589	0,2592	0,0991	0,1792
Rauschen	0,6067	0,7006	0,6436	0,2994	0,0896	0,1945
Alle	0,6202	0,6976	0,6511	0,3024	0,0835	0,1930

Tabelle 4.13: Gegenüberstellung der Maße auf Basis ihrer Bewertungsgrundlagen nach verschiedenen Augmentierungskategorien für das Modell NN-SVM auf dem Akkord-Datensatz.

In Tabelle 4.13 sind sowohl die Bewertungsmaße F-Maß und BKF, sowie deren Grundmaße Präzision, Sensitivität, Falsch-Positiven-Rate und Falsch-Negativen-Rate zu sehen. Aus der Tabelle ist abzulesen, dass das Training des NN-SVM Ansatzes mit zusätzlichen Augmentierungen zu einer höheren Tendenz der Modelle führt, negative Klassifizierungsentscheidungen zu treffen. Folglich ergibt sich eine schlechtere Sensitivität und Falsch-Negativen-Rate, während die anderen Grundmaße nach dem jeweiligen Maß bessere Werte ergeben. Der Bewertungsunterschied der beiden Maße

ist bedingt durch die harmonische Mittelung des F-Maßes und der einfachen Mittelung des balancierten Klassifizierungsfehlers. Mögliche Erklärungen für das Zunehmen von negativen Klassifizierungsentscheidungen sind zum einen das in Abschnitt 3.4.5 erwähnte Binärisierungsproblem der SVM und zum anderen die im Schnitt höhere Rate an Negativbeispielen im Akkord-Datensatz (siehe Tabellen 4.2 und 4.4). Durch die zusätzlichen Augmentierungsdaten kann der Effekt so zusätzlich verstärkt werden.

4.2.3 Generalisierungstest

Im Generalisierungstest werden die Modelle, welche auf dem einen Datensatz trainiert wurden, auf dem jeweils anderen Datensatz evaluiert, zu sehen im Vergleich in Abbildung 4.3. Die Ergebnisse bei Verwendung von Augmentierungen kann zusätzlich in Anhang A.10 eingesehen werden.

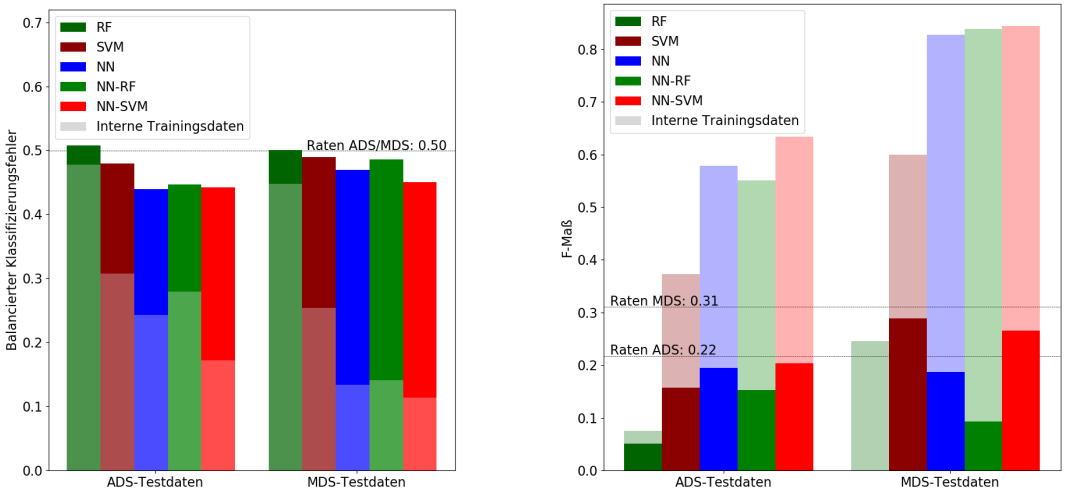


Abbildung 4.3: Ergebnisse der Generalisierungstests nach dem BKF (links) und dem F-Maß (rechts). Farblich visualisiert sind die verschiedenen Modelle, trainiert auf dem jeweils anderen Datensatz, sowie zum Vergleich die Klassifikationsleistung der Modelle, wenn auf den datensatzinternen Daten trainiert wird (farblich schwach).

Abweichung zu den internen Testergebnissen

Die Null-Hypothese ist für den Vergleich wie folgt gegeben:
Die durchschnittliche Klassifikationsleistung auf den Instrumenten des Modells weicht auf dem anderen Datensatz nicht von den Ergebnissen aus Abschnitt 4.2.1 ab.

Wie in Abbildung 4.3 zu sehen ist die Klassifikationsleistung auf dem anderen Datensatz jeweils deutlich und auch statistisch signifikant schlechter, weshalb die Null-Hypothese für jedes Maß und Modell verworfen werden kann.

Vergleich mit statistischem Raten

Zum Vergleich mit statistischem Raten wird ein Zufalls-Klassifikator ausgewertet, welcher jeweils bei jedem Instrument mit 50% Wahrscheinlichkeit eine positive Klassenentscheidung trifft.
Null-Hypothese: Der Klassifikationsfehler der Modelle auf dem jeweils anderen Datensatz unterscheidet sich nicht signifikant vom Klassifikationsfehler des Zufalls-Klassifikators.
Die Null-Hypothese kann für das F-Maß vollständig verworfen werden, da alle Modelle sogar signifikant schlechter als statistisches Raten abschneiden.
Nach dem BKF sind die Modelle bis auf das RF-Modell jedoch auf dem anderen Datensatz signifikant besser als einfaches statistisches Raten.

Vergleich von hybriden und nicht-hybriden Modelle

Null-Hypothese: Hybride Methoden erreichen eine gleichwertige Generalisierung wie nicht-hybride Ansätze.
Die Null-Hypothese lässt sich für den NN-RF Ansatz im Vergleich zum RF für jedes Maß und Datensatz zugunsten des NN-RF Ansatzes verwerfen. Im Vergleich mit der SVM kann nur nach dem BKF eine Verbesserung festgestellt werden. Nach dem F-Maß ist der NN-RF trainiert auf dem MDS nicht signifikant besser und trainiert auf dem ADS und evaluiert auf dem MDS sogar signifikant schlechter als die SVM ohne NN. Auch im Vergleich mit dem NN ist der NN-RF im Generalisierungstest nach jedem Maß und Datensatz schlechter.
Der Ansatz aus NN-SVM ist in diesem Test nach allen Maßen und Datensätzen besser als der RF und erreicht auch trainiert auf dem ADS bessere Ergebnisse als das NN. Trainiert auf dem MDS und evaluiert auf dem ADS erreicht der Ansatz aus NN-SVM keine signifikanten Unterschiede zum NN. Nach dem BKF erreicht die NN-SVM auch bessere Ergebnisse als die unhybride SVM, allerdings schneidet der Ansatz aus NN-SVM signifikant schlechter als die SVM nach dem F-Maß ab, wenn auf dem ADS trainiert und auf dem MDS evaluiert wird.

4.2.4 Kombinierung der Datensätze

Im Folgenden werden die zwei Datensätze zu einem kombinierten Datensatz (KDS) zusammengefasst, die jeweiligen Modelle analog zu den vorangegangenen Tests trainiert und evaluiert und die Ergebnisse mit den vorangegangenen Tests verglichen.

Kombinationsmethodik

Die Kombination der beiden Datensätze erfolgt zunächst über eine Reduzierung der Trainingsbeispiele des MIDI-Datensatzes auf den Mittelwert der Größe der beiden Datensätze, um den zeitlichen Rahmen dieser Masterarbeit einzuhalten. Da die in dieser Arbeit zum Training der SVM verwendete Klasse `OneVsRestClassifier` der Bibliothek Scikit-learn keine direkte Unterstützung für unterschiedliche Gewichtungen von Beispielen bietet, werden die geringer vorkommenden Beispiele des Akkord-Datensatzes wiederholt eingefügt, bis der Anteil der beiden Datensätze in den Trainingsdaten ausgeglichen ist.
Es werden in den folgenden Abschnitten die entsprechenden Null-Hypothesen zum so entstandenen kombinierten Datensatz aufgestellt und ausgewertet.

Ergebnisse

In Tabelle 4.14 kann der Vergleich der einzelnen Modelle über die Datensätze eingesehen werden. Die Spalte „Train-DS“ gibt dabei den zugrundeliegenden Trainingsdatensatz und die Spalte „Test-DS“ den Evaluierungsdatensatz an.
In der jeweils ersten Zeile „KDS“ ist die Klassifikationsleistung der Modelle zu finden, deren Trainingsgrundlage der kombinierte Datensatz ist. In den jeweils darunterliegenden Zeilen ADS und MDS sind die Differenzwerte zwischen Testergebnissen des KDS und des anderen Trainingsdatensatzes zu finden. Grün markierte Differenzen geben ein besseres und rot markierte ein schlechteres Ergebnis auf dem KDS an. Zusätzlich sind statistisch signifikante Differenzen fett ausgeschrieben. Während die vorliegende Tabelle den Durchschnitt der 6 Instrumente abbildet, kann die Gegenüberstellung auf den einzelnen Instrumenten in Anhang A.11 eingesehen werden.

Vergleich mit Ergebnissen des Generalisierungstests

Null-Hypothese: Die Ergebnisse der einzelnen Modelle weichen auf dem kombinierten Datensatz nicht signifikant von den Ergebnissen des Generalisierungstests (siehe Abschnitt 4.2.3) ab.

Test-DS	Maß	Modell / Train-DS	RF	SVM	NN	NN-RF	NN-SVM
ADS	BKF	KDS	0,4767	0,3251	0,2105	0,2578	0,2006
		ADS	-0,0007	**+0,0174**	**-0,0325**	**-0,0214**	**+0,0288**
		MDS	**-0,0308**	**-0,1547**	**-0,2289**	**-0,1893**	**-0,2414**
	F-Maß	KDS	0,0766	0,4059	0,6427	0,5920	0,6597
		ADS	+0,0016	**+0,0332**	**+0,0646**	**+0,0408**	**+0,0263**
		MDS	**+0,0259**	**+0,2485**	**+0,4475**	**+0,4389**	**+0,4562**
MDS	BKF	KDS	0,4654	0,3199	0,1539	0,1751	0,1393
		ADS	**-0,0346**	**-0,1692**	**-0,3153**	**-0,3104**	**-0,3112**
		MDS	**+0,0180**	**+0,0659**	**+0,0200**	**+0,0337**	**+0,0254**
	F-Maß	KDS	0,1948	0,5227	0,7979	0,7942	0,8103
		ADS	**+0,1939**	**+0,2334**	**+0,6102**	**+0,7009**	**+0,5448**
		MDS	**-0,0505**	**-0,0768**	**-0,0288**	**-0,0444**	**-0,0336**

Tabelle 4.14: Vergleich des kombinierten Datensatzes mit dem Akkord- und MIDI-Datensatz.

Wie in Tabelle 4.14 zu sehen, sind die Ergebnisse des kombinierten Datensatzes im Vergleich zu den Werten des Generalisierungstest (ADS-MDS und MDS-ADS) nach beiden Maßen jeweils signifikant besser.

Vergleich der Testdaten der jeweiligen Datensätze

Null-Hypothese: Die Ergebnisse der Modelle auf dem kombinierten Datensatz weichen nicht signifikant von den Ergebnissen der Datensatz-Tests (siehe Abschnitt 4.2.1) ab.

Akkord-Datensatz: Der kombinierte Datensatz erzielt auf den Testdaten des ADS nach dem F-Maß eine bessere und bis auf das RF-Modell auch signifikant bessere Klassifikationsleistung. Nach dem BKF gemessen, werden für die Modelle NN und NN-RF ebenfalls signifikant bessere Ergebnisse erzielt, während sich mit den Modellen SVM und NN-SVM eine Verschlechterung feststellen lässt.

MIDI-Datensatz: Während einige Modelle, welche auf dem KDS trainiert und auf dem ADS evaluiert werden, eine bessere Klassifikationsleistung erzielen, so erzielen dieselben Modelle, wenn sie auf dem MDS evaluiert werden nach beiden Maßen signifikant schlechtere Ergebnisse, als die Modelle, welche nur auf dem MDS trainiert werden.

Eine mögliche Schlussfolgerung aus den Ergebnissen ist, dass die am Anfang dieses Abschnitts 4.2.4 vorgestellte Halbierung der Trainingsdaten des MDS eine negative

Auswirkung auf die Klassifikationsleistung der auf dem KDS trainierten Modelle hat. Demgegenüber steht der vom Trainingsbeispielumfang kleinere ADS, auf welchem Modelle von den zusätzlichen Beispielen des MDS profitieren.

4.2.5 Vergleich von Merkmalsextraktionsschichten der hybriden Methoden

Vorgehensweise

In diesem Test werden unterschiedliche Extraktionsschichten der trainierten NN-Modelle untersucht, welche als Merkmale für hybride NN-RF und NN-SVM Methoden verwendet werden.
Für die vorangegangenen Tests wird die letzte Faltungsschicht (Conv256) (siehe Schicht 23 in Anhang A.4) verwendet. Im Folgenden werden Merkmale anhand der letzten Faltungsschicht mit 64 Kanälen (Conv64) und der globalen Poolingschicht (GMP) berechnet (siehe Schicht 11 bzw. 25 in Anhang A.4). Da der Merkmalsraum der Conv64-Schicht mit einer Größe von $16 \cdot 32 \cdot 64 = 32768$ zu groß für ein effizientes Training der Modelle ist, wird (analog zu der in Abschnitt 4.1.7 beschriebenen Vorgehensweise) der zeitliche Mittelwert der extrahierten Merkmale berechnet. Es ergibt sich so eine neue Merkmalsraumgröße von $1 \cdot 32 \cdot 64 = 2048$, was auch dem Umfang der Conv256-Schicht entspricht.

Ergebnisse

Der Vergleich über den Durchschnitt der Instrumente der drei Extraktionsschichten ist in Tabelle 4.15 zu finden. Analog zu den vorangehenden Tests sind die Tabellen zu einzelnen Instrumenten in Anhang A.12 zu finden. Die Basis des Vergleichs bietet die jeweils erste Zeile mit der Klassifikationsleistung der hybriden Methoden auf dem Merkmalsraum der Conv256-Schicht. Darunter findet sich die Klassifikationsleistung mit den anderen Extraktionsschichten, so wie der üblichen farblichen Hervorhebung, wenn Ergebnisse besser, schlechter und signifikant sind.
Aus der Tabelle kann abgelesen werden, dass bei Benutzung der GMP-Schicht für beide Maße und Datensätze keine signifikante Veränderung der Klassifikationsleistung stattfindet. Während die Klassifikationsleistung des NN-RF in drei von vier Fällen leicht gesteigert werden kann, so wird die Klassifikationsleistung des NN-SVM-Modells in drei von vier Fällen leicht verschlechtert.

Typ	DS/Maß / Schicht	ADS		MDS	
		BKF	F-Maß	BKF	F-Maß
NN-RF	Conv256[23]	0,2792	0,5513	0,1414	0,8386
	GMP[25]	0,2726	0,5636	0,1413	0,8366
	Conv64[11]	**0,4476**	**0,1243**	**0,3358**	**0,4894**
NN-SVM	Conv256[23]	0,1718	0,6334	0,1139	0,8439
	GMP[25]	0,1752	0,6358	0,1177	0,8394
	Conv64[11]	**0,2185**	**0,5098**	**0,1207**	**0,8286**

Tabelle 4.15: Vergleich verschiedener Merkmalsextraktionsschichten der hybriden Methoden.

Die Extraktion und Durchschnittsbildung nach der Conv64-Schicht führt bei beiden Modellen, Maßen und Datensätzen zu einer signifikanten Verschlechterung. Eine Ausnahme für die Verschlechterung bildet die im Anhang zu findende Tabelle A.25, welche den Vergleich über das Instrument Sitar darstellt. Die auf dem MIDI-Datensatz ohnehin im Vergleich zu den anderen Instrumenten sehr gute Erkennungsrate der Sitar kann durch das NN-SVM-Modell und die frühere Extraktion zusätzlich signifikant verbessert werden.

4.2.6 Vergleich auf verrauschten Daten des MIDI-Datensatzes

In Abbildung 4.4 ist eine grafische Visualisierung der Testergebnisse des NN-Modells auf verrauschten Daten des MIDI-Datensatzes zu sehen. Die unterschiedlichen Augmentierungskategorien sind dabei farblich visualisiert. Wie in der Legende beschrieben, bezeichnen die Symbole den Typ der verrauschten Daten, wobei unverrauschte Daten als „Vanilla“ bezeichnet werden. Neben den jeweils 5 Testergebnissen aus der ersten Verschachtelung der Kreuzvalidierung ist auch die durchschnittliche Klassifikationsleistung als größeres Symbol dargestellt. Die analoge Grafik für den Akkord-Datensatz kann in Anhang A.13.1 eingesehen werden. Im Folgenden werden Erkenntnisse aus der Grafik für den MIDI-Datensatz diskutiert.

Training ohne Augmentierungen

Wird ohne Augmentierungen trainiert, so werden vor allem lautere und Raumhall-Daten von den Modellen schlecht erkannt. Auch Daten mit weißem Rauschen werden nach beiden Maßen deutlich schlechter erkannt, als die restlichen Augmentierungsarten „Leiser“, „Vanilla“ und „DRC“.

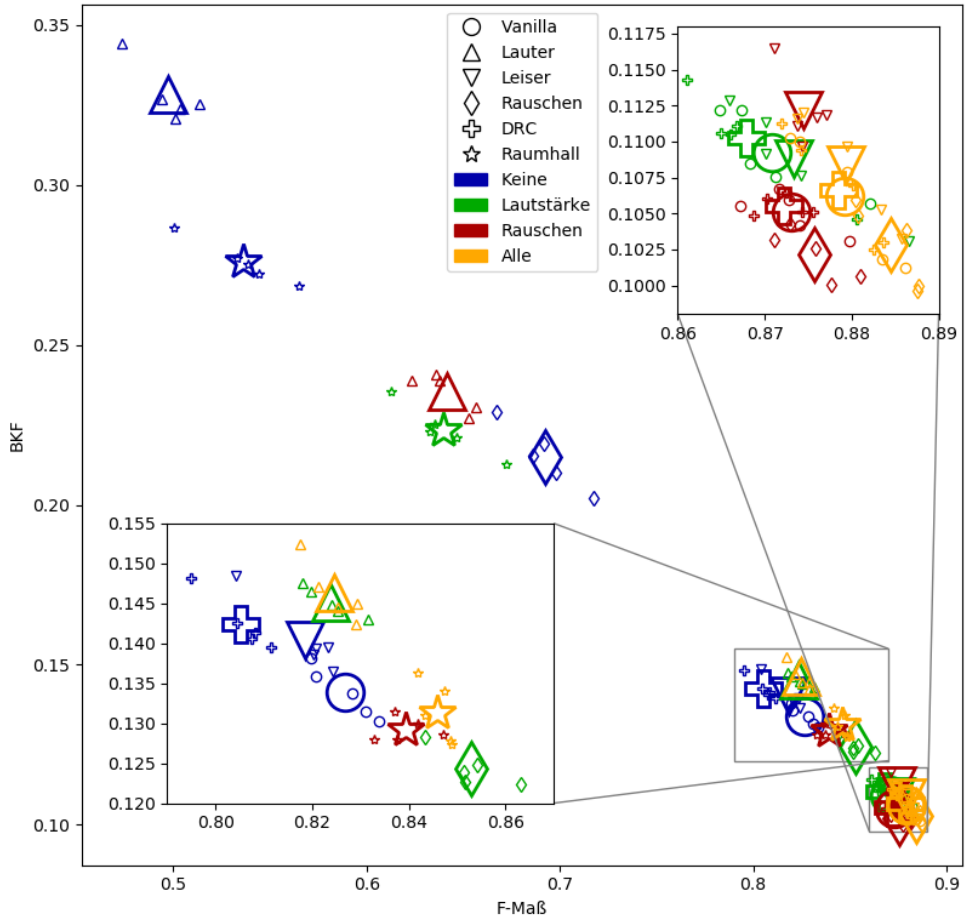


Abbildung 4.4: Vergleich von verrauschten Daten auf verschiedenen Augmentierungskategorien des Modells NN auf dem MIDI-Datensatz.

Signifikanz von lauteren und Raumhall-Augmentierungen

Neben der thematisierten schlechten Klassifikationsleistung bei unaugmentiertem Training, ist auch im mittleren Bereich der Abbildung zu beobachten, dass Modelle welche auf Augmentierungen der Kategorie Lautstärke trainiert werden, Raumhall-Daten schlecht erkennen. Ebenso weisen Modelle, welche nur auf der Kategorie Rauschen trainiert werden, eine schlechte Erkennung von lauteren Daten auf. Es kann gegebenenfalls daraus gefolgert werden, dass die zwei Augmentierungen „Lauter“ und „Raumhall“ durch die einhergehende Übersteuerung und Überlagerung den Merkmalsraum stärker verändern, als andere Augmentierungen wie beispielsweise leisere

oder DRC-augmentierte Daten. Die Wahl von signifikanten Augmentierungen wird auch im Ausblick in Abschnitt 5.2.3 abschließend thematisiert.

Erkennungsrate von Daten mit weißem Rauschen

Daten, welche mit einem weißen Rauschen hinterlegt werden, weisen im Vergleich zu anderen Augmentierungsarten eine gute Erkennungsrate auf. Zum einen erreichen Modelle, welche auf Lautstärke-Augmentierungen trainiert werden, eine bessere Klassifikationsleistung auf diesen weiß-verrauschten Daten, als auf lauteren Daten, obwohl bei letzterem der Störfaktor im Training vorhanden ist. Betrachtet man Modelle, welche unter anderem auf Daten aus der Kategorie Rauschen trainiert werden (in der Grafik rot und gelb), so erzielen diese nach beiden Maßen eine bessere Klassifikationsleistung auf Daten mit weißem Rauschen (Raute), als auf unverrauschten Daten (Kreis), was ein Hinweis auf eine mögliche Überanpassung sein kann.

4.2.7 Weitere relevante Ergebnisse

PCEN-Normalisierung vs. log-Skalierung

DS	Maß	Methode \ Aug.	Vanilla	Lauter	Leiser	Rauschen	DRC	Raumhall
ADS	BKF	log	0,2292	0,3276	0,2290	0,3056	0,2308	0,3981
		PCEN	0,2430	0,3130	**0,2501**	**0,2673**	0,2457	**0,3778**
	F-Maß	log	0,5956	0,4014	0,5952	0,4605	0,5918	0,2760
		PCEN	0,5781	0,4343	0,5661	**0,5288**	0,5744	0,3029
MDS	BKF	log	0,0941	0,3652	0,0925	0,2179	0,0938	0,1926
		PCEN	**0,1339**	**0,3282**	**0,1405**	0,2151	**0,1424**	**0,2760**
	F-Maß	log	0,8838	0,3918	0,8844	0,6742	0,8791	0,7227
		PCEN	**0,8267**	**0,4971**	**0,8185**	0,6922	**0,8053**	**0,5362**

Tabelle 4.16: Vergleich der PCEN-Normalisierung mit der log-Skalierung.

In Tabelle 4.16 ist eine Gegenüberstellung von NN-Modellen sehen, welche jeweils auf PCEN- und log-normalisierten Mel-Spektrogrammen trainiert wurden. Aus der Tabelle lässt sich ablesen, dass die PCEN-Normalisierung für lautere Daten und Daten mit weißem Rauschen bessere Ergebnisse erzielt. Bei unverrauschten, leiseren und dynamikkomprimierten Daten fällt die Klassifikationsleistung jedoch schlechter aus. Für Datenpunkte mit Raumhall ist die Datenlage unklar, da auf dem ADS mit der PCEN-Normalisierung bessere Ergebnisse erzielt werden, während die Klassifikationsleistung auf dem MDS signifikant schlechter ist.

Zusammenhang von NN und hybriden Methoden

Wie in Abbildung 4.5 zu sehen, ist die Klassifikationsleistung der NN-Modelle nicht unbedingt mit der Klassifikationsleistung der korrespondierenden NN-SVM Ansätze korreliert. So erzielen NN-SVM-Ansätze, welche auf NN-Modellen basieren, die nach beiden Maßen am schlechtesten abschneiden (1), die zweitbeste Klassifikationsleistung nach dem BKF (2). Gleichzeitig erzielen NN-SVM-Ansätze, welche auf einem überdurchschnittlich guten NN basieren (3), die schlechteste Klassifikationsleistung der NN-SVM-Ansätze (4). Eine mögliche weitere Untersuchung wird im Ausblick in Abschnitt 5.2.4 motiviert.

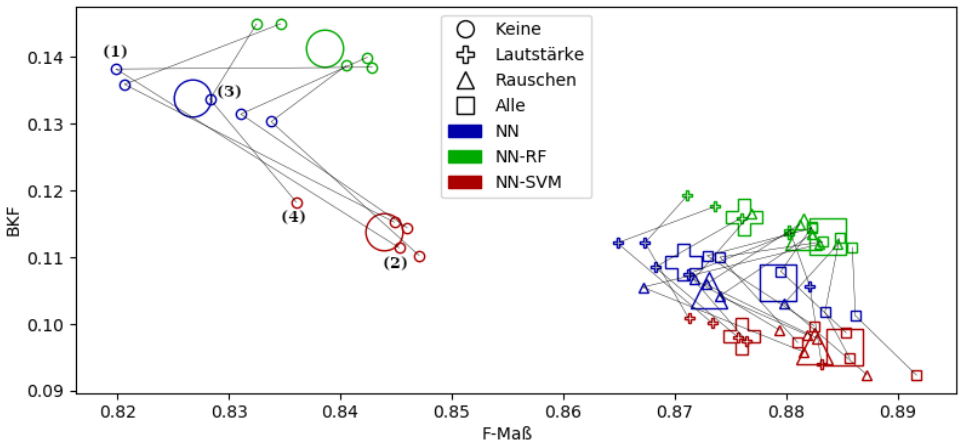


Abbildung 4.5: Vergleich der 5 Testdaten der Kreuzvalidierung auf dem MIDI-Datensatz. Die Symbole der Ergebnisse der hybriden Methoden sind jeweils mit den Ergebnissen der NN-Modelle verbunden, welche die Grundlage der hybriden Methoden bilden.

Trainingsdauer

Wie in Abbildung 4.6 zu sehen fällt die Trainingsdauer je nach Modell und Augmentierungskategorie unterschiedlich aus. Im Vergleich zu den anderen Klassifikatoren weist, bei zunehmender Anzahl an Augmentierungen, vor allem die hybride NN-SVM Version eine deutlich längere Trainingsdauer auf. Auf dem vom Umfang her kleineren Akkord-Datensatz fällt die durch die hybriden Methoden längere Trainingszeit weniger drastisch aus (siehe Anhang A.13.2).

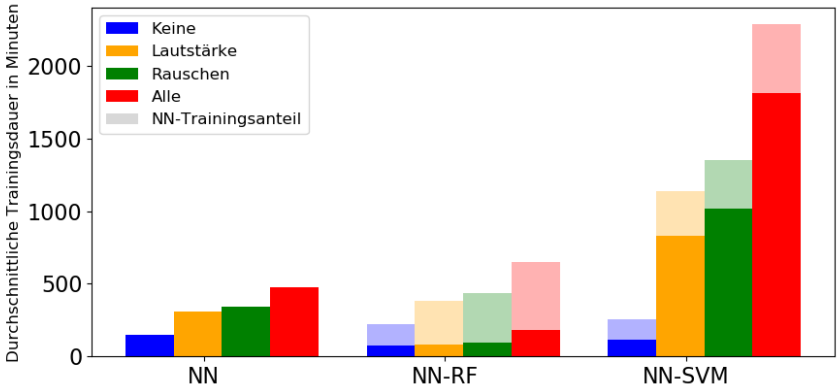


Abbildung 4.6: Vergleich der Trainingsdauer der Klassifikatoren NN, NN-RF und NN-SVM auf dem MIDI-Datensatz und den verschiedenen Augmentierungskategorien. Die zusätzliche Trainingsdauer des NN bei Erstellung der hybriden Methoden ist zudem schwach farblich visualisiert.

5 Fazit und Ausblick

5.1 Fazit

In dieser Arbeit wurden mithilfe von künstlichen neuronalen Netzen, Random-Forest-Klassifikatoren und Stützvektormaschinen Klassifikatormodelle erstellt, welche Ausschnitte aus Musikdaten nach ihren Instrumenten klassifizieren. Als Merkmale wurden die jeweils PCEN-normalisierten Magnituden- und Leistungsspektrogramme der Audioausschnitte verwendet.
Bereits vor der Evaluierung wurden Untersuchungen angestellt, welche zur Auswahl und Optimierung der verwendeten NN-Architektur führten (siehe Anhang A.4).
In der Evaluierung wurden die genannten Klassifikatormodelle zunächst einzeln und auch als hybride Methoden gegenübergestellt. Die Auswertung wurde auf zwei Datensätzen durchgeführt und die Ergebnisse auf Grundlage der Bewertungskriterien F-Maß und BKF miteinander verglichen.
Es konnte zunächst in Abschnitt 4.2.1 gezeigt werden, dass die Kombination aus NN und SVM signifikant bessere Ergebnisse erzielt als unhybride NN, während die Kombination aus NN und RF keine Verbesserungen hervorbrachte. Die Vergleichswerte der unhybriden RF- und SVM-Modelle erreichten auf dem verwendeten Merkmalsraum nahezu vollständig signifikant schlechtere Ergebnisse, als die restlichen Klassifikatoren. Dieser Sachverhalt wird auch im Ausblick in Abschnitt 5.2.3 als weitere Untersuchung diskutiert.
Des Weiteren wurde in dieser Arbeit die Verwendung von Datenaugmentierungen in Abschnitt 4.2.2 untersucht. Die fünf Augmentierungen wurden in die Kategorien „Lautstärke“ und „Rauschen und Kompression“ eingeteilt und anschließend Modelle auf Daten ohne Augmentierungen, mit einer der beiden Augmentierungskategorien und mit beiden Augmentierungskategorien trainiert und ausgewertet. Es konnte gezeigt werden, dass für die meisten Modelle jegliche Art von Augmentierung signifikant besser als ein Training ohne Augmentierungen ist. Eine Ausnahme für die Aussage bildete unter anderem das NN-SVM-Modell auf dem Akkord-Datensatz, da hier die zwei Bewertungskriterien gegensätzliche Ergebnisse ergaben. In 4.2.2

wurde deshalb für das spezifische Modell eine kurze Analyse der Bewertungskriterien durchgeführt und der Widerspruch in dem Aufbau der Bewertungskriterien dargelegt. Darüber hinaus wurden unterschiedliche Ergebnisse auf den Datensätzen beim direkten Vergleich der Augmentierungskategorien festgestellt. Auf dem MIDI-Datensatz trugen vor allem Augmentierungen der Kategorie „Rauschen“ zu einer besseren Klassifikationsleistung bei, während auf dem Akkord-Datensatz für das NN-Modell Augmentierungen der Kategorie „Lautstärke“ signifikante Verbesserungen hervorbrachten. Auch zur weiteren Untersuchung von Augmentierungen wird in Abschnitt 5.2.3 ein kurzer Ausblick gegeben.
Der Generalisierungstest in Abschnitt 4.2.3 hat ergeben, dass die Modelle zwar auf den Testdaten des trainierten Datensatzes gute Ergebnisse erzielen konnten, auf dem anderen Datensatz aber mitunter nicht besser als statistisches Raten waren. Eine Weiterführung der Generalisierungsfähigkeit wurde mit dem kombinierten Datensatz in Abschnitt 4.2.4 angestellt, was zu einer guten Klassifikationsleistung auf beiden Datensätzen führte, allerdings die Generalisierungsfähigkeit auf einem dritten, unbekannten Datensatz noch offen lässt. Mögliche weitere Datensätze werden dafür im Ausblick in Abschnitt 5.2.1 vorgestellt.
Abschließend wurden in Abschnitt 4.2.6 auch die Klassifikationsleistungen der Modelle auf verrauschten Daten des MIDI-Datensatzes gegenübergestellt und ermittelt, dass im Vergleich vor allem lautere und Raumhall-veränderte Audiosignale zu größeren Klassifikationsfehlern führten.

Zusammenfassend lässt sich sagen, dass obwohl die Generalisierungsfähigkeit der Ansätze auf den eigenen Datensatz beschränkt ist, erreichen die verwendeten Modelle NN, NN-RF und NN-SVM dort eine gute Klassifikationsleistung (vergleiche Ergebnisse auf dem Akkord-Datensatz aus Tabelle A.8 mit Ergebnissen aus [96]). Eine weitere wichtige Erkenntnis ist, dass meistens das hybride NN-SVM-Modell zur besten Klassifikationsleistung der untersuchten Ansätze führt, allerdings auch auf Kosten einer im Vergleich hohen Trainingsdauer. Analog trainieren die Modelle auch mit zusätzlichen Augmentierungsdaten länger, erzielen aber signifikant bessere Ergebnisse. Zusätzlich konnte in dieser Arbeit die NN-Architektur aus [35] durch das Hinzufügen von Batch-Normalisierungsschichten signifikant verbessert werden.

Des Weiteren gibt es durch die Vielzahl an Kombinations- und Konfigurationsmöglichkeiten von hybriden Methoden und künstlichen neuronalen Netzen zahlreiche weitere Aspekte, welche in dieser Arbeit nicht berücksichtigt werden konnten und deshalb im folgenden Ausblick motiviert werden.

5.2 Ausblick

Im Folgenden werden Herangehensweisen und Methoden thematisiert, welche im Rahmen dieser Masterarbeit nicht durchgeführt wurden, aber als Erweiterungen für fortführende Untersuchungen von Relevanz sein können.

5.2.1 Teststrukturen und Datensätze

In dieser Arbeit wird die Evaluierung auf den vorgestellten MIDI- und Akkord-Datensätzen mit einer verschachtelten Kreuzvalidierung durchgeführt.

Die verschachtelte 5×4 Kreuzvalidierung generiert insgesamt 20 Iterationen pro Datensatz, lässt aber beispielsweise keinen direkten Vergleich der Trainingsabläufe eines NN zu, da sich bei jedem Durchlauf mindestens eine Untermenge von den jeweils anderen unterscheidet. Andere Validierungsstrategien wie zusätzliche statistische Wiederholungen können ebenso neue Erkenntnisse hervorbringen, wie die Wahl verschiedener Aufteilungsgrößen der Trainings-, Validierungs-, und Testmengen.

Neben Parameteränderungen der Evaluierungskonfiguration ist auch das Heranziehen von weiteren Datensätzen (siehe Tabelle 5.1) eine Möglichkeit, um die genannten Hypothesen und insbesondere die Generalisierungsfähigkeit der Ansätze weiter zu überprüfen.

Datensatz	Quelle	Beschreibung
MedleyDB	[10][11]	196 mehrspurige Stücke mit Annotierungen von Instrumenten und Melodien
NSynth	[23]	Insgesamt 305.979 einzelne Töne von 11 Instrumentenklassen, sowie unter anderem Annotierungen zu Note, Verzerrung und Hall
Magnatagatune	[49]	25.863 Stücke und 188 einzigartige Annotierungsarten wie Künstler, Genre und Stimmung
Million Song Dataset	[7]	1.000.000 Stücke und Metadaten wie Künstler, Länge, Lautstärke und vorberechnete Merkmale (ohne originale Audiospuren)

Tabelle 5.1: Weitere Datensätze zur Musik- und Instrumentenerkennung.

5.2.2 Weitere Schichten und Methoden künstlicher neuronaler Netze

Künstliche neuronale Netze bieten, auch aufgrund aktueller Forschungsergebnisse der letzten Jahre, zahlreiche Konfigurations- und Erweiterungsmöglichkeiten. Einige der fortführenden Methoden sind in den folgenden Abschnitten erläutert.

Implementierung einer PCEN-Schicht

Die Parameter der in dieser Arbeit eingesetzten Form der PCEN-Normalisierung sind, wie bereits in Abschnitt 2.3.3 dargelegt, ableitbar und somit ist es möglich die Normalisierung der Daten in den Optimierungsprozess des NN als Schicht einzubinden [98]. Ergebnisse der Implementierung einer PCEN-Schicht können in [108] eingesehen werden. Gegebenenfalls können so auch die teils variierenden Ergebnisse aus Abschnitt 4.2.7 verbessert werden.

Eindimensionale Faltungsschichten

Die in dieser Arbeit verwendete Architektur (siehe Anhang A.4) setzt vor allem zweidimensionale Faltungsschichten zur Datenverarbeitung ein. Eine Möglichkeit die Verarbeitung zu ändern ist eine Reduktion der Dimensionalität der Faltungskernel auf eine einzelne Dimension und eine entsprechende einfache Faltung über die Frequenz- oder Zeitdimension der Daten. Durch die geringere Parameteranzahl eindimensionaler Faltungsschichten kann so auch gegebenenfalls die Fenstergröße der Kernel größer gewählt werden.

Rekurrente Schichten

Weiterhin ist die Verwendung der in Abschnitt 3.2.2 vorgestellten rekurrenten Schichten möglich. Unter anderem Spracherkennungssysteme wie *Deep Speech* [36][3] setzen diese Form der Schichten erfolgreich bei der Erkennung von Audiodaten ein. Insbesondere bei einer größeren Fensterlänge von Audioausschnitten können so zeitliche Merkmale gegebenenfalls besser in den Klassifikationsvorgang eingebunden werden.

Attention

Als *Attention* (übersetzt: „Aufmerksamkeit“) werden Mechanismen in künstlichen neuronalen Netzen bezeichnet, welche Masken über die jeweiligen Eingaben berechnen und es so dem NN ermöglichen den Fokus auf bestimmte Regionen in den Daten zu legen.
Attention wird zum Beispiel in der Übersetzung von Texten [94] und in der Bild-Annotation erfolgreich eingesetzt (visualisiert im Anhang in Abbildung A.4). Auch die Erkennung von Audiodaten [47][88] und insbesondere Musikdaten [33] kann unter anderem mit dem Einsatz von Attention-Mechanismen realisiert werden.

Autoencoder

Ein Autoencoder ist eine besondere Form von NN, bei welchem in der Regel an Ein- und Ausgabe dieselben Daten vorliegen und dazwischen mindestens eine Schicht mit geringer Parameteranzahl vorliegt, sodass die eingegebenen Daten in einer komprimierten Form repräsentiert werden müssen (Aufbau visualisiert im Anhang in Abbildung A.5). Die so kompaktere Darstellung der Daten kann beispielsweise als Eingabe für andere Klassifikatoren verwendet werden.

Entrauschende Autoencoder

Eine spezielle Form von Autoencoder ist der entrauschende Autoencoder, bei welchem am Eingang die verrauschten/augmentierten Daten und am Ausgang die originalen Daten anliegen. Entrauschende Autoencoder können als Vorverarbeitung vor weiteren Klassifikatoren eingesetzt werden, um für diese ein Signal mit höherem Signal-Rausch-Verhältnis zu erhalten. Der Einsatz von entrauschenden Autoencodern ist deshalb vor allem in Umgebungen mit vielen Störfaktoren sinnvoll [4][26].

Generative Adversarial Networks

Die zwei bei einem *Generative Adversarial Network* (GAN) eingesetzten NN führen als „Generator“ und „Diskriminator“ ein Nullsummenspiel durch, indem ersteres lernt künstliche Daten nach einer bestimmten Verteilung zu generieren und letzteres lernt die künstlichen Beispiele von den originalen zu unterscheiden.
GAN können zum Beispiel bei unbalancierten Datensätzen eingesetzt werden, um die Anzahl der Trainingsbeispiele von unterrepräsentierten Klassen anzugleichen [58]. Außerdem können, wie bereits in Abschnitt 2.4.1 thematisiert, künstliche Trainingsbeispiele generiert werden, welche so augmentiert sind, dass sie eine maximale Fehlklassifikation des Klassifikators erzeugen [32].

5.2.3 Augmentierungs- und Merkmalsselektion

Augmentierungen

Die Ergebnisse des Evaluierungsabschnitts 4.2.6 legen nah, dass die Augmentierungen „Lauter“ und „Raumhall“ das Signal stärker verändern, als beispielsweise die Augmentierungen „Leiser“ oder „DRC“. Die Wahl von optimalen Augmentierungsarten und -parametern kann somit, zusammen mit GAN-generierten Augmentierungen, ebenfalls Bestandteil weiterer Untersuchungen sein.

Merkmale

Während in dieser Arbeit ausschließlich einsekündige Mel-Spektrogramme verwendet werden, so könnte eine Analyse zu verschiedenen Merkmalen, wie beispielsweise in [24], [95] oder [96], unter anderem für einen besseren Vergleich mit den Klassifikatoren RF und SVM, zusätzliche Erkenntnisse bringen. Auch eine Wahl verschiedener Extraktionslängen kann in weiteren Untersuchungen angestellt werden. Eine weitere aktuelle Forschung zur Instrumentenerkennung mit anderen Merkmalsräumen für ein NN ist in [51] mit einer Hilbert-Huang-Transformation der Audiodaten zu finden.

5.2.4 Hybride Kombinationsmöglichkeiten

Neben der Wahl von abweichenden Merkmalen und Modellkonfigurationen kann auch eine größere Untersuchung der hybriden Kombinationsmöglichkeiten angestellt werden. So ist neben dem sequentiellen, Transferlernen-basierten Training sowohl eine parallele Auswertung, als auch der Einsatz weiterer Klassifikatoren wie der Nächster-Nachbar- oder *Gradient-Boosting*-Klassifikator möglich. Auch eine durch Abschnitt 4.2.7 motivierte Untersuchung ob bestimmte (durch ein NN berechnete) Merkmale für die hybride Herangehensweise von Vorteil sind, kann ebenfalls in Betracht gezogen werden.

5.2.5 Weiterverwendung der Instrumentenerkennung

Abschließend können, wie bereits in der Motivation der Arbeit dargelegt, die Ergebnisse der Instrumentenerkennung weiterverwendet werden, um weitere Eigenschaften wie beispielsweise Genre [78] oder Tonhöhe [50] von Musikdaten zu klassifizieren.

A Anhang

A.1 Verwendete Bibliotheken

Bibliothek	Version	Quelle	Verwendung
ADT	19.11.2019	[37]	Pythonversion von Audio Degradation Toolbox; Erstellung von Audio-Augmentierungen
Keras	2.2.4	[18]	High-Level-API für die Erstellung tiefer neuronaler Netze in Tensorflow
Librosa	0.7.1	[61]	Vorverarbeitung von Audiodaten
NumPy	1.17.4	[68]	Transformation von Arrays
Python	3.6	[93]	Verwendete Programmiersprache
Tensorflow	2.1.0	[1]	Training von NN und Verarbeitung der Daten
Scikit-learn	0.21.3	[73]	Training von RF und SVM
SciPy	1.4.1	[97]	Durchführung statistischer Tests

A.2 Ablauf der Merkmalsberechnung

Listing A.1: Ablauf der Merkmalsberechnung und Normalisierung in Python.

```
1  SAMPLE_RATE = 16000
2  WINDOW_SIZE = 2048
3  HOP_LENGTH = 1024
4
5  def getMelSpectrogram(samples, power):
6    return librosa.feature.melspectrogram(y=samples, sr=SAMPLE_RATE, n_fft=
         WINDOW_SIZE, hop_length=HOP_LENGTH, win_length=None, window='hann', center=
         True, pad_mode='reflect', power=power)
7
8  def normalizePCEN(spec):
9    return librosa.pcen(spec * (2**31), sr=SAMPLE_RATE, hop_length=512, gain=0.98,
         bias=2, power=0.5, time_constant=0.4, eps=1e-06, b=None, max_size=1, ref=None
         , axis=-1, max_axis=None, zi=None, return_zf=False)
10
11 def getFeatures(samples):
12   # Calculate spectrograms
13   magnitude_spec = getMelSpectrogram(samples, 1.0)
14   power_spec = getMelSpectrogram(samples, 2.0)
15
16   # Normalize
17   normalized_magnitude_spec = normalizePCEN(magnitude_spec)
18   normalized_power_spec = normalizePCEN(power_spec)
19
20   # Transpose and stack
21   return np.stack([normalized_magnitude_spec.T, normalized_power_spec.T], axis=-1)
```

A.3 Greedy-Algorithmus zur repräsentativen Kreuzvalidierungsaufteilung

Listing A.2: Die Berechnung der Behälteraufteilung für die Kreuzvalidierung in Python.

```
def calculateCrossValidationSplit(items, num_classes=6, cv_amount=5):
  FileActivationCount = {}
  # Count every instrument activation per file
  for item in items:
    if (not item.path in FileActivationCount):
      FileActivationCount[item.path] = np.zeros((num_classes+1))
    for insID, onset in enumerate(item.labels):
      if (onset):
        FileActivationCount[item.path][insID] += 1
    # Count "no-activations" as N+1-th "instrument"
    if (not any(item.labels)):
      FileActivationCount[item.path][-1] += 1
  # Count total instrument activations
  TotalActivationCount = np.zeros((num_classes+1))
  for _, activations in FileActivationCount.items():
    TotalActivationCount += activations

  # Sort files into cross-validation containers
  cv_split_list = [ [] for _ in range(cv_amount) ]
  ActivationsRelative = np.zeros((cv_amount, num_classes+1))
  while (FileActivationCount):
    # Calculate relatively least-filled cv-container
    containerID = np.argmin(np.sum(ActivationsRelative, axis=1))
    # Try to fill the least-filled instrument of the chosen container
    for instrumentIndex in np.argsort(ActivationsRelative[containerID]):
      # Find sample-file with highest activation-count for that instrument
      sortValue = lambda i: FileActivationCount[i][instrumentIndex]
      maxFileName = max(FileActivationCount.keys(), key=sortValue);

      # If a sample-file with activation was found, update
      # and add sample to container
      # else continue with next least-filled instrument
      if (FileActivationCount[maxFileName][instrumentIndex] > 0):
        activations = FileActivationCount.pop(maxFileName)
        ActivationsRelative[containerID] += activations / TotalActivationCount[
            insID]
        cv_split_list[containerID].append(maxFileName)
        break
  return cv_split_list
```

A.4 Architektur des künstlichen neuronalen Netzes

#	Typ	Parameter	Einheiten	Ausgabegröße
1	Input			(16, 128, 2)
2	Convolutional2D	(3x3)	32	(16, 128, 32)
3	Convolutional2D	(3x3)	32	(16, 128, 32)
4	BatchNormalization			(16, 128, 32)
5	MaxPooling2D	(1x2)		(16, 64, 32)
7	Dropout	0.25		(16, 64, 32)
8	Convolutional2D	(3x3)	64	(16, 64, 64)
9	Convolutional2D	(3x3)	64	(16, 64, 64)
10	BatchNormalization			(16, 64, 64)
11	MaxPooling2D	(1x2)		(16, 32, 64)
12	Dropout	0.25		(16, 32, 64)
13	Convolutional2D	(3x3)	128	(16, 32, 128)
14	BatchNormalization			(16, 32, 128)
15	MaxPooling2D	(2x2)		(8, 16, 128)
16	Convolutional2D	(3x3)	128	(8, 16, 128)
17	BatchNormalization			(8, 16, 128)
18	MaxPooling2D	(2x2)		(4, 8, 128)
19	Dropout	0.25		(4, 8, 128)
20	Convolutional2D	(3x3)	256	(4, 8, 256)
21	BatchNormalization			(4, 8, 256)
22	MaxPooling2D	(2x2)		(2, 4, 256)
23	Convolutional2D	(3x3)	256	(2, 4, 256)
24	BatchNormalization			(2, 4, 256)
25	GlobalMaxPooling2D			(256)
26	Dropout	0.5		(256)
27	Dense	relu	1024	(1024)
26	Dropout	0.5		(1024)
28	Dense	sigmoid	6	(6)

Tabelle A.1: Architektur des künstlichen neuronalen Netzes bei einem einsekündigen Eingabefenster und sechs Instrumenten.

A.5 Audio Degradation Toolbox - Raumhalleffekt-Datei

Listing A.3: Die Erzeugung eines Raumhalleffekts mit Audio Degradation Toolbox [37].

```
[
  {
    "name": "impulse_response",
    "path": "./samples/IR_GreatHall.wav"
  },
  {
    "name": "noise",
    "color": "pink",
    "snr": 40
  },
  {
    "name": "normalize"
  }
]
```

A.6 Vergleiche von Hyperparametern

A.6.1 Batch-Normalisierung

Maß	Modell \ Ins.	Bratsche	Sitar	Violine	Trompete	Flöte	Piano	Ø
BKF	NN(mit BN)	**0,1196**	0,0711	0,1339	0,0573	0,1190	**0,3025**	**0,1339**
	NN(ohne BN)	0,1561	0,0736	0,1410	0,0577	0,1302	0,3540	0,1521
F-Maß	NN(mit BN)	**0,7962**	0,9126	0,7758	0,9133	0,8170	**0,7455**	**0,8267**
	NN(ohne BN)	0,7523	0,9062	0,7542	0,9212	0,8134	0,6949	0,8070

Tabelle A.2: Vergleich von NN-Modellen mit und ohne Batch-Normalisierung (BN) auf dem MIDI-Datensatz.

A.6.2 NN-SVM C-Parameter

Maß	Modell \ Ins.	Bratsche	Sitar	Violine	Trompete	Flöte	Piano	Ø
BKF	NN-SVM-C=0,1	0,1069	0,0563	0,1088	0,0367	0,0984	0,2764	0,1139
	NN-SVM-C=1,0	0,1102	0,0565	0,1120	0,0389	0,1051	0,2818	0,1174
F-Maß	NN-SVM-C=0,1	0,8104	0,9240	0,7880	0,9369	0,8412	0,7625	0,8439
	NN-SVM-C=1,0	0,8133	0,9240	0,7946	0,9377	0,8394	0,7641	0,8455

Tabelle A.3: Vergleich von NN-SVM-Modellen mit Hyperparameter C=0,1 und C=1,0 auf dem MIDI-Datensatz.

A.6.3 Melspektrogramme

Maß	Merkmale \ Ins.	Bratsche	Sitar	Violine	Trompete	Flöte	Piano	Ø
BKF	Magnitude	0,1344	0,0816	0,1304	0,0488	0,1287	0,3238	0,1413
	Leistung	0,1287	0,0663	0,1334	0,0459	0,1267	0,3164	0,1362
	Dual	0,1196	0,0711	0,1339	0,0573	0,1190	0,3025	0,1339
F-Maß	Magnitude	0,7849	0,8994	0,7739	0,9237	0,8092	0,7187	0,8183
	Leistung	0,7872	0,9168	0,7756	0,9293	0,8112	0,7352	0,8259
	Dual	0,7962	0,9126	0,7758	**0,9133**	0,8170	0,7455	0,8267

Tabelle A.4: Vergleich der Magnituden-, Leistungs- und dualen Melspektrogramme als Eingabe für ein NN auf dem MIDI-Datensatz.

A.7 Modellvergleiche auf einzelnen Instrumenten

A.7.1 MIDI-Datensatz

Ins.	Model	RF	SVM	NN	NN-RF	NN-SVM
Bratsche	RF	0,0120	**-0,5355**	**-0,7842**	**-0,7908**	**-0,7984**
	SVM	**+0,5355**	0,5475	**-0,2487**	**-0,2553**	**-0,2630**
	NN	**+0,7842**	**+0,2487**	0,7962	-0,0066	-0,0142
	NN-RF	**+0,7908**	**+0,2553**	+0,0066	0,8028	-0,0076
	NN-SVM	**+0,7984**	**+0,2630**	+0,0142	+0,0076	0,8104
Sitar	RF	0,6406	**-0,1137**	**-0,2721**	**-0,2845**	**-0,2835**
	SVM	**+0,1137**	0,7543	**-0,1583**	**-0,1708**	**-0,1697**
	NN	**+0,2721**	**+0,1583**	0,9126	-0,0124	-0,0114
	NN-RF	**+0,2845**	**+0,1708**	+0,0124	0,9251	+0,0010
	NN-SVM	**+0,2835**	**+0,1697**	+0,0114	-0,0010	0,9240
Violine	RF	0,0067	**-0,5293**	**-0,7691**	**-0,7854**	**-0,7813**
	SVM	**+0,5293**	0,5360	**-0,2398**	**-0,2561**	**-0,2520**
	NN	**+0,7691**	**+0,2398**	0,7758	-0,0163	-0,0122
	NN-RF	**+0,7854**	**+0,2561**	+0,0163	0,7921	+0,0041
	NN-SVM	**+0,7813**	**+0,2520**	+0,0122	-0,0041	0,7880
Trompete	RF	0,0794	**-0,5175**	**-0,8340**	**-0,8516**	**-0,8575**
	SVM	**+0,5175**	0,5969	**-0,3165**	**-0,3341**	**-0,3400**
	NN	**+0,8340**	**+0,3165**	0,9133	**-0,0176**	**-0,0235**
	NN-RF	**+0,8516**	**+0,3341**	**+0,0176**	0,9309	-0,0059
	NN-SVM	**+0,8575**	**+0,3400**	**+0,0235**	+0,0059	0,9369
Flöte	RF	0,0116	**-0,5237**	**-0,8054**	**-0,7972**	**-0,8296**
	SVM	**+0,5237**	0,5353	**-0,2817**	**-0,2735**	**-0,3059**
	NN	**+0,8054**	**+0,2817**	0,8170	+0,0082	-0,0243
	NN-RF	**+0,7972**	**+0,2735**	-0,0082	0,8088	**-0,0325**
	NN-SVM	**+0,8296**	**+0,3059**	+0,0243	**+0,0325**	0,8412
Piano	RF	0,7216	**+0,0946**	**-0,0240**	**-0,0503**	**-0,0410**
	SVM	**-0,0946**	0,6270	**-0,1186**	**-0,1449**	**-0,1356**
	NN	**+0,0240**	**+0,1186**	0,7455	**-0,0264**	-0,0170
	NN-RF	**+0,0503**	**+0,1449**	**+0,0264**	0,7719	+0,0094
	NN-SVM	**+0,0410**	**+0,1356**	+0,0170	-0,0094	0,7625
Ø	RF	0,2453	**-0,3542**	**-0,5814**	**-0,5933**	**-0,5985**
	SVM	**+0,3542**	0,5995	**-0,2273**	**-0,2391**	**-0,2444**
	NN	**+0,5814**	**+0,2273**	0,8267	-0,0119	**-0,0171**
	NN-RF	**+0,5933**	**+0,2391**	+0,0119	0,8386	-0,0053
	NN-SVM	**+0,5985**	**+0,2444**	**+0,0171**	+0,0053	0,8439

Tabelle A.5: Vergleich der Modelle auf dem MIDI-Datensatz nach dem F-Maß.

Ins.	Model	RF	SVM	NN	NN-RF	NN-SVM
Bratsche	RF	0,4970	**+0,2615**	**+0,3774**	**+0,3504**	**+0,3901**
	SVM	**-0,2615**	0,2355	**+0,1158**	**+0,0889**	**+0,1285**
	NN	**-0,3774**	**-0,1158**	0,1196	**-0,0270**	+0,0127
	NN-RF	**-0,3504**	**-0,0889**	**+0,0270**	0,1466	**+0,0397**
	NN-SVM	**-0,3901**	**-0,1285**	-0,0127	**-0,0397**	0,1069
Sitar	RF	0,2636	**+0,0893**	**+0,1925**	**+0,1998**	**+0,2073**
	SVM	**-0,0893**	0,1743	**+0,1032**	**+0,1105**	**+0,1180**
	NN	**-0,1925**	**-0,1032**	0,0711	+0,0073	**+0,0148**
	NN-RF	**-0,1998**	**-0,1105**	-0,0073	0,0638	+0,0075
	NN-SVM	**-0,2073**	**-0,1180**	**-0,0148**	-0,0075	0,0563
Violine	RF	0,4983	**+0,2536**	**+0,3644**	**+0,3498**	**+0,3895**
	SVM	**-0,2536**	0,2448	**+0,1108**	**+0,0962**	**+0,1359**
	NN	**-0,3644**	**-0,1108**	0,1339	-0,0146	**+0,0251**
	NN-RF	**-0,3498**	**-0,0962**	+0,0146	0,1486	**+0,0398**
	NN-SVM	**-0,3895**	**-0,1359**	**-0,0251**	**-0,0398**	0,1088
Trompete	RF	0,4793	**+0,2729**	**+0,4220**	**+0,4200**	**+0,4427**
	SVM	**-0,2729**	0,2064	**+0,1491**	**+0,1471**	**+0,1698**
	NN	**-0,4220**	**-0,1491**	0,0573	-0,0020	**+0,0207**
	NN-RF	**-0,4200**	**-0,1471**	+0,0020	0,0593	**+0,0227**
	NN-SVM	**-0,4427**	**-0,1698**	**-0,0207**	**-0,0227**	0,0367
Flöte	RF	0,4972	**+0,2566**	**+0,3782**	**+0,3444**	**+0,3988**
	SVM	**-0,2566**	0,2406	**+0,1216**	**+0,0878**	**+0,1422**
	NN	**-0,3782**	**-0,1216**	0,1190	**-0,0339**	**+0,0206**
	NN-RF	**-0,3444**	**-0,0878**	**+0,0339**	0,1528	**+0,0544**
	NN-SVM	**-0,3988**	**-0,1422**	**-0,0206**	**-0,0544**	0,0984
Piano	RF	0,4488	**+0,0264**	**+0,1463**	**+0,1714**	**+0,1724**
	SVM	**-0,0264**	0,4223	**+0,1198**	**+0,1449**	**+0,1459**
	NN	**-0,1463**	**-0,1198**	0,3025	**+0,0251**	**+0,0261**
	NN-RF	**-0,1714**	**-0,1449**	**-0,0251**	0,2774	+0,0010
	NN-SVM	**-0,1724**	**-0,1459**	**-0,0261**	-0,0010	0,2764
Ø	RF	0,4474	**+0,1934**	**+0,3135**	**+0,3060**	**+0,3335**
	SVM	**-0,1934**	0,2540	**+0,1201**	**+0,1126**	**+0,1401**
	NN	**-0,3135**	**-0,1201**	0,1339	-0,0075	**+0,0200**
	NN-RF	**-0,3060**	**-0,1126**	+0,0075	0,1414	**+0,0275**
	NN-SVM	**-0,3335**	**-0,1401**	**-0,0200**	**-0,0275**	0,1139

Tabelle A.6: Vergleich der Modelle auf dem MIDI-Datensatz nach dem balancierten Klassifizierungsfehler.

A.7.2 Akkord-Datensatz

Ins.	Model	RF	SVM	NN	NN-RF	NN-SVM
Bratsche	RF	0,0232	**-0,3970**	**-0,5066**	**-0,4367**	**-0,5389**
	SVM	**+0,3970**	0,4201	**-0,1096**	**-0,0397**	**-0,1419**
	NN	**+0,5066**	**+0,1096**	0,5297	**+0,0698**	**-0,0324**
	NN-RF	**+0,4367**	**+0,0397**	**-0,0698**	0,4599	**-0,1022**
	NN-SVM	**+0,5389**	**+0,1419**	**+0,0324**	**+0,1022**	0,5621
Sitar	RF	0,0104	**-0,3147**	**-0,8402**	**-0,8397**	**-0,8746**
	SVM	**+0,3147**	0,3252	**-0,5254**	**-0,5250**	**-0,5598**
	NN	**+0,8402**	**+0,5254**	0,8506	+0,0004	**-0,0344**
	NN-RF	**+0,8397**	**+0,5250**	-0,0004	0,8502	**-0,0348**
	NN-SVM	**+0,8746**	**+0,5598**	**+0,0344**	**+0,0348**	0,8850
Violine	RF	0,0000	**-0,3486**	**-0,4057**	**-0,4013**	**-0,4996**
	SVM	**+0,3486**	0,3486	**-0,0570**	**-0,0527**	**-0,1510**
	NN	**+0,4057**	**+0,0570**	0,4057	+0,0044	**-0,0940**
	NN-RF	**+0,4013**	**+0,0527**	-0,0044	0,4013	**-0,0983**
	NN-SVM	**+0,4996**	**+0,1510**	**+0,0940**	**+0,0983**	0,4996
Trompete	RF	0,0404	**-0,2996**	**-0,5065**	**-0,5048**	**-0,5437**
	SVM	**+0,2996**	0,3400	**-0,2068**	**-0,2052**	**-0,2441**
	NN	**+0,5065**	**+0,2068**	0,5468	+0,0017	**-0,0372**
	NN-RF	**+0,5048**	**+0,2052**	-0,0017	0,5452	**-0,0389**
	NN-SVM	**+0,5437**	**+0,2441**	**+0,0372**	**+0,0389**	0,5841
Flöte	RF	0,0030	**-0,3604**	**-0,3357**	**-0,2377**	**-0,4366**
	SVM	**+0,3604**	0,3634	+0,0247	**+0,1227**	**-0,0762**
	NN	**+0,3357**	-0,0247	0,3387	**+0,0981**	**-0,1009**
	NN-RF	**+0,2377**	**-0,1227**	**-0,0981**	0,2407	**-0,1989**
	NN-SVM	**+0,4366**	**+0,0762**	**+0,1009**	**+0,1989**	0,4396
Piano	RF	0,3730	**-0,0663**	**-0,4239**	**-0,4374**	**-0,4569**
	SVM	**+0,0663**	0,4393	**-0,3576**	**-0,3711**	**-0,3906**
	NN	**+0,4239**	**+0,3576**	0,7969	-0,0135	**-0,0330**
	NN-RF	**+0,4374**	**+0,3711**	+0,0135	0,8104	**-0,0195**
	NN-SVM	**+0,4569**	**+0,3906**	**+0,0330**	**+0,0195**	0,8299
Ø	RF	0,0750	**-0,2978**	**-0,5031**	**-0,4763**	**-0,5584**
	SVM	**+0,2978**	0,3728	**-0,2053**	**-0,1785**	**-0,2606**
	NN	**+0,5031**	**+0,2053**	0,5781	**+0,0268**	**-0,0553**
	NN-RF	**+0,4763**	**+0,1785**	**-0,0268**	0,5513	**-0,0821**
	NN-SVM	**+0,5584**	**+0,2606**	**+0,0553**	**+0,0821**	0,6334

Tabelle A.7: Vergleich der Modelle auf dem Akkord-Datensatz nach dem F-Maß.

Ins.	Model	RF	SVM	NN	NN-RF	NN-SVM
Bratsche	RF	0,4941	**+0,1898**	**+0,2242**	**+0,1563**	**+0,2981**
	SVM	**-0,1898**	0,3043	**+0,0344**	**-0,0335**	**+0,1082**
	NN	**-0,2242**	**-0,0344**	0,2699	**-0,0679**	**+0,0739**
	NN-RF	**-0,1563**	**+0,0335**	**+0,0679**	0,3378	**+0,1417**
	NN-SVM	**-0,2981**	**-0,1082**	**-0,0739**	**-0,1417**	0,1961
Sitar	RF	0,4974	**+0,2925**	**+0,4132**	**+0,3699**	**+0,4343**
	SVM	**-0,2925**	0,2049	**+0,1207**	**+0,0774**	**+0,1419**
	NN	**-0,4132**	**-0,1207**	0,0842	**-0,0432**	+0,0212
	NN-RF	**-0,3699**	**-0,0774**	**+0,0432**	0,1274	**+0,0644**
	NN-SVM	**-0,4343**	**-0,1419**	-0,0212	**-0,0644**	0,0630
Violine	RF	0,5000	**+0,1557**	**+0,1461**	**+0,1334**	**+0,2742**
	SVM	**-0,1557**	0,3443	-0,0096	**-0,0223**	**+0,1186**
	NN	**-0,1461**	+0,0096	0,3539	-0,0127	**+0,1282**
	NN-RF	**-0,1334**	**+0,0223**	+0,0127	0,3666	**+0,1408**
	NN-SVM	**-0,2742**	**-0,1186**	**-0,1282**	**-0,1408**	0,2258
Trompete	RF	0,4897	**+0,1279**	**+0,2199**	**+0,1932**	**+0,2824**
	SVM	**-0,1279**	0,3618	**+0,0920**	**+0,0653**	**+0,1545**
	NN	**-0,2199**	**-0,0920**	0,2698	**-0,0267**	**+0,0625**
	NN-RF	**-0,1932**	**-0,0653**	**+0,0267**	0,2964	**+0,0891**
	NN-SVM	**-0,2824**	**-0,1545**	**-0,0625**	**-0,0891**	0,2073
Flöte	RF	0,4994	**+0,1685**	**+0,1182**	**+0,0677**	**+0,2331**
	SVM	**-0,1685**	0,3309	**-0,0503**	**-0,1008**	**+0,0646**
	NN	**-0,1182**	**+0,0503**	0,3812	**-0,0505**	**+0,1149**
	NN-RF	**-0,0677**	**+0,1008**	**+0,0505**	0,4316	**+0,1654**
	NN-SVM	**-0,2331**	**-0,0646**	**-0,1149**	**-0,1654**	0,2662
Piano	RF	0,3842	**+0,0840**	**+0,2850**	**+0,2688**	**+0,3115**
	SVM	**-0,0840**	0,3002	**+0,2010**	**+0,1849**	**+0,2275**
	NN	**-0,2850**	**-0,2010**	0,0992	-0,0162	**+0,0265**
	NN-RF	**-0,2688**	**-0,1849**	+0,0162	0,1154	**+0,0427**
	NN-SVM	**-0,3115**	**-0,2275**	**-0,0265**	**-0,0427**	0,0727
Ø	RF	0,4775	**+0,1697**	**+0,2344**	**+0,1982**	**+0,3056**
	SVM	**-0,1697**	0,3077	**+0,0647**	**+0,0285**	**+0,1359**
	NN	**-0,2344**	**-0,0647**	0,2430	**-0,0362**	**+0,0712**
	NN-RF	**-0,1982**	**-0,0285**	**+0,0362**	0,2792	**+0,1074**
	NN-SVM	**-0,3056**	**-0,1359**	**-0,0712**	**-0,1074**	0,1718

Tabelle A.8: Vergleich der Modelle auf dem Akkord-Datensatz nach dem balancierten Klassifizierungsfehler.

A.8 P-Werte beim Vergleich der Klassifikatormodelle

A.8.1 MIDI-Datensatz

Ins.	Model	RF	SVM	NN	NN-RF	NN-SVM
Bratsche	RF	1,0000	6,7336e-24	4,7099e-25	6,1391e-24	1,0569e-29
	SVM	6,7336e-24	1,0000	1,5784e-19	4,2078e-18	1,6179e-23
	NN	4,7099e-25	1,5784e-19	1,0000	0,6915	0,2788
	NN-RF	6,1391e-24	4,2078e-18	0,6915	1,0000	0,5950
	NN-SVM	1,0569e-29	1,6179e-23	0,2788	0,5950	1,0000
Sitar	RF	1,0000	7,5298e-12	8,7674e-25	8,5364e-25	2,6997e-23
	SVM	7,5298e-12	1,0000	6,5337e-16	1,6307e-16	8,3209e-16
	NN	8,7674e-25	6,5337e-16	1,0000	0,0567	0,0576
	NN-RF	8,5364e-25	1,6307e-16	0,0567	1,0000	0,8533
	NN-SVM	2,6997e-23	8,3209e-16	0,0576	0,8533	1,0000
Violine	RF	1,0000	1,3769e-27	1,5670e-25	5,3593e-27	1,6863e-30
	SVM	1,3769e-27	1,0000	1,4157e-19	8,2684e-23	8,9327e-28
	NN	1,5670e-25	1,4157e-19	1,0000	0,2276	0,3070
	NN-RF	5,3593e-27	8,2684e-23	0,2276	1,0000	0,7047
	NN-SVM	1,6863e-30	8,9327e-28	0,3070	0,7047	1,0000
Trompete	RF	1,0000	7,2278e-34	1,6217e-44	1,2147e-35	1,2049e-37
	SVM	7,2278e-34	1,0000	3,2881e-26	3,8131e-22	4,1141e-23
	NN	1,6217e-44	3,2881e-26	1,0000	0,0278	0,0051
	NN-RF	1,2147e-35	3,8131e-22	0,0278	1,0000	0,2181
	NN-SVM	1,2049e-37	4,1141e-23	0,0051	0,2181	1,0000
Flöte	RF	1,0000	1,6295e-29	8,2939e-27	1,4246e-34	1,6861e-37
	SVM	1,6295e-29	1,0000	2,8043e-21	6,7131e-28	4,6337e-30
	NN	8,2939e-27	2,8043e-21	1,0000	0,5098	0,0518
	NN-RF	1,4246e-34	6,7131e-28	0,5098	1,0000	0,0005
	NN-SVM	1,6861e-37	4,6337e-30	0,0518	0,0005	1,0000
Piano	RF	1,0000	3,2424e-15	0,0180	2,9688e-10	6,9670e-07
	SVM	3,2424e-15	1,0000	5,4330e-13	5,6335e-21	2,3255e-19
	NN	0,0180	5,4330e-13	1,0000	0,0114	0,1074
	NN-RF	2,9688e-10	5,6335e-21	0,0114	1,0000	0,1953
	NN-SVM	6,9670e-07	2,3255e-19	0,1074	0,1953	1,0000
Ø	RF	1,0000	8,1269e-49	2,0778e-34	1,4301e-46	1,8635e-54
	SVM	8,1269e-49	1,0000	1,2238e-25	3,2240e-36	2,0638e-41
	NN	2,0778e-34	1,2238e-25	1,0000	0,0583	0,0055
	NN-RF	1,4301e-46	3,2240e-36	0,0583	1,0000	0,2172
	NN-SVM	1,8635e-54	2,0638e-41	0,0055	0,2172	1,0000

Tabelle A.9: P-Werte beim Vergleich der Modelle auf dem MIDI-Datensatz nach dem F-Maß.

Ins.	Model	RF	SVM	NN	NN-RF	NN-SVM
Bratsche	RF	1,0000	1,9927e-19	4,1356e-20	4,4084e-19	2,8195e-22
	SVM	1,9927e-19	1,0000	5,3549e-12	9,0940e-09	2,2207e-15
	NN	4,1356e-20	5,3549e-12	1,0000	0,0489	0,2831
	NN-RF	4,4084e-19	9,0940e-09	0,0489	1,0000	0,0022
	NN-SVM	2,8195e-22	2,2207e-15	0,2831	0,0022	1,0000
Sitar	RF	1,0000	7,8921e-13	8,5221e-30	3,7936e-30	5,8377e-28
	SVM	7,8921e-13	1,0000	2,1943e-14	1,0507e-14	1,4422e-14
	NN	8,5221e-30	2,1943e-14	1,0000	0,1502	0,0023
	NN-RF	3,7936e-30	1,0507e-14	0,1502	1,0000	0,0592
	NN-SVM	5,8377e-28	1,4422e-14	0,0023	0,0592	1,0000
Violine	RF	1,0000	2,0002e-25	3,9761e-22	8,7930e-22	6,5518e-25
	SVM	2,0002e-25	1,0000	2,5332e-14	7,5668e-13	1,0297e-20
	NN	3,9761e-22	2,5332e-14	1,0000	0,1419	0,0065
	NN-RF	8,7930e-22	7,5668e-13	0,1419	1,0000	5,5781e-05
	NN-SVM	6,5518e-25	1,0297e-20	0,0065	5,5781e-05	1,0000
Trompete	RF	1,0000	5,2378e-23	7,1337e-27	1,5958e-49	1,0624e-52
	SVM	5,2378e-23	1,0000	5,1864e-19	7,7236e-18	1,4017e-17
	NN	7,1337e-27	5,1864e-19	1,0000	0,7766	0,0050
	NN-RF	1,5958e-49	7,7236e-18	0,7766	1,0000	1,2961e-08
	NN-SVM	1,0624e-52	1,4017e-17	0,0050	1,2961e-08	1,0000
Flöte	RF	1,0000	2,9022e-23	7,7874e-22	5,3077e-26	8,5712e-30
	SVM	2,9022e-23	1,0000	8,1822e-15	1,5345e-16	3,1494e-24
	NN	7,7874e-22	8,1822e-15	1,0000	0,0005	0,0198
	NN-RF	5,3077e-26	1,5345e-16	0,0005	1,0000	1,2440e-11
	NN-SVM	8,5712e-30	3,1494e-24	0,0198	1,2440e-11	1,0000
Piano	RF	1,0000	1,8905e-05	5,3071e-18	5,7712e-27	3,6718e-26
	SVM	1,8905e-05	1,0000	3,6546e-16	2,9118e-24	8,6413e-24
	NN	5,3071e-18	3,6546e-16	1,0000	0,0037	0,0030
	NN-RF	5,7712e-27	2,9118e-24	0,0037	1,0000	0,8787
	NN-SVM	3,6718e-26	8,6413e-24	0,0030	0,8787	1,0000
Ø	RF	1,0000	2,0613e-27	2,1721e-29	3,9975e-32	1,2850e-37
	SVM	2,0613e-27	1,0000	3,0463e-26	2,3644e-27	3,2967e-31
	NN	2,1721e-29	3,0463e-26	1,0000	0,0908	1,7778e-05
	NN-RF	3,9975e-32	2,3644e-27	0,0908	1,0000	2,8365e-09
	NN-SVM	1,2850e-37	3,2967e-31	1,7778e-05	2,8365e-09	1,0000

Tabelle A.10: P-Werte beim Vergleich der Modelle auf dem MIDI-Datensatz nach dem balancierten Klassifizierungsfehler.

A.8.2 Akkord-Datensatz

Ins.	Model	RF	SVM	NN	NN-RF	NN-SVM
Bratsche	RF	1,0000	3,6635e-46	5,4886e-23	8,7713e-21	1,1588e-35
	SVM	3,6635e-46	1,0000	1,5271e-09	0,0030	3,3341e-20
	NN	5,4886e-23	1,5271e-09	1,0000	7,3992e-05	0,0121
	NN-RF	8,7713e-21	0,0030	7,3992e-05	1,0000	1,1588e-08
	NN-SVM	1,1588e-35	3,3341e-20	0,0121	1,1588e-08	1,0000
Sitar	RF	1,0000	5,2976e-43	3,1078e-24	1,2095e-29	3,0099e-32
	SVM	5,2976e-43	1,0000	4,2645e-20	4,0160e-25	8,8412e-28
	NN	3,1078e-24	4,2645e-20	1,0000	0,9802	0,0466
	NN-RF	1,2095e-29	4,0160e-25	0,9802	1,0000	0,0095
	NN-SVM	3,0099e-32	8,8412e-28	0,0466	0,0095	1,0000
Violine	RF	1,0000	1,8343e-32	9,4793e-14	8,8903e-20	9,2385e-27
	SVM	1,8343e-32	1,0000	0,0161	4,7198e-05	4,2379e-19
	NN	9,4793e-14	0,0161	1,0000	0,8556	0,0004
	NN-RF	8,8903e-20	4,7198e-05	0,8556	1,0000	1,7230e-09
	NN-SVM	9,2385e-27	4,2379e-19	0,0004	1,7230e-09	1,0000
Trompete	RF	1,0000	1,5527e-36	7,5644e-24	2,3074e-31	2,1086e-41
	SVM	1,5527e-36	1,0000	2,4523e-14	4,4995e-19	1,4031e-27
	NN	7,5644e-24	2,4523e-14	1,0000	0,9055	0,0069
	NN-RF	2,3074e-31	4,4995e-19	0,9055	1,0000	0,0003
	NN-SVM	2,1086e-41	1,4031e-27	0,0069	0,0003	1,0000
Flöte	RF	1,0000	1,0629e-31	1,3248e-12	5,8554e-13	8,6578e-27
	SVM	1,0629e-31	1,0000	0,2553	3,0339e-08	7,5130e-13
	NN	1,3248e-12	0,2553	1,0000	0,0004	0,0001
	NN-RF	5,8554e-13	3,0339e-08	0,0004	1,0000	1,1509e-12
	NN-SVM	8,6578e-27	7,5130e-13	0,0001	1,1509e-12	1,0000
Piano	RF	1,0000	9,0796e-06	2,2366e-26	4,2505e-22	7,9874e-22
	SVM	9,0796e-06	1,0000	1,1839e-22	1,2180e-34	6,7601e-40
	NN	2,2366e-26	1,1839e-22	1,0000	0,1110	0,0003
	NN-RF	4,2505e-22	1,2180e-34	0,1110	1,0000	0,0002
	NN-SVM	7,9874e-22	6,7601e-40	0,0003	0,0002	1,0000
∅	RF	1,0000	3,0769e-36	1,0603e-25	1,0480e-32	1,9999e-46
	SVM	3,0769e-36	1,0000	3,3142e-16	5,7274e-19	2,4627e-27
	NN	1,0603e-25	3,3142e-16	1,0000	0,0103	1,6725e-06
	NN-RF	1,0480e-32	5,7274e-19	0,0103	1,0000	2,9325e-14
	NN-SVM	1,9999e-46	2,4627e-27	1,6725e-06	2,9325e-14	1,0000

Tabelle A.11: P-Werte beim Vergleich der Modelle auf dem Akkord-Datensatz nach dem F-Maß.

Ins.	Model	RF	SVM	NN	NN-RF	NN-SVM
Bratsche	RF	1,0000	1,0224e-24	8,4845e-15	1,6922e-16	2,0698e-27
	SVM	1,0224e-24	1,0000	0,0048	3,4769e-05	1,4013e-23
	NN	8,4845e-15	0,0048	1,0000	4,2223e-06	8,1835e-07
	NN-RF	1,6922e-16	3,4769e-05	4,2223e-06	1,0000	3,0251e-19
	NN-SVM	2,0698e-27	1,4013e-23	8,1835e-07	3,0251e-19	1,0000
Sitar	RF	1,0000	8,6995e-24	7,0325e-21	3,2405e-22	7,2821e-27
	SVM	8,6995e-24	1,0000	2,5040e-12	2,1685e-10	7,4946e-22
	NN	7,0325e-21	2,5040e-12	1,0000	0,0007	0,0541
	NN-RF	3,2405e-22	2,1685e-10	0,0007	1,0000	1,5703e-08
	NN-SVM	7,2821e-27	7,4946e-22	0,0541	1,5703e-08	1,0000
Violine	RF	1,0000	6,9236e-24	5,4048e-11	4,7118e-17	1,8594e-24
	SVM	6,9236e-24	1,0000	0,4077	0,0002	8,0542e-23
	NN	5,4048e-11	0,4077	1,0000	0,3026	1,1023e-10
	NN-RF	4,7118e-17	0,0002	0,3026	1,0000	1,6337e-23
	NN-SVM	1,8594e-24	8,0542e-23	1,1023e-10	1,6337e-23	1,0000
Trompete	RF	1,0000	1,6902e-18	3,2730e-14	1,6267e-22	1,4674e-25
	SVM	1,6902e-18	1,0000	5,6968e-08	4,8347e-13	3,9925e-25
	NN	3,2730e-14	5,6968e-08	1,0000	0,0355	2,3359e-05
	NN-RF	1,6267e-22	4,8347e-13	0,0355	1,0000	4,6255e-17
	NN-SVM	1,4674e-25	3,9925e-25	2,3359e-05	4,6255e-17	1,0000
Flöte	RF	1,0000	1,8566e-22	3,9058e-09	4,5288e-11	1,5527e-22
	SVM	1,8566e-22	1,0000	0,0004	2,7515e-17	3,5139e-14
	NN	3,9058e-09	0,0004	1,0000	0,0005	2,0129e-09
	NN-RF	4,5288e-11	2,7515e-17	0,0005	1,0000	1,3608e-24
	NN-SVM	1,5527e-22	3,5139e-14	2,0129e-09	1,3608e-24	1,0000
Piano	RF	1,0000	1,1085e-15	2,8745e-26	1,8618e-33	1,8085e-33
	SVM	1,1085e-15	1,0000	3,7077e-18	8,1162e-31	7,1979e-39
	NN	2,8745e-26	3,7077e-18	1,0000	0,0527	0,0022
	NN-RF	1,8618e-33	8,1162e-31	0,0527	1,0000	4,6733e-12
	NN-SVM	1,8085e-33	7,1979e-39	0,0022	4,6733e-12	1,0000
Ø	RF	1,0000	7,4416e-38	2,1885e-24	4,3252e-28	1,1175e-41
	SVM	7,4416e-38	1,0000	4,9681e-14	3,3276e-10	6,3212e-38
	NN	2,1885e-24	4,9681e-14	1,0000	1,1214e-08	2,7174e-15
	NN-RF	4,3252e-28	3,3276e-10	1,1214e-08	1,0000	7,3659e-27
	NN-SVM	1,1175e-41	6,3212e-38	2,7174e-15	7,3659e-27	1,0000

Tabelle A.12: P-Werte beim Vergleich der Modelle auf dem Akkord-Datensatz nach dem balancierten Klassifizierungsfehler.

A.9 Tabellarische Vergleiche der Aumentierungskategorien

DS	Maß	Aug.-Kat.	Keine	Lautstärke	Rauschen	Alle
Akkord-Datensatz	BKF	Keine	0,4775	-0,0020	+0,0016	+0,0004
		Lautstärke	+0,0020	0,4795	**+0,0036**	**+0,0024**
		Rauschen	-0,0016	**-0,0036**	0,4759	-0,0012
		Alle	-0,0004	**-0,0024**	+0,0012	0,4771
	F-Maß	Keine	0,0750	+0,0056	-0,0049	-0,0025
		Lautstärke	-0,0056	0,0694	**-0,0104**	**-0,0081**
		Rauschen	+0,0049	**+0,0104**	0,0799	+0,0023
		Alle	+0,0025	**+0,0081**	-0,0023	0,0775
MIDI-Datensatz	BKF	Keine	0,4474	**-0,0041**	**+0,0118**	**+0,0068**
		Lautstärke	**+0,0041**	0,4515	**+0,0159**	**+0,0109**
		Rauschen	**-0,0118**	**-0,0159**	0,4356	**-0,0050**
		Alle	**-0,0068**	**-0,0109**	**+0,0050**	0,4406
	F-Maß	Keine	0,2453	**+0,0103**	**-0,0361**	**-0,0224**
		Lautstärke	**-0,0103**	0,2350	**-0,0464**	**-0,0327**
		Rauschen	**+0,0361**	**+0,0464**	0,2814	**+0,0137**
		Alle	**+0,0224**	**+0,0327**	**-0,0137**	0,2677

Tabelle A.13: Vergleich der Augmentierungskategorien für das Modell RF.

DS	Maß	Aug.-Kat.	Keine	Lautstärke	Rauschen	Alle
Akkord-Datensatz	BKF	Keine	0,3077	**+0,0270**	**+0,0284**	**+0,0366**
		Lautstärke	**-0,0270**	0,2807	+0,0014	**+0,0096**
		Rauschen	**-0,0284**	-0,0014	0,2793	**+0,0082**
		Alle	**-0,0366**	**-0,0096**	**-0,0082**	0,2711
	F-Maß	Keine	0,3728	**-0,0421**	**-0,0525**	**-0,0722**
		Lautstärke	**+0,0421**	0,4149	**-0,0104**	**-0,0300**
		Rauschen	**+0,0525**	**+0,0104**	0,4252	**-0,0197**
		Alle	**+0,0722**	**+0,0300**	**+0,0197**	0,4449
MIDI-Datensatz	BKF	Keine	0,2540	**+0,0154**	**+0,0315**	**+0,0350**
		Lautstärke	**-0,0154**	0,2386	**+0,0161**	**+0,0196**
		Rauschen	**-0,0315**	**-0,0161**	0,2225	+0,0035
		Alle	**-0,0350**	**-0,0196**	-0,0035	0,2190
	F-Maß	Keine	0,5995	**-0,0201**	**-0,0555**	**-0,0590**
		Lautstärke	**+0,0201**	0,6195	**-0,0355**	**-0,0389**
		Rauschen	**+0,0555**	**+0,0355**	0,6550	-0,0034
		Alle	**+0,0590**	**+0,0389**	+0,0034	0,6584

Tabelle A.14: Vergleich der Augmentierungskategorien für das Modell SVM.

DS	Maß	Aug.-Kat.	Keine	Lautstärke	Rauschen	Alle
Akkord-Datensatz	BKF	Keine	0,2430	**+0,0201**	+0,0078	**+0,0168**
		Lautstärke	**-0,0201**	0,2229	**-0,0123**	-0,0034
		Rauschen	-0,0078	**+0,0123**	0,2352	**+0,0089**
		Alle	**-0,0168**	+0,0034	**-0,0089**	0,2263
	F-Maß	Keine	0,5781	**-0,0372**	**-0,0206**	**-0,0404**
		Lautstärke	**+0,0372**	0,6153	+0,0166	-0,0032
		Rauschen	**+0,0206**	-0,0166	0,5987	**-0,0198**
		Alle	**+0,0404**	+0,0032	**+0,0198**	0,6185
MIDI-Datensatz	BKF	Keine	0,1339	**+0,0247**	**+0,0288**	**+0,0277**
		Lautstärke	**-0,0247**	0,1092	+0,0041	+0,0030
		Rauschen	**-0,0288**	-0,0041	0,1051	-0,0011
		Alle	**-0,0277**	-0,0030	+0,0011	0,1062
	F-Maß	Keine	0,8267	**-0,0440**	**-0,0464**	**-0,0525**
		Lautstärke	**+0,0440**	0,8708	-0,0023	-0,0085
		Rauschen	**+0,0464**	+0,0023	0,8731	-0,0061
		Alle	**+0,0525**	+0,0085	+0,0061	0,8792

Tabelle A.15: Vergleich der Augmentierungskategorien für das Modell NN.

DS	Maß	Aug.-Kat.	Keine	Lautstärke	Rauschen	Alle
Akkord-Datensatz	BKF	Keine	0,2792	**+0,0140**	**+0,0107**	**+0,0129**
		Lautstärke	**-0,0140**	0,2652	-0,0033	-0,0011
		Rauschen	**-0,0107**	+0,0033	0,2685	+0,0022
		Alle	**-0,0129**	+0,0011	-0,0022	0,2663
	F-Maß	Keine	0,5513	**-0,0310**	**-0,0230**	**-0,0316**
		Lautstärke	**+0,0310**	0,5823	+0,0080	-0,0006
		Rauschen	**+0,0230**	-0,0080	0,5743	-0,0085
		Alle	**+0,0316**	+0,0006	+0,0085	0,5828
MIDI-Datensatz	BKF	Keine	0,1414	**+0,0253**	**+0,0277**	**+0,0283**
		Lautstärke	**-0,0253**	0,1161	+0,0023	+0,0030
		Rauschen	**-0,0277**	-0,0023	0,1138	+0,0007
		Alle	**-0,0283**	-0,0030	-0,0007	0,1131
	F-Maß	Keine	0,8386	**-0,0377**	**-0,0430**	**-0,0451**
		Lautstärke	**+0,0377**	0,8762	-0,0053	**-0,0075**
		Rauschen	**+0,0430**	+0,0053	0,8816	-0,0021
		Alle	**+0,0451**	**+0,0075**	+0,0021	0,8837

Tabelle A.16: Vergleich der Augmentierungskategorien für das Modell NN-RF.

DS	Maß	Aug.-Kat.	Keine	Lautstärke	Rauschen	Alle
Akkord-Datensatz	BKF	Keine	0,1718	**-0,0073**	**-0,0226**	**-0,0211**
		Lautstärke	**+0,0073**	0,1792	**-0,0153**	**-0,0138**
		Rauschen	**+0,0226**	**+0,0153**	0,1945	+0,0015
		Alle	**+0,0211**	**+0,0138**	-0,0015	0,1930
	F-Maß	Keine	0,6334	**-0,0255**	**-0,0102**	**-0,0177**
		Lautstärke	**+0,0255**	0,6589	**+0,0152**	+0,0078
		Rauschen	**+0,0102**	**-0,0152**	0,6436	-0,0074
		Alle	**+0,0177**	-0,0078	+0,0074	0,6511
MIDI-Datensatz	BKF	Keine	0,1139	**+0,0158**	**+0,0172**	**+0,0174**
		Lautstärke	**-0,0158**	0,0981	+0,0014	+0,0016
		Rauschen	**-0,0172**	-0,0014	0,0967	+0,0002
		Alle	**-0,0174**	-0,0016	-0,0002	0,0966
	F-Maß	Keine	0,8439	**-0,0321**	**-0,0387**	**-0,0414**
		Lautstärke	**+0,0321**	0,8760	**-0,0066**	**-0,0093**
		Rauschen	**+0,0387**	**+0,0066**	0,8826	-0,0027
		Alle	**+0,0414**	**+0,0093**	+0,0027	0,8853

Tabelle A.17: Vergleich der Augmentierungskategorien für das Modell NN-SVM.

A.10 Generalisierungstestergebnisse mit verschiedenen Augmentierungskategorien

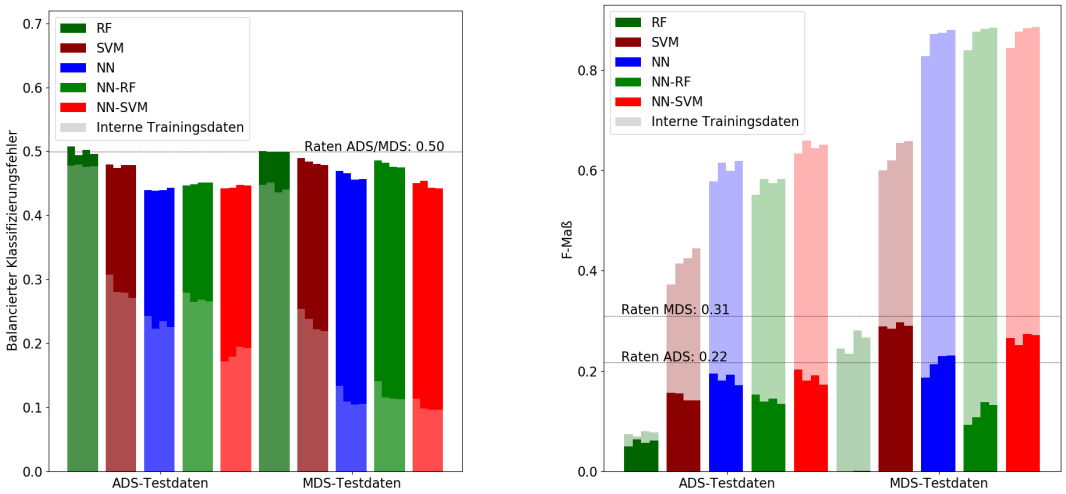


Abbildung A.1: Ergebnisse der Generalisierungstests nach dem BKF (links) und dem F-Maß (rechts). Farblich visualisiert sind die verschiedenen Modelle, trainiert auf dem jeweils anderen Datensatz, sowie zum Vergleich die Klassifikationsleistung der Modelle, wenn auf den datensatzinternen Daten trainiert wird (farblich schwach). Die 4 Balken pro Modell geben jeweils von links nach rechts die folgenden Augmentierungskategorien wieder: Keine, Lautstärke, Rauschen, Alle.

A.11 Vergleich des kombinierten Datensatzes auf einzelnen Instrumenten

Test-DS	Maß	Modell / Train-DS	RF	SVM	NN	NN-RF	NN-SVM
ADS	BKF	KDS	0,4955	0,2944	0,2565	0,3247	0,2436
		ADS	+0,0013	**-0,0100**	-0,0134	-0,0131	**+0,0476**
		MDS	**-0,0046**	**-0,1660**	**-0,2072**	**-0,1623**	**-0,2208**
	F-Maß	KDS	0,0182	0,4461	0,5577	0,4911	0,5864
		ADS	-0,0050	**+0,0260**	+0,0280	**+0,0313**	**+0,0244**
		MDS	**+0,0182**	**+0,2088**	**+0,3542**	**+0,3987**	**+0,3788**
MDS	BKF	KDS	0,4989	0,2949	0,1653	0,1981	0,1405
		ADS	**-0,0011**	**-0,1549**	**-0,3161**	**-0,2990**	**-0,3142**
		MDS	**+0,0019**	**+0,0594**	**+0,0456**	**+0,0515**	**+0,0336**
	F-Maß	KDS	0,0046	0,4491	0,7318	0,7274	0,7517
		ADS	**+0,0046**	**+0,1789**	**+0,6181**	**+0,7020**	**+0,5313**
		MDS	**-0,0074**	**-0,0983**	**-0,0644**	**-0,0754**	**-0,0587**

Tabelle A.18: Vergleich des kombinierten Datensatzes mit dem Akkord- und MIDI-Datensatz auf dem Instrument Bratsche.

Test-DS	Maß	Modell / Train-DS	RF	SVM	NN	NN-RF	NN-SVM
ADS	BKF	KDS	0,4898	0,2874	0,0639	0,0710	0,0531
		ADS	**-0,0076**	**+0,0825**	-0,0202	**-0,0564**	-0,0099
		MDS	**-0,0523**	**-0,1817**	**-0,2861**	**-0,2694**	**-0,3360**
	F-Maß	KDS	0,0416	0,4238	0,9097	0,9170	0,9206
		ADS	**+0,0312**	**+0,0986**	**+0,0591**	**+0,0669**	**+0,0356**
		MDS	-0,0095	**+0,2989**	**+0,7437**	**+0,7496**	**+0,7727**
MDS	BKF	KDS	0,3537	0,2825	0,0732	0,0777	0,0662
		ADS	**-0,1463**	**-0,2348**	**-0,3569**	**-0,3866**	**-0,3628**
		MDS	**+0,0901**	**+0,1082**	+0,0021	**+0,0139**	**+0,0099**
	F-Maß	KDS	0,4520	0,6026	0,9075	0,9109	0,9077
		ADS	**+0,4520**	**+0,2845**	**+0,6603**	**+0,7771**	**+0,6555**
		MDS	**-0,1886**	**-0,1517**	-0,0052	**-0,0141**	**-0,0163**

Tabelle A.19: Vergleich des kombinierten Datensatzes mit dem Akkord- und MIDI-Datensatz auf dem Instrument Sitar.

Test-DS	Maß	Train-DS \ Modell	RF	SVM	NN	NN-RF	NN-SVM
ADS	BKF	KDS	0,4992	0,3361	0,2851	0,3536	0,2666
		ADS	**-0,0008**	-0,0082	**-0,0688**	**-0,0130**	**+0,0409**
		MDS	**-0,0008**	**-0,1569**	**-0,1791**	**-0,1161**	**-0,1828**
	F-Maß	KDS	0,0034	0,3849	0,5185	0,4327	0,5419
		ADS	**+0,0034**	**+0,0363**	**+0,1129**	**+0,0314**	**+0,0423**
		MDS	**+0,0034**	**+0,2995**	**+0,3423**	**+0,2933**	**+0,3293**
MDS	BKF	KDS	0,4998	0,2950	0,1589	0,1986	0,1411
		ADS	**-0,0002**	**-0,1566**	**-0,3157**	**-0,2817**	**-0,2923**
		MDS	**+0,0015**	**+0,0503**	**+0,0250**	**+0,0500**	**+0,0323**
	F-Maß	KDS	0,0008	0,4413	0,7251	0,7222	0,7331
		ADS	**+0,0008**	**+0,1735**	**+0,5790**	**+0,6076**	**+0,4493**
		MDS	**-0,0060**	**-0,0947**	**-0,0507**	**-0,0699**	**-0,0550**

Tabelle A.20: Vergleich des kombinierten Datensatzes mit dem Akkord- und MIDI-Datensatz auf dem Instrument Violine.

Test-DS	Maß	Train-DS \ Modell	RF	SVM	NN	NN-RF	NN-SVM
ADS	BKF	KDS	0,4959	0,3143	0,2176	0,2743	0,2082
		ADS	**+0,0062**	**-0,0475**	**-0,0522**	**-0,0221**	+0,0009
		MDS	**-0,0042**	**-0,1676**	**-0,1937**	**-0,1691**	**-0,2049**
	F-Maß	KDS	0,0163	0,4252	0,6269	0,5930	0,6473
		ADS	**-0,0240**	**+0,0852**	**+0,0800**	**+0,0478**	**+0,0632**
		MDS	**+0,0162**	**+0,2379**	**+0,3441**	**+0,3712**	**+0,3518**
MDS	BKF	KDS	0,4952	0,2605	0,0570	0,0805	0,0467
		ADS	**-0,0048**	**-0,2771**	**-0,3897**	**-0,3973**	**-0,3698**
		MDS	**+0,0159**	**+0,0541**	-0,0004	**+0,0212**	**+0,0100**
	F-Maß	KDS	0,0192	0,5038	0,9035	0,9061	0,9170
		ADS	**+0,0192**	**+0,2773**	**+0,7202**	**+0,8201**	**+0,6384**
		MDS	**-0,0602**	**-0,0930**	-0,0098	**-0,0249**	**-0,0199**

Tabelle A.21: Vergleich des kombinierten Datensatzes mit dem Akkord- und MIDI-Datensatz auf dem Instrument Trompete.

Test-DS	Maß	Train-DS \ Modell	RF	SVM	NN	NN-RF	NN-SVM
ADS	BKF	KDS	0,5000	0,3406	0,3270	0,4036	0,3193
		ADS	+0,0006	+0,0098	**-0,0542**	**-0,0281**	**+0,0530**
		MDS	0,0000	**-0,1520**	**-0,1704**	**-0,0946**	**-0,1771**
	F-Maß	KDS	0,0000	0,3737	0,4403	0,3157	0,4546
		ADS	**-0,0030**	+0,0103	**+0,1016**	**+0,0751**	+0,0150
		MDS	0,0000	**+0,2729**	**+0,3736**	**+0,2971**	**+0,3850**
MDS	BKF	KDS	0,4988	0,3059	0,1400	0,1892	0,1199
		ADS	**-0,0013**	**-0,1845**	**-0,3449**	**-0,3089**	**-0,3529**
		MDS	**+0,0016**	**+0,0654**	**+0,0210**	**+0,0364**	**+0,0215**
	F-Maß	KDS	0,0051	0,4107	0,7851	0,7540	0,8026
		ADS	**+0,0051**	**+0,1553**	**+0,6401**	**+0,7268**	**+0,5507**
		MDS	**-0,0065**	**-0,1246**	**-0,0319**	**-0,0548**	**-0,0386**

Tabelle A.22: Vergleich des kombinierten Datensatzes mit dem Akkord- und MIDI-Datensatz auf dem Instrument Flöte.

Test-DS	Maß	Train-DS \ Modell	RF	SVM	NN	NN-RF	NN-SVM
ADS	BKF	KDS	0,3801	0,3779	0,1129	0,1197	0,1131
		ADS	-0,0041	**+0,0776**	+0,0137	+0,0043	**+0,0404**
		MDS	**-0,1231**	**-0,1041**	**-0,3368**	**-0,3242**	**-0,3265**
	F-Maß	KDS	0,3801	0,3819	0,8029	0,8027	0,8074
		ADS	+0,0071	**-0,0574**	+0,0060	-0,0077	**-0,0225**
		MDS	**+0,1270**	**+0,1733**	**+0,5273**	**+0,5235**	**+0,5196**
MDS	BKF	KDS	0,4460	0,4803	0,3289	0,3066	0,3214
		ADS	**-0,0539**	**-0,0076**	**-0,1684**	**-0,1890**	**-0,1753**
		MDS	-0,0028	**+0,0580**	**+0,0264**	**+0,0292**	**+0,0450**
	F-Maß	KDS	0,6869	0,7289	0,7345	0,7445	0,7497
		ADS	**+0,6815**	**+0,3309**	**+0,4437**	**+0,5720**	**+0,4435**
		MDS	**-0,0347**	**+0,1019**	-0,0111	**-0,0274**	-0,0129

Tabelle A.23: Vergleich des kombinierten Datensatzes mit dem Akkord- und MIDI-Datensatz auf dem Instrument Piano.

A.12 Vergleich von Merkmalsextraktionsschichten auf einzelnen Instrumenten

Typ	DS/Maß / Schicht	ADS		MDS	
		BKF	F-Maß	BKF	F-Maß
NN-RF	Conv256[23]	0,3378	0,4599	0,1466	0,8028
	GMP[25]	0,3295	0,4743	0,1466	0,8000
	Conv64[11]	**0,4994**	**0,0025**	**0,4622**	**0,1390**
NN-SVM	Conv256[23]	0,1961	0,5621	0,1069	0,8104
	GMP[25]	0,2012	0,5677	0,1099	0,8058
	Conv64[11]	**0,2350**	**0,4848**	0,1136	**0,7812**

Tabelle A.24: Vergleich verschiedener Schichten auf dem Instrument Bratsche.

Typ	DS/Maß / Schicht	ADS		MDS	
		BKF	F-Maß	BKF	F-Maß
NN-RF	Conv256[23]	0,1274	0,8502	0,0638	0,9251
	GMP[25]	0,1223	0,8551	0,0631	0,9238
	Conv64[11]	**0,4967**	**0,0128**	**0,0926**	**0,8933**
NN-SVM	Conv256[23]	0,0630	0,8850	0,0563	0,9240
	GMP[25]	0,0663	0,8894	0,0593	0,9191
	Conv64[11]	**0,1409**	**0,5277**	0,0539	**0,9370**

Tabelle A.25: Vergleich verschiedener Schichten auf dem Instrument Sitar.

Typ	DS/Maß / Schicht	ADS		MDS	
		BKF	F-Maß	BKF	F-Maß
NN-RF	Conv256[23]	0,3666	0,4013	0,1486	0,7921
	GMP[25]	0,3593	0,4134	0,1444	0,7918
	Conv64[11]	**0,4993**	**0,0028**	**0,4404**	**0,2121**
NN-SVM	Conv256[23]	0,2258	0,4996	0,1088	0,7880
	GMP[25]	0,2299	0,5051	0,1114	0,7851
	Conv64[11]	**0,2721**	**0,4193**	0,1210	**0,7695**

Tabelle A.26: Vergleich verschiedener Schichten auf dem Instrument Violine.

Typ	DS/Maß / Schicht	ADS		MDS	
		BKF	F-Maß	BKF	F-Maß
NN-RF	Conv256[23]	0,2964	0,5452	0,0593	0,9309
	GMP[25]	0,2898	0,5536	0,0604	0,9284
	Conv64[11]	**0,4999**	**0,0005**	**0,1922**	**0,7575**
NN-SVM	Conv256[23]	0,2073	0,5841	0,0367	0,9369
	GMP[25]	0,2111	0,5812	0,0393	0,9326
	Conv64[11]	**0,2760**	**0,4339**	0,0406	**0,9238**

Tabelle A.27: Vergleich verschiedener Schichten auf dem Instrument Trompete.

Typ	DS/Maß / Schicht	ADS		MDS	
		BKF	F-Maß	BKF	F-Maß
NN-RF	Conv256[23]	0,4316	0,2407	0,1528	0,8088
	GMP[25]	0,4186	0,2756	0,1512	0,8083
	Conv64[11]	**0,4999**	**0,0006**	**0,4411**	**0,2059**
NN-SVM	Conv256[23]	0,2662	0,4396	0,0984	0,8412
	GMP[25]	0,2670	0,4420	0,1032	0,8351
	Conv64[11]	**0,2953**	**0,4111**	**0,1124**	**0,8197**

Tabelle A.28: Vergleich verschiedener Schichten auf dem Instrument Flöte.

Typ	DS/Maß / Schicht	ADS		MDS	
		BKF	F-Maß	BKF	F-Maß
NN-RF	Conv256[23]	0,1154	0,8104	0,2774	0,7719
	GMP[25]	0,1160	0,8098	0,2818	0,7670
	Conv64[11]	**0,1904**	**0,7262**	**0,3864**	**0,7289**
NN-SVM	Conv256[23]	0,0727	0,8299	0,2764	0,7625
	GMP[25]	0,0759	0,8293	0,2829	0,7588
	Conv64[11]	**0,0920**	**0,7818**	0,2826	**0,7405**

Tabelle A.29: Vergleich verschiedener Schichten auf dem Instrument Piano.

A.13 Weitere Ergebnisse

A.13.1 Vergleich auf verrauschten Daten des Akkord-Datensatzes

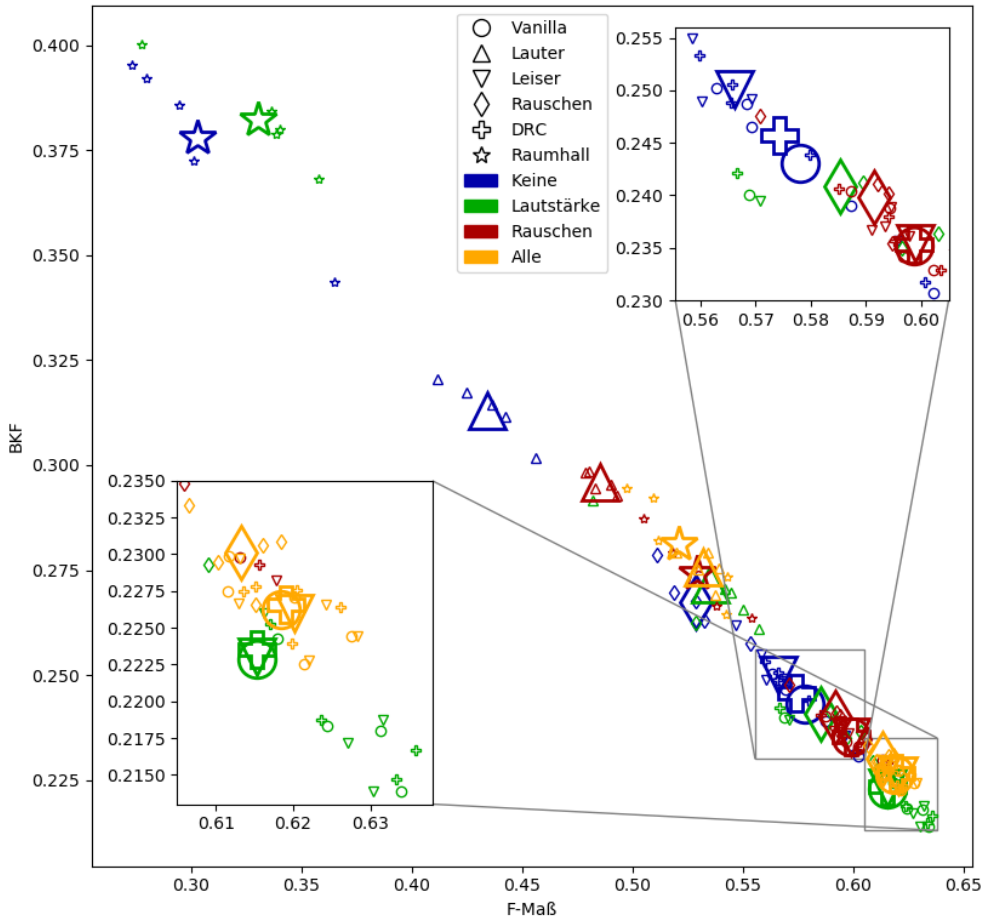


Abbildung A.2: Vergleich von verrauschten Daten auf verschiedenen Augmentierungskategorien des Modells NN auf dem Akkord-Datensatz.

A.13.2 Trainingsdauer auf dem Akkord-Datensatz

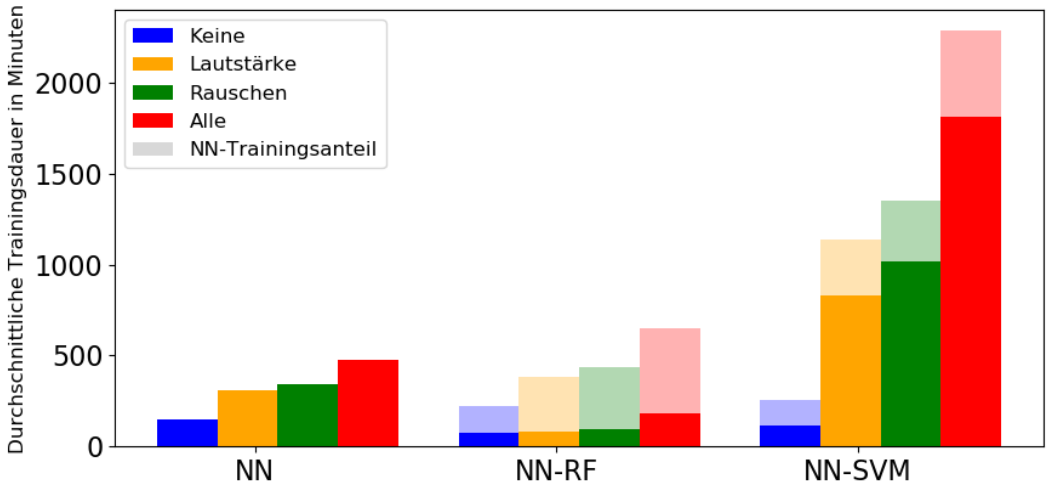


Abbildung A.3: Vergleich der Trainingsdauer der Klassifikatoren NN, NN-RF und NN-SVM auf dem Akkord-Datensatz und den verschiedenen Augmentierungskategorien. Die zusätzliche Trainingsdauer des NN bei Erstellung der hybriden Methoden ist zudem schwach farblich visualisiert.

A.14 Visualisierter Ausblick über Attention und Autoencoder

A.14.1 Attention

Eine Frau wirft eine Frisbee in einem Park.

Ein Hund steht auf einem Hartholz-Boden.

Ein Stop-Schild ist auf einer Straße mit einem Berg im Hintergrund.

Abbildung A.4: Die visualisierte Fokussierung (jeweils rechts) des Originalbildes (jeweils links) bei dem unterstrichenen Wort durch Verwendung von *Attention* (nach [106, Abbildung 3]).

A.14.2 Autoencoder

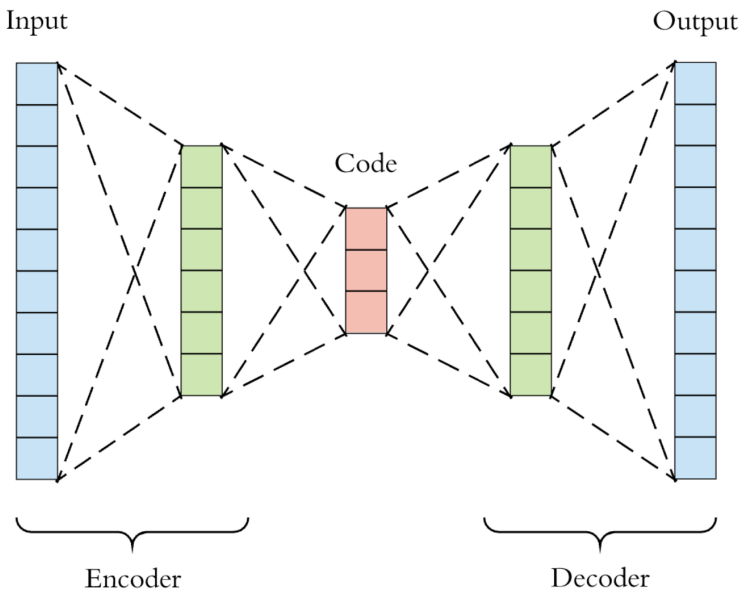


Abbildung A.5: Schematischer Aufbau eines Autoencoders mit Eingabe (*Input*), Ausgabe (*Output*) und komprimierter Darstellung (*Code*), sowie den Bestandteilen der Kompression (*Encoder*) und Rekonstruktion (*Decoder*) [59].

Abbildungsverzeichnis

1.1 Baumdiagramm verschiedener Datenkategorien 2

2.1 Menschliches Ohr 7
2.2 Alias-Effekt 8
2.3 Spektrogramm-Beispiel 9
2.4 Spektrogramm vs. Mel-Spektrogramm 11
2.5 log-Skalierung vs. PCEN-Skalierung 13
2.6 Datenaugmentierung: Spiegelung 15
2.7 Künstlich erzeugte Fehler durch Datenaugmentierung 15
2.8 Vergleich von Trainings- und Validierungsleistung mit und ohne Augmentierungen 16
2.9 Weißes Rauschen 18
2.10 Dynamikkompression 19
2.11 Berechnung des Faltungshalls 19

3.1 Verzerrung und Varianz 23
3.2 Neuron vs. Perzeptron 23
3.3 Aktivierungsfunktionen 24
3.4 2x2-Filterkernel-Beispiel 26
3.5 Dropout 28
3.6 Gradientenabstieg mit Momentum 31
3.7 Trainings- und Validierungsfehler 32
3.8 Beispielhafter Entscheidungsbaum 33
3.9 SVM Klassifizierung 36
3.10 SVM-XOR-Problem 38
3.11 Neural Random Forest 40
3.12 Trend zu hybriden und Ensemble-Methoden 41
3.13 Netzwerk-basiertes Transferlernen 42

4.1 Aufteilung der Kreuzvalidierung 46

4.2 Vergleich der Augmentierungskategorien für die Modelle NN, NN-RF und NN-SVM 52
4.3 Generalisierungstest 56
4.4 Vergleich von verrauschten Daten 62
4.5 Verbindungen zwischen NN und hybriden Methoden 65
4.6 Trainingsdauer auf MIDI-Datensatz 66

A.1 Generalisierungstest nach Augmentierungskategorien 90
A.2 Vergleich von verrauschten Daten 96
A.3 Trainingsdauer auf Akkord-Datensatz 97
A.4 Attention 98
A.5 Autoencoder 98

Abkürzungsverzeichnis

ADAM	Adaptive-Momente-Optimierer
ADS	Akkord-Datensatz
ADT	Audio Degradation Toolbox
AGC	Automatische Verstärkungsregelung
AKN	Augmentierungskategorie: Rauschen und Kompression
AKV	Augmentierungskategorie: Lautstärke
Aug.	Augmentierung
Aug.-Kat.	Augmentierungskategorie
BKF	Balancierter Klassifizierungsfehler
BN	Batch-Normalisierung
CART	Classification and Regression Trees
CQT	Konstante-Q-Transformation
Conv64	Letzte Faltungsschicht mit 64 Kanälen
Conv256	Letzte Faltungsschicht
DRC	Dynamikkompression
DS	Datensatz
FN	Falsch klassifizierte Positivbeispiele
FNR	Falsch-Negativen-Rate
FP	Falsch klassifizierte Negativbeispiele
FPR	Falsch-Positiven-Rate
GAN	Generative Adversarial Network
GMP	Globale Poolingschicht
KDS	Kombinierter Datensatz
LSTM	Long short-term Memory
MDS	MIDI-Datensatz
MFCC	Mel-Frequenz-Cepstrum-Koeffizienten
MS	Mel-Spektrogramm
NN	Künstliches neuronales Netz
NN-RF	Hybride Kombination aus NN und RF
NN-SVM	Hybride Kombination aus NN und SVM
PCEN	Pro-Kanal-Energienormalisierung
ReLU	Rectified Linear Unit
RF	Random-Forest-Klassifikator
RMS	Quadratisches Mittel
RMSProp	Root Mean Square Propagation Algorithm
SGD	Stochastischer Gradientenabstieg
SNR	Signal-Rausch-Verhältnis
STFT	Kurzzeit-Fourier-Transformation
SVM	Stützvektormaschine
TN	Korrekt klassifizierte Negativbeispiele
TP	Korrekt klassifizierte Positivbeispiele

Literaturverzeichnis

[1] Abadi, Martin; Barham, Paul; Chen, Jianmin; Chen, Zhifeng; Davis, Andy; Dean, Jeffrey; Devin, Matthieu; Ghemawat, Sanjay; Irving, Geoffrey; Isard, Michael; Kudlur, Manjunath; Levenberg, Josh; Monga, Rajat; Moore, Sherry; Murray, Derek G.; Steiner, Benoit; Tucker, Paul; Vasudevan, Vijay; Warden, Pete; Wicke, Martin; Yu, Yuan; Zheng, Xiaoqiang: TensorFlow: A system for large-scale machine learning. In: *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, URL `https://www.tensorflow.org/`, 2016, S. 265–283

[2] Abdel-Hamid, O.; Mohamed, A.; Jiang, H.; Penn, G.: Applying Convolutional Neural Networks concepts to hybrid NN-HMM model for speech recognition. In: *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, S. 4277–4280

[3] Amodei, Dario; Anubhai, Rishita; Battenberg, Eric; Case, Carl; Casper, Jared; Catanzaro, Bryan; Chen, Jingdong; Chrzanowski, Mike; Coates, Adam; Diamos, Greg; Elsen, Erich; Engel, Jesse; Fan, Linxi; Fougner, Christopher; Han, Tony; Hannun, Awni; Jun, Billy; LeGresley, Patrick; Lin, Libby; Narang, Sharan; Ng, Andrew; Ozair, Sherjil; Prenger, Ryan; Raiman, Jonathan; Satheesh, Sanjeev; Seetapun, David; Sengupta, Shubho; Wang, Yi; Wang, Zhiqian; Wang, Chong; Xiao, Bo; Yogatama, Dani; Zhan, Jun; Zhu, Zhenyao: Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. In: *ArXiv* (2015)

[4] Araki, S.; Hayashi, T.; Delcroix, M.; Fujimoto, M.; Takeda, K.; Nakatani, T.: Exploring multi-channel features for denoising-autoencoder-based speech enhancement. In: *Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, S. 116–120

[5] Ardabili, Sina; Mosavi, Amir; Várkonyi-Kóczy, Annamária R.: Advances in Machine Learning Modeling Reviewing Hybrid and Ensemble Methods.

In: Várkonyi-Kóczy, Annamária R. (Hrsg.): *Proceedings of Engineering for Sustainable Future*. Cham: Springer International Publishing, 2020, S. 215–227

[6] Atienza, Rowel: *Advanced Deep Learning with Keras: Apply Deep Learning Techniques, Autoencoders, GANs, Variational Autoencoders, Deep Reinforcement Learning, Policy Gradients, and More*. Packt Publishing, 2018

[7] Bertin-Mahieux, Thierry; Ellis, Daniel P.; Whitman, Brian; Lamere, Paul: The Million Song Dataset. In: *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011

[8] Biau, Gérard; Scornet, Erwan; Welbl, Johannes: Neural Random Forests. In: *ArXiv* (2016)

[9] Bishop, Christopher M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 1. Auflage. Springer, 2007

[10] Bittner, Rachel; Salamon, Justin; Tierney, Mike; Mauch, Matthias; Cannam, Chris; Bello, Juan P.: *MedleyDB Audio: A Dataset of Multitrack Audio for Music Research*. 2014

[11] Bittner, Rachel; Wilkins, Julia; Yip, Hanna; Bello, Juan P.: *MedleyDB 2.0 Audio*. 2016

[12] Breiman, L.: Bagging Predictors. In: *Machine Learning* (1996), S. 123–140

[13] Breiman, L.: Random Forests. In: *Machine Learning* (2001), S. 5–32

[14] Breiman, L.; Friedman, J. H.; Olshen, R. A.; Stone, C. J.: *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984

[15] Chinchor, Nancy: MUC-4 Evaluation Metrics. In: *Proceedings of the 4th Conference on Message Understanding*. USA: Association for Computational Linguistics, 1992 (MUC4 '92), S. 22–29

[16] Chittka, Lars; Brockmann, Axel: Perception Space—The Final Frontier. In: *PLoS biology* (2005), S. 137

[17] Choi, K.; Fazekas, G.; Sandler, M.; Cho, K.: Convolutional recurrent neural networks for music classification. In: *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, S. 2392–2396

[18] Chollet, François u. a.: *Keras*. `https://keras.io`. 2015

[19] Cortes, Corinna; Vapnik, Vladimir: Support-Vector Networks. In: *Machine Learning* (1995), S. 273–297

[20] Costa, Yandre M.; Oliveira, Luiz S.; Silla, Carlos N.: An evaluation of Convolutional Neural Networks for music classification using spectrograms. In: *Applied Soft Computing* (2017), S. 28 – 38

[21] Cui, X.; Goel, V.; Kingsbury, B.: Data Augmentation for Deep Neural Network Acoustic Modeling. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2015), S. 1469–1477

[22] Efron, B.: Bootstrap Methods: Another Look at the Jackknife. In: *The Annals of Statistics* (1979), S. 1–26

[23] Engel, Jesse; Resnick, Cinjon; Roberts, Adam; Dieleman, Sander; Eck, Douglas; Simonyan, Karen; Norouzi, Mohammad: Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. In: *ArXiv* (2017)

[24] Eronen, A.: Comparison of features for musical instrument recognition. In: *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No.01TH8575)*, 2001, S. 19–22

[25] Essid, Slim; Richard, Gaël; David, Bertrand: Instrument recognition in polyphonic music based on automatic taxonomies. In: *Audio, Speech, and Language Processing, IEEE Transactions on* (2006), S. 68 – 80

[26] Feng, X.; Zhang, Y.; Glass, J.: Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition. In: *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, S. 1759–1763

[27] Fernandes, B. J. T.; Cavalcanti, G. D. C.; Ren, T. I.: Lateral Inhibition Pyramidal Neural Network for Image Classification. In: *IEEE Transactions on Cybernetics* (2013), S. 2082–2092

[28] Fink, Gernot A.: *Vorlesungsskript Mustererkennung, Fakultät für Informatik, TU Dortmund*. 2016

[29] Friedrich, Hans J.: *Tontechnik für Mediengestalter*. Springer-Verlag, 2008

[30] Fulop, Sean A.: *The Fourier Power Spectrum and Spectrogram.* S. 69–106. In: *Speech Spectrum Analysis.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2011

[31] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron: *Deep Learning - Das umfassende Handbuch: Grundlagen, aktuelle Verfahren und Algorithmen, neue Forschungsansätze.* MITP, 2018

[32] Goodfellow, Ian J.; Shlens, Jonathon; Szegedy, Christian: Explaining and Harnessing Adversarial Examples. (2014)

[33] Gururani, Siddharth; Sharma, Mohit; Lerch, Alexander: An Attention Mechanism for Musical Instrument Recognition. In: *ArXiv* (2019)

[34] Gómez, Juan S.; Abesser, Jakob; Cano, Estefanía: Jazz Solo Instrument Classification with Convolutional Neural Networks, Source Separation, and Transfer Learning. In: *Proceedings of the 19th International Society for Music Information Retrieval Conference.* Paris, France: ISMIR, 2018, S. 577–584

[35] Han, Yoonchang; Kim, Jaehun; Lee, Kyogu; Han, Yoonchang; Kim, Jaehun; Lee, Kyogu: Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music. In: *IEEE/ACM Transactions Audio, Speech and Language Processing* (2017), S. 208–221

[36] Hannun, Awni; Case, Carl; Casper, Jared; Catanzaro, Bryan; Diamos, Greg; Elsen, Erich; Prenger, Ryan; Satheesh, Sanjeev; Sengupta, Shubho; Coates, Adam; Ng, Andrew Y.: Deep Speech: Scaling up end-to-end speech recognition. In: *ArXiv* (2014)

[37] Hanssian, Sevag: *Audio Degradation Toolbox in Python.* – URL `https://github.com/sevagh/audio-degradation-toolbox`. – [Online; accessed May 6, 2020]

[38] Hinton, Geoffrey E.; Srivastava, Nitish; Krizhevsky, Alex; Sutskever, Ilya; Salakhutdinov, Ruslan: Improving neural networks by preventing co-adaptation of feature detectors. In: *ArXiv* (2012)

[39] Hochreiter, Sepp; Schmidhuber, Jürgen: Long Short-term Memory. In: *Neural computation* (1997), S. 1735–1780

[40] Ioffe, Sergey; Szegedy, Christian: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, JMLR.org, 2015 (ICML'15), S. 448–456

[41] J. Guerrero-Turrubiates, J. d.; Gonzalez-Reyna, S. E.; Ledesma-Orozco, S. E.; Avina-Cervantes, J. G.: Pitch estimation for musical note recognition using Artificial Neural Networks. In: *Proceedings of the 2014 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, 2014, S. 53–58

[42] Jayalakshmi, T.; A., Santhakumaran: Statistical normalization and back propagation for classification. In: *International Journal Computer Theory Engineering (IJCTE)* (2011), S. 89–93

[43] Khan, Asifullah; Sohail, Anabia; Zahoora, Umme; Qureshi, Aqsa S.: A Survey of the Recent Architectures of Deep Convolutional Neural Networks. In: *ArXiv* (2019)

[44] Kingma, Diederik P.; Ba, Jimmy: Adam: A Method for Stochastic Optimization. In: *International Conference on Learning Representations* (2014)

[45] Klapuri, Anssi; Davy, Manuel: *Signal Processing Methods for Music Transcription*. 1. Auflage. Springer Publishing Company, Incorporated, 2010

[46] Ko, Tom; Peddinti, Vijayaditya; Povey, Daniel; Khudanpur, Sanjeev: Audio augmentation for speech recognition. In: *Proceedings of INTERSPEECH*, ISCA, 2015, S. 3586–3589

[47] Kong, Q.; Yu, C.; Xu, Y.; Iqbal, T.; Wang, W.; Plumbley, M. D.: Weakly Labelled AudioSet Tagging With Attention Neural Networks. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2019), S. 1791–1802

[48] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira, F. (Hrsg.); Burges, C. J. C. (Hrsg.); Bottou, L. (Hrsg.); Weinberger, K. Q. (Hrsg.): *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, S. 1097–1105

[49] Law, Edith; West, Kris; Mandel, Michael; Bay, Mert; Downie, J Stephen: Evaluation of algorithms using games: The case of music tagging. In:

Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, 2009, S. 387–392

[50] Lewis, Rory A.; Zhang, Xin; Raś, Zbigniew W.: Knowledge discovery-based identification of musical pitches and instruments in polyphonic sounds. In: *Engineering Applications of Artificial Intelligence* (2007), S. 637 – 645

[51] Li, Xiaoquan; Wang, Kaiqi; Soraghan, John; Ren, Jinchang: Fusion of Hilbert-Huang Transform and Deep Convolutional Neural Network for Predominant Musical Instruments Recognition. In: Romero, Juan (Hrsg.); Ekárt, Anikó (Hrsg.); Martins, Tiago (Hrsg.); Correia, João (Hrsg.): *Proceedings of Artificial Intelligence in Music, Sound, Art and Design*. Cham: Springer International Publishing, 2020, S. 80–89

[52] Lipshitz, Stanley; Vanderkooy, John: Pulse-Code Modulation–An Overview. In: *Journal of the Audio Engineering Society. Audio Engineering Society* (2004)

[53] Lipton, Zachary C.; Elkan, Charles; Naryanaswamy, Balakrishnan: Optimal Thresholding of Classifiers to Maximize F1 Measure. In: Calders, Toon (Hrsg.); Esposito, Floriana (Hrsg.); Hüllermeier, Eyke (Hrsg.); Meo, Rosa (Hrsg.): *Proceedings of Machine Learning and Knowledge Discovery in Databases*, Springer Berlin Heidelberg, 2014, S. 225–239

[54] Lostanlen, V.; Salamon, J.; Cartwright, M.; McFee, B.; Farnsworth, A.; Kelling, S.; Bello, J. P.: Per-Channel Energy Normalization: Why and How. In: *IEEE Signal Processing Letters* (2019), S. 39–43

[55] Lostanlen, Vincent; Salamon, Justin; Farnsworth, Andrew; Kelling, Steve; Bello, Juan P.: Robust sound event detection in bioacoustic sensor networks. In: *PLOS ONE* (2019), S. 1–31

[56] Malik, Miroslav; Adavanne, Sharath; Drossos, Konstantinos; Virtanen, Tuomas; Ticha, Dasa; Jarina, Roman: Stacked Convolutional and Recurrent Neural Networks for Music Emotion Recognition. In: *ArXiv* (2017)

[57] Mancini, Ron; Carter, Bruce: Chapter 12 - Op Amp Noise Theory and Applications. In: *Op Amps for Everyone (Third Edition)*. 3. Auflage. Boston: Newnes, 2009, S. 163 – 188

[58] Mariani, Giovanni; Scheidegger, Florian; Istrate, Roxana; Bekas, Costas; Malossi, Cristiano: BAGAN: Data Augmentation with Balancing GAN. In: *ArXiv* (2018)

[59] Matthew Stewart, TowardsDataScience: *Comprehensive Introduction to Autoencoders.* 2019. – URL `https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368`. – [Online; accessed April 30, 2020]

[60] Mauch, Matthias; Ewert, Sebastian: The Audio Degradation Toolbox and its Application to Robustness Evaluation. In: *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR 2013).* Curitiba, Brazil, 2013

[61] McFee Brian; Raffel Colin; Liang Dawen; P.W. Ellis Daniel; McVicar Matt; Battenberg Eric; Nieto Oriol: librosa: Audio and Music Signal Analysis in Python. In: Huff Kathryn (Hrsg.); Bergstra James (Hrsg.): *Proceedings of the 14th Python in Science Conference*, 2015, S. 18 – 24

[62] Mertens, Stephan: The Easiest Hard Problem: Number Partitioning. In: *Computational Complexity and Statistical Physics* (2003)

[63] Mingers, John: An Empirical Comparison of Pruning Methods for Decision Tree Induction. In: *Machine Learning* (1989), S. 227–243

[64] Minsky, Marvin; Papert, Seymour: *Perceptrons: An Introduction to Computational Geometry.* Cambridge, MA, USA: MIT Press, 1969

[65] Muller, M.; Ellis, D. P. W.; Klapuri, A.; Richard, G.: Signal Processing for Music Analysis. In: *IEEE Journal of Selected Topics in Signal Processing* (2011), S. 1088–1110

[66] Müller, Meinard: *Information Retrieval for Music and Motion.* Berlin, Heidelberg: Springer-Verlag, 2007

[67] Ng, Hong-Wei; Nguyen, Viet D.; Vonikakis, Vassilios; Winkler, Stefan: Deep Learning for Emotion Recognition on Small Datasets Using Transfer Learning. In: *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction.* New York, NY, USA: Association for Computing Machinery, 2015 (ICMI '15), S. 443–449

[68] Oliphant, Travis E.: *A guide to NumPy.* Trelgol Publishing USA, 2006

[69] Oord, Aaron van den; Dieleman, Sander; Zen, Heiga; Simonyan, Karen; Vinyals, Oriol; Graves, Alex; Kalchbrenner, Nal; Senior, Andrew; Kavukcuoglu, Koray: WaveNet: A Generative Model for Raw Audio. In: *ArXiv* (2016)

[70] O'Shaughnessy, Douglas: *Speech communication: human and machine.* Addison-Wesley, 1987

[71] Owers, James: *An Exploration into the Application of Convolutional Neural Networks for Instrument Classification on Monophonic Note Samples*, University of Edinburgh, Data Science, Masterarbeit, 2016

[72] O'Shaughnessy, Douglas: Invited paper: Automatic speech recognition: History, methods and challenges. In: *Pattern Recognition* (2008), S. 2965 – 2979

[73] Pedregosa, Fabian; Varoquaux, Gaël; Gramfort, Alexandre; Michel, Vincent; Thirion, Bertrand; Grisel, Olivier; Blondel, Mathieu; Louppe, Gilles; Prettenhofer, Peter; Weiss, Ron; Dubourg, Vincent; Vanderplas, Jacob; Passos, Alexandre; Cournapeau, David; Brucher, Matthieu; Perrot, Matthieu; Duchesnay, Edouard: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* (2011), S. 2825–2830

[74] Pons, J.; Lidy, T.; Serra, X.: Experimenting with musically motivated convolutional neural networks. In: *Proceedings of the 14th International Workshop on Content-Based Multimedia Indexing (CBMI)*, 2016, S. 1–6

[75] Prechelt, Lutz: Early Stopping-But When? In: Orr, Genevieve B. (Hrsg.); Müller, Klaus-Robert (Hrsg.): *Neural Networks: Tricks of the Trade.* Springer, 1996 (Lecture Notes in Computer Science), S. 55–69

[76] Qian, Y.; Bi, M.; Tan, T.; Yu, K.: Very Deep Convolutional Neural Networks for Noise Robust Speech Recognition. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2016), S. 2263–2276

[77] Quinlan, J. R.: Induction of Decision Trees. In: *Machine Learning* (1986), S. 81–106

[78] Rosner, Aldona; Kostek, Bozena: Automatic Music Genre Classification Based on Musical Instrument Track Separation. In: *Journal of Intelligent Information Systems* (2018), S. 363–384

[79] Sahidullah, Md.; Saha, Goutam: Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition. In: *Speech Communication* (2012), S. 543 – 565

[80] Sainath, Tara N.; Kingsbury, Brian; Saon, George; Soltau, Hagen; Mohamed, Abdel rahman; Dahl, George; Ramabhadran, Bhuvana: Deep Convolutional Neural Networks for Large-scale Speech Tasks. In: *Neural Networks* (2015), S. 39 – 48

[81] Salamon, Justin; Bello, Juan P.: Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification. In: *IEEE Signal Processing Letters* (2017), S. 279–283

[82] Sammut, Claude; Webb, Geoffrey I.: *Encyclopedia of Machine Learning*. 1. Auflage. Springer Publishing Company, Incorporated, 2011

[83] Schmidhuber, Jürgen: Deep learning in neural networks: An overview. In: *Neural networks: the official journal of the International Neural Network Society* (2015), S. 85–117

[84] Seipel, Fabian: *Music Instrument Identification using Convolutional Neural Networks*, TU Berlin, Institut für Kommunikation und Sprache, Fachgebiet Audiotechnologie, Masterarbeit, 2018

[85] Shorten, Connor; Khoshgoftaar, Taghi: A survey on Image Data Augmentation for Deep Learning. In: *Journal of Big Data* (2019)

[86] Solanki, Arun; Pandey, Sachin: Music instrument recognition using deep convolutional neural networks. In: *International Journal of Information Technology* (2019)

[87] Srivastava, Nitish; Hinton, Geoffrey; Krizhevsky, Alex; Sutskever, Ilya; Salakhutdinov, Ruslan: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In: *Journal of Machine Learning Research* (2014), S. 1929–1958

[88] Sterpu, George; Saam, Christian; Harte, Naomi: Attention-Based Audio-Visual Fusion for Robust Automatic Speech Recognition. In: *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, Association for Computing Machinery, 2018 (ICMI '18), S. 111–115

[89] Stevens, S. S.; Volkmann, J.; Newman, E. B.: A Scale for the Measurement of the Psychological Magnitude Pitch. In: *The Journal of the Acoustical Society of America* (1937), S. 185–190

[90] Tan, Chuanqi; Sun, Fuchun; Kong, Tao; Zhang, Wenchang; Yang, Chao; Liu, Chunfang: A Survey on Deep Transfer Learning. In: *ArXiv* (2018)

[91] Tensorflow: *Tensorflow Datenaugmentierung*. 2020. – URL `https://www.tensorflow.org/tutorials/images/data_augmentation`. – [Online; accessed March 23, 2020]

[92] Turk, Matija; Turk, Ivan; Dimkaroski, Ljuben; Blackwell, Bonnie; Horusitzky, François; Otte, Marcel; Bastiani, Giuliano; Korat, Lidija: The Mousterian Musical Instrument from the Divje babe I cave (Slovenia): Arguments on the Material Evidence for Neanderthal Musical Behaviour. In: *L'Anthropologie* (2018)

[93] Van Rossum, Guido; Drake, Fred L.: *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009

[94] Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N.; Kaiser, Ł u.; Polosukhin, Illia: Attention is All you Need. In: Guyon, I. (Hrsg.); Luxburg, U. V. (Hrsg.); Bengio, S. (Hrsg.); Wallach, H. (Hrsg.); Fergus, R. (Hrsg.); Vishwanathan, S. (Hrsg.); Garnett, R. (Hrsg.): *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, S. 5998–6008

[95] Vatolkin, Igor: Generalisation Performance of Western Instrument Recognition Models in Polyphonic Mixtures with Ethnic Samples. In: Correia, João (Hrsg.); Ciesielski, Vic (Hrsg.); Liapis, Antonios (Hrsg.): *Proceedings of Computational Intelligence in Music, Sound, Art and Design*. Cham: Springer International Publishing, 2017, S. 304–320

[96] Vatolkin, Igor; Rudolph, Günter: Comparison of Audio Features for Recognition of Western and Ethnic Instruments in Polyphonic Mixtures. In:

Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018, 2018, S. 554–560

[97] Virtanen, Pauli; Gommers, Ralf; Oliphant, Travis E.; Haberland, Matt; Reddy, Tyler; Cournapeau, David; Burovski, Evgeni; Peterson, Pearu; Weckesser, Warren; Bright, Jonathan; van der Walt, Stéfan J.; Brett, Matthew; Wilson, Joshua; Jarrod Millman, K.; Mayorov, Nikolay; Nelson, Andrew R. J.; Jones, Eric; Kern, Robert; Larson, Eric; Carey, CJ; Polat, İlhan; Feng, Yu; Moore, Eric W.; Vand erPlas, Jake; Laxalde, Denis; Perktold, Josef; Cimrman, Robert; Henriksen, Ian; Quintero, E. A.; Harris, Charles R.; Archibald, Anne M.; Ribeiro, Antônio H.; Pedregosa, Fabian; van Mulbregt, Paul; Contributors, SciPy 1. 0.: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. In: *Nature Methods* (2020)

[98] Wang, Y.; Getreuer, P.; Hughes, T.; Lyon, R. F.; Saurous, R. A.: Trainable frontend for robust and far-field keyword spotting. In: *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, S. 5670–5674

[99] Wen, Haiguang; Shi, Junxing; Zhang, Yizhen; Lu, Kun-Han; Cao, Jiayue; Liu, Zhongming: Neural Encoding and Decoding with Deep Learning for Dynamic Natural Vision. In: *Cerebral Cortex* (2018), S. 4136–4160

[100] Werbos, P. J.: Backpropagation through time: what it does and how to do it. In: *Proceedings of the IEEE* (1990), S. 1550–1560

[101] Wikipedia, the free encyclopedia, Loisel: *Aliased chessboard*. 2003. – URL `https://commons.wikimedia.org/wiki/File:Aliased.png`. – [Online; accessed March 18, 2020]

[102] Wikipedia, the free encyclopedia, Mayranna: *Perceptron*. 2013. – URL `https://commons.wikimedia.org/wiki/File:Perceptron_moj.png`. – [Online; accessed March 10, 2020]

[103] Wikipedia, the free encyclopedia, Quasar J.: *Neuron Hand-tuned*. 2009. – URL `https://commons.wikimedia.org/wiki/File:Neuron_Hand-tuned.svg`. – [Online; accessed March 10, 2020]

[104] Wikipedia, the free encyclopedia, The Rain M.: *Convolution Raster*. 2006. – URL `https://commons.wikimedia.org/wiki/File:Convolution_Cross_Multiplication.jpg`. – [Online; accessed March 28, 2020]

[105] Xiong, Wayne; Droppo, Jasha; Huang, Xuedong; Seide, Frank; Seltzer, Mike; Stolcke, Andreas; Yu, Dong; Zweig, Geoffrey: Achieving Human Parity in Conversational Speech Recognition. In: *ArXiv* (2016)

[106] Xu, Kelvin; Ba, Jimmy; Kiros, Ryan; Cho, Kyunghyun; Courville, Aaron; Salakhudinov, Ruslan; Zemel, Rich; Bengio, Yoshua: Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In: Bach, Francis (Hrsg.); Blei, David (Hrsg.): *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France: PMLR, 2015 (Proceedings of Machine Learning Research), S. 2048–2057

[107] Yosinski, Jason; Clune, Jeff; Bengio, Yoshua; Lipson, Hod: How transferable are features in deep neural networks? In: Ghahramani, Z. (Hrsg.); Welling, M. (Hrsg.); Cortes, C. (Hrsg.); Lawrence, N. D. (Hrsg.); Weinberger, K. Q. (Hrsg.): *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014, S. 3320–3328

[108] Zinemanas, P.; Cancela, P.; Rocamora, M.: End-to-end Convolutional Neural Networks for Sound Event Detection in Urban Environments. In: *Proceedings of 24th Conference of Open Innovations Association (FRUCT)*, 2019, S. 533–539

Eidesstattliche Versicherung
(Affidavit)

Adrian, Benedikt
Name, Vorname
(Last name, first name)

140960
Matrikelnr.
(Enrollment number)

Ich versichere hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit/Masterarbeit* mit dem folgenden Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present Bachelor's/Master's* thesis with the following title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution.

Titel der Bachelor-/Masterarbeit*:
(Title of the Bachelor's/ Master's* thesis):

Implementierung von hybriden Methoden zur Instrumentenerkennung in verrauschten Musikdaten

*Nichtzutreffendes bitte streichen
(Please choose the appropriate)

Duisburg, 22.05.2020
Ort, Datum
(Place, date)

B. Adr
Unterschrift
(Signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to €50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, section 63, subsection 5 of the North Rhine-Westphalia Higher Education Act (*Hochschulgesetz*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:**

Duisburg, 22.05.2020
Ort, Datum
(Place, date)

B. Adr
Unterschrift
(Signature)

**Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.