

# Güteklassifikation von Musik mit Hilfe künstlicher Intelligenz

Studienarbeit

Angewandte Informatik  
Duale Hochschule Baden-Württemberg

Von Michael Wellner

13. Juni 2011

**Matrikelnr./ Kurs:** 123905, TIT08-AIA  
**Ausbildungsbetrieb:** IBM Deutschland GmbH, Ehningen  
**Betreuer:** Timo Holzherr

Michael Wellner

*Güteklassifikation von Musik mit Hilfe künstlicher Intelligenz*, ©2011.

Website:

<http://www.michaelwellner.de>

E-Mail:

[info@michaelwellner.de](mailto:info@michaelwellner.de)

## **Ehrenwörtliche Erklärung**

Ich erkläre ehrenwörtlich,

1. dass ich meine Studienarbeit ohne fremde Hilfe angefertigt habe;
2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe;
3. dass ich meine Projektarbeit bei keiner anderen Prüfung vorgelegt habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

*München, 2011*

---

Michael Wellner

## **Abstract**

Meine Studienarbeit „*Güteklassifikation von Musik mit Hilfe künstlicher Intelligenz*“ analysiert die Möglichkeiten zur Nutzung von künstlicher Intelligenz zu Klassifizierung von Musik. So sollen gut klingende Melodien möglichst von nicht gut klingenden unterschieden werden können. Im Verlauf der Arbeit wird daher zunächst die Frage geklärt, was gute Musik ausmacht. Weiterhin werden verschiedene Ansätze der künstlichen Intelligenz beleuchtet. Im zweiten Teil der Arbeit wird das Wissen über die Musik und die Grundlagen der künstlichen Intelligenz kombiniert und erörtert, welche Möglichkeiten sich bieten, ein intelligentes System umzusetzen. So ist auch die Erstellung von zwei einfachen Prototypen Inhalt dieser Arbeit.

## Hinweise

Diese Arbeit entstand im Rahmen der Studienarbeit im 5. bzw. 6. Semester des Studiums der Angewandte Informatik an der Dualen Hochschule Baden-Württemberg in Stuttgart. Der Zeitraum der Arbeit lag zwischen Januar und Juli 2001.

Um den Rahmen der Arbeit nicht zu sprengen werden einige angeschnittene Themengebiete nicht von Grund auf erläutert. Es wird davon ausgegangen, dass der Leser bereits über fundierte Kenntnisse der Informatik verfügt – Zum Beispiel durch ein (fast) abgeschlossenes Informatik Studium. Gegebenenfalls wird jedoch an den entsprechenden Stellen auf grundlegende Literatur verwiesen.

Außerdem wird auf ein Abkürzungsverzeichnis und einen Glossar verzichtet, da beide kaum Inhalt vorweisen würden. Alle wichtigen Begriffe sind in den Grundlagen der Arbeit erläutert. Die einzige Abkürzung die ab und an verwendet wird ist KI (Künstliche Intelligenz).

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>8</b>
<b>Listings</b>	<b>8</b>
<b>1 Einleitung</b>	<b>9</b>
<b>2 Möglichkeiten zur Klassifikation von Musik. Oder: Was ist gute Musik?</b>	<b>11</b>
2.1 Die Komposition . . . . .	12
2.2 Das Arrangement . . . . .	13
2.3 Der Text . . . . .	14
<b>3 Verfahren der künstlichen Intelligenz</b>	<b>15</b>
3.1 Darstellung von Wissen mit Hilfe von Logik . . . . .	17
3.2 Darstellung von Wissen in semantischen Netzen . . . . .	19
3.3 Darstellung von vagen Wissen . . . . .	21
3.4 Problemlösung mittels Suche . . . . .	23
3.5 Wissensbasierte Systeme . . . . .	24
3.6 Künstliche neuronale Netze . . . . .	27
<b>4 Digitale Verarbeitung von Musik</b>	<b>35</b>
4.1 Musikanalyse nach dem Vorbild der Spracherkennung . . . . .	35
4.2 Nutzung des MIDI-Formats . . . . .	36
4.3 Komprimierte Musik . . . . .	37
<b>5 Definition des Prototyps</b>	<b>39</b>
5.1 Auswahl des Datenformates . . . . .	39
5.2 Auswahl der zu untersuchenden Merkmale von Musik . . . . .	41
5.3 Auswahl von Ansätzen der künstlichen Intelligenz . . . . .	42
5.4 Weitere Vorüberlegungen . . . . .	46
<b>6 Realisierung</b>	<b>49</b>
6.1 Architektur-Überblick . . . . .	49
6.2 Einlesen und Analyse eines Musikstückes . . . . .	50
6.3 Entwicklung eines einfachen MusicStream-Generators . . . . .	54

6.4	Implementierung eines neuronalen Netzes . . . . .	57
6.4.1	Entwicklung eines Frameworks zur Erstellung neuronaler Netze . .	57
6.4.2	Aufbau eines Netzes zur Analyse von Musik . . . . .	62
6.5	Tests am neuronalen Netz . . . . .	65
6.6	Implementierung eines einfachen wissensbasierten Systems . . . . .	67
6.7	Tests am wissensbasierten System . . . . .	69
<b>7</b>	<b>Zusammenfassung und Ausblick</b>	<b>71</b>
	<b>Anhang</b>	<b>73</b>
	Anhang A - Lern- und Testdaten . . . . .	73
	Anhang B - Projektdateien . . . . .	74
	<b>Literatur</b>	<b>75</b>

## Abbildungsverzeichnis

1	Ein Beispiel für ein semantisches Netz. [Kriz (2010)] . . . . .	20
2	Charakteristische Funktion für die Geraden Zahlen . . . . .	22
3	Charakteristische Funktion für <i>alt</i> . . . . .	23
4	Aufbau eines Wissenbasierten Systems [G.F.Luger (2001)] . . . . .	26
5	Biologisches Neuron. [Reif und Reschenhofer (2005)] . . . . .	29
6	Schematische Darstellung des Jordan-Netzes. [Elmenreich (2011)] . . . . .	32
7	Schematische Darstellung des Kohonen-Netzes. [Reif und Reschenhofer (2005), Seite 10] . . . . .	33
8	Hauptbestandteile des Prototyps . . . . .	50
9	Die Komponente Song mit ihren wichtigsten Bestandteilen. . . . .	52
10	Wiederholungen finden; Schritt 1 . . . . .	53
11	Wiederholungen finden; Schritt 2 . . . . .	53
12	Wiederholungen finden; Schritt 3 . . . . .	53
13	Die Funktion $f(x) = x^2$ . . . . .	55
14	Architektur eines Backpropagation Netzes . . . . .	58
15	Ein neuronales Netz als objektorientiertes Klassen-Modell . . . . .	58
16	Eine sigmoide Funktion als Aktivierungsfunktion. . . . .	60
17	Eine vereinfachte Darstellung des Frameworks . . . . .	62
18	Die veränderte Architektur des Netzes . . . . .	66

## Listings

1	Musikregeln mit Prolog überprüfen . . . . .	18
2	Eine einfache Schwellwert-Funktion . . . . .	59
3	Der Backpropagation-Algorithmus . . . . .	60
4	Normalisieren des Notenwertes . . . . .	63

# 1 Einleitung

Wir erleben es fast unterbewusst in unserem Alltag - Ob im Auto, beim Einkaufen, in einem Fußballstadion oder an einem beliebig anderen Ort: Wir hören Musik. Mit dem Hören der Musik ist zeitgleich eine Bewertung des Gehörten verbunden - Gefällt oder missfällt uns dieses Lied? Die Entscheidung ob wir eine Melodie oder ein Musikstück mögen findet völlig automatisch in unserem Unterbewusstsein statt. Auf die Frage warum uns ein Stück gefällt oder nicht, können wir in aller Regel keine sachliche Antwort geben. Es gefällt oder wir wollen diesen „Mist“ einfach nicht mehr hören.

Gefallen oder nicht gefallen scheint also auf keiner darstellbaren Funktion zu basieren, welche auch durch eine Maschine bzw. einen Computer umgesetzt werden kann. Offensichtlich scheint das Klassifizieren bzw. das Bilden einer Meinung ausschließlich uns Menschen, bestenfalls noch anderen intelligenten Lebewesen vorbehalten zu sein. Jedoch wurden in den letzten Jahren durch die Weiterentwicklung der künstlichen Intelligenz und Arbeiten im Bereich des emotional Computing große Fortschritte gemacht. So konnten bereits andere menschliche Verhaltensweisen nachgebildet werden.

Die Aufgabe dieser Arbeit ist es zu untersuchen inwiefern es möglich ist, die subjektive Bewertung von Musik durch einen Computer zu realisieren. Hierfür wird zunächst der menschliche Bewertungsvorgang analysiert. Einblicke in die Musiktheorie und Grundlagen der menschlichen Wahrnehmung bilden die Grundlage für die Analyse der technischen Möglichkeiten. Als mögliche Lösungsansätze werden naturanaloge Verfahren aus dem Bereich des Soft Computing betrachtet. Weiterhin sollen aber auch klassische Ansätze der künstlichen Intelligenz betrachtet werden bzw. parallelen zu anderen Forschungsgebieten in Betracht gezogen werden. Nach Möglichkeit soll am Ende der Arbeit ein Prototyp eines Systems entwickelt werden, welches einen bestimmten Musikgeschmack erlernen und so neue Musikstücke nach dem erlernten Geschmack beurteilen kann.

Motivation zu dieser Arbeit bieten mehrere nützliche Einsatzgebiete der automatisierten Klassifikation von Musik.

Zum einen wäre es möglich Maschinen nicht nur die Fähigkeit zu geben „gehörte“ Eindrücke zu bewerten, sondern sie würden dann auch die Grundlage besitzen selbst Musik

zu erzeugen bzw. eigene Musik zu „komponieren“, also einen künstlerischen Schaffungsprozess zu vollziehen. So wären Computer selbst in der Lage Neues zu schaffen und es könnte ihnen eine gewisse Intelligenz unterstellt werden.

Der Einsatz eines solchen Programms ist aber auch in Bereichen außerhalb der künstlichen Intelligenz denkbar. Bereits heute gibt es verschiedene Programme und Dienste, die Musikhörern basierend auf den Musiktiteln ihrer Musikkbibliothek bzw. einer Liste von Titeln, welche ihnen offensichtlich gefallen, neue Titellisten mit vorgeschlagener Musik erzeugen. Diese Programme und Dienste arbeiten jedoch meist auf Basis von Benutzerdaten anderer Anwender. Sie erstellen Vorschläge basierend auf Statistiken nach dem Prinzip „Andere Benutzer die Titel X gehört haben, hörten auch Titel Y“. Durch den Einsatz einer automatisierten, geschmacksbasierten Klassifizierung wäre es möglich neue Titelvorschläge ohne die Benutzung anderer Anwenderdaten zu finden. So können auch unbekannte, bisher kaum (von anderen) gehörte Titel gefunden werden.

Sicherlich würde man noch weitere Einsatzgebiete finden. Als Motivation zu dieser Arbeit reichen die zwei Genannten jedoch aus.

## 2 Möglichkeiten zur Klassifikation von Musik. Oder: Was ist gute Musik?

Bevor in Kapitel 5 eine genaue Zieldefinition für den in dieser Studienarbeit zu erstellenden Prototyp definiert wird, soll in diesem und in den folgenden Kapiteln die Basis für diese Zieldefinition geschaffen werden. In diesem Kapitel werden zunächst die musikalischen Grundlagen betrachtet; Es werden die Fragen geklärt: Was ist gute Musik? Und woran erkennt man sie? Am Ende des Kapitels soll also herausgestellt sein anhand welcher Faktoren die Musik bewertet werden kann.

*Was ist gute Musik?* - Auf den ersten Blick eine banale Frage. Jedoch ohne eindeutige Antwort: Stellt man diese Frage zehn Probanden erhält man mit hoher Wahrscheinlichkeit zehn verschiedene Antworten. Ein Auszug aus einem Forum<sup>1</sup>, in dem das Thema diskutiert wurde, belegt das:

- *„für mich persönlich bedeutet gute musik, dass sie von herzen kommt, dass gefühle und gedanken drin stecken. wenn musik nur gemacht wird um damit geld zu verdienen kann es sich unter umständen gut anhören, aber verliert trotzdem an glaubwürdigkeit und ehrlichkeit.“* (lioness)
- *„Gute Musik ist für Musik die mich innerlich berührt.“* (Steffen16)
- *„Gute Lieder sind gut arrangiert, gehen mindestens 4 Minuten und haben sowohl starke Strophen, als auch einen guten Refrain. “* (Lady of ice)
- u.v.m.

Eine wissenschaftliche Definition für *gute Musik* kann nicht gefunden werden, denn bereits der Begriff *gut* weist auf eine menschlich, subjektive Empfindung hin. Betrachtet man die verschiedenen Antworten auf die Frage „Was ist gute Musik?“, fällt jedoch eines auf: Zwar sind alle Antworten letztendlich unterschiedlich, jedoch ist die Begründung meist auf einen Nenner zurückzuführen – Musik ist gut, wenn sie emotional berührt. Diese emotionale Verbundenheit kann durch den Inhalt, den Sound oder durch den Interpret

<sup>1</sup>Siehe <http://www.uni-protokolle.de/foren/viewt/115551,0.html>. Zuletzt abgerufen am 01. Juni 2011

erzeugt werden. So lässt sich ein Musikstück auch in drei Komponenten teilen. Ein (gehörtes) Lied besteht aus

- einer Komposition,
- einem Arrangement bzw. einer Interpretation und
- optional aus einem Text.

Unterbewusst werden alle drei Komponenten bewertet. Werden alle drei Komponenten als *gut* klassifiziert, gefällt dem Hörer das Lied. In den folgenden Abschnitten sollen die drei Komponenten nun genauer beleuchtet werden.

## 2.1 Die Komposition

Die Komposition eines Musikstückes kann mit einer Bauanleitung verglichen werden. Sie enthält Anweisungen, die beschreiben wie das Musikstück vorgetragen werden soll. Dabei können beispielsweise Tonhöhen und Tonlängen, das Tempo, die Artikulation und die Dynamik des Stückes festgelegt werden. Außerdem kann der Komponist die zu benutzenden Instrumente festlegen. Tonhöhen und Tonlängen ergeben die Melodie eines Stückes - Die Melodie ist der Grundstock jeder Komposition, alle anderen Parameter sind optional, müssen also nicht in der Komposition enthalten sein. Sie werden deswegen auch als sekundäre, Tonhöhe und Tonlänge als primäre, Parameter bezeichnet.

Die Komposition, speziell die Lage der Töne, obliegt dabei musikalischen Regeln. So steht eine Melodie in der Regel in einer Tonart. Alle Töne der Melodie sollten sich in dieser Tonart befinden. Andere Töne werden gewöhnlich als *schief* bzw. *unpassend* empfunden. Allerdings gibt es Ausnahmen, so kommen vor allem in anspruchsvolleren Stücken auch Töne vor, die nicht zur Tonart des Liedes gehören. Einige Stücke enthalten auch komplette Tonart-Wechsel. Eine Komposition besitzt jedoch (fast ausnahmslos) ein tonales Zentrum. Das tonale Zentrum ist ein Ton, um dieses Zentrum bauen sich die Harmonien auf. Dadurch wird zum Beispiel Spannung erzeugt oder gelöst. In den allermeisten Fällen ist das tonale Zentrum der Grundton der Tonart.

Auch die Definition der Tonlängen obliegt genauen mathematischen Regeln. So hat jeder Ton<sup>2</sup> eine relativ zum Tempo festgelegte Länge.

Nähere Informationen zu den musiktheoretischen Grundlagen finden sich in diverser Literatur und sollen hier nicht weiter erläutert werden.

Ähnlich wie Gestaltgesetze, welche Regeln zur Darstellung von gut aussehenden Objekten bzw. gut aussehenden Oberflächen Interaktiver Systeme beschreiben [zu Bexten (2008)], beschreiben diese musikalischen Regeln wie gut klingende Musik aufgebaut sein muss. Die Regeln wurden im Laufe der letzten Jahrhunderte auf Basis von Erfahrungswerten über die menschliche Wahrnehmung aufgestellt.

Ein weiterer interessanter Aspekt im Zusammenhang mit der Komposition wurde durch den argentinischen Physiker Damian Zamette bewiesen. Er stellt fest, dass Melodie-Folgen auf gleichen logisch, mathematischen Gesetzen basieren wie natürliche Sprache. Zamette wandte auf die Verteilung der Töne eines Musikstücks das so genannte Zipf'sche Gesetz an. Dieses, von dem amerikanischen Linguisten George Kingsley Zipf in den 1940er Jahren entwickelte Gesetz, beschreibt die Häufigkeitsverteilung der Wörter in einem Text: Das zweithäufigste Wort kommt demnach etwa halb so oft vor wie das häufigste, das dritthäufigste ein Drittel mal so oft und so weiter. Dieses Gesetz gilt näherungsweise für fast alle Sprachen – und für Musik, wie Zamette zeigen konnte. ([Dewald (2004)])

## 2.2 Das Arrangement

Ist die Komposition musiktheoretisch richtig und wohlklingend, obliegt es zum Großteil der Interpretation bzw. dem Arrangement, ob dem Zuhörer das Lied gefällt oder nicht. Das Arrangement setzt die „Bauanleitung“ in die Tat um. Dabei bleiben dem Künstler verschiedenste Variationsmöglichkeiten. Beispielsweise kann er sekundäre Parameter der Komposition nach seinem Geschmack anpassen und verändern.

---

<sup>2</sup>Hinweis: Es wird in den Ausführungen nicht zwischen Ton und Pause unterschieden.

So können klangmäßig völlig unterschiedliche Werke entstehen. Vergleicht man beispielsweise die Interpretation des Stückes *Mad World* von *Michael Andrews* mit der Interpretation der Band *Die Toten Hosen*, stellt man fest, dass die Stücke sehr unterschiedlich klingen.

### 2.3 Der Text

Der Text ist neben der eigentlichen Musik, die durch Komposition und Arrangement dargestellt wird, vergleichbar mit einer Meta-Information. Als Meta-Informationen bezeichnet man Daten, die Informationen über andere Daten enthalten. Ein Liedtext enthält den Inhalt, die Geschichte eines Liedes. Diese Geschichte wird bereits durch die Melodie, die Stimmung des Liedes und letztendlich auch durch die Interpretation verkörpert. Der Text stellt den Inhalt jedoch „unkodiert“, in Form von natürlicher Sprache dar.

Dass der Text für die Wahrnehmung bzw. die Bewertung eines Liedes durchaus von Bedeutung ist, lässt sich vom Erfolg von Liedern in anderen Sprachräumen ableiten. So haben deutschsprachige Songs nur selten Erfolg im Ausland. Jedoch gibt es auch eine große Menge von Ausnahmen, sodass man daraus schließen kann, dass das Verstehen und damit das Bewerten des Textes nicht zwingend für das *Gefallen* eines Musikstückes nötig ist. Dennoch kann der Text großen Einfluss auf die Bewertung haben. So finden Lieder von *Herbert Grönemeyer* sehr viele Fans, obwohl seine Musik und sein Gesang im Allgemeinen nicht als musikalisch herausragend gelten - seine Lieder erregen vor allem durch ihren Text das Empfinden der Zuhörer.

## 3 Verfahren der künstlichen Intelligenz

An dieser Stelle soll keine „klassische“ Einführung in die Thematik der künstlichen Intelligenz (im Folgenden KI) gegeben werden. Bei Interesse können Grundlagen der KI zum Beispiel in [Lämmel und Cleve (2008)] oder [Russel und Norvig (2004)] erarbeitet werden. Vielmehr soll ein kurzer Überblick gegeben werden welche Möglichkeiten sich heute durch die Nutzung von KI ergeben. Anschließend werden einige grundlegende Techniken der KI betrachtet, welche im Hinblick auf die Verwendbarkeit zur Klassifikation von Musik analysiert werden. Ziel dieses Kapitels ist es die Basis für die Auswahl, der in dieser Arbeit verwendeten KI-Techniken, ausreichend darzulegen, sodass die Entscheidungsfindung später schlüssig begründet werden kann.

Erst vor wenigen Monaten - Im Februar 2011 - hat der von IBM entwickelte Supercomputer Watson in den Medien auf sich aufmerksam gemacht <sup>3</sup> - Er besiegte in der US-Spielshow *Jeopardy!* zwei menschliche Quiz-Könige. Watson ist in der Lage Daten aller Art auszuwerten und daraus logische Rückschlüsse zu ziehen; Fragen an ihn können in natürlicher menschlicher Sprache (Englisch) gestellt werden. Dabei agiert der Computer so schnell und genau wie ein menschlicher Experte. Da das Verhalten Watsons kaum von menschlichen Verhalten zu unterscheiden ist, kann man zweifelsohne von einer intelligenten Maschine sprechen – in der Tat vereint Watson den aktuellen Stand der Forschung aus verschiedenen Teilgebieten der künstlichen Intelligenz. Einige davon sind auch für die in dieser Arbeit betrachtete Aufgabe interessant:

- **Spracherkennung/ Sprachverstehen.** Spracherkennung ist dabei die einfachere Disziplin. Sie wird bereits von Smartphones und anderen Geräten des Alltags beherrscht. Bei der Spracherkennung geht es darum das Gesprochene schriftlich wiederzugeben, um es dann als Datum nutzen zu können – Zum Beispiel zum Starten eines Anrufes bei einem Smartphone.  
Soll ein System auf eine gestellte Frage sinnvoll bzw. intelligent antworten, ist es nötig die Sprache und die Bedeutung des Inhalts zu verstehen. Dies ist schwierig, da menschliche Sprache oft Mehrdeutigkeiten enthält und nicht alle Tatsachen und Zusammenhänge explizit erwähnt werden. Um die Mehrdeutigkeiten zu erkennen

---

<sup>3</sup> Zum Beispiel Spiegel Online [ics (2011)]

und die Tatsachen inferieren zu können, muss das System nicht nur sprachliches Wissen zur Verfügung haben, sondern auch außersprachliches Wissen wie Welt- und Situationswissen. Watson hat durch die Beantwortung der komplexen Fragen der *Jeopardy!*-Show bewiesen, dass es mit der heutigen Technik und Rechengeschwindigkeit für Maschinen möglich ist Sprache zu verstehen – und zwar genauso schnell wie ein Mensch.

Die Erkennung von Sprache kann durchaus mit der Erkennung bzw. der Klassifizierung von Musik verglichen werden. Schließlich beinhaltet auch jedes Musikstück Inhalt, welchen es ausdrücken soll. Dabei steht der Inhalt bzw. die Aussage eines Musikstückes oft im Zusammenhang mit dessen Komponist bzw. dessen Interpret. Ein intelligentes System könnte auch hier Zusammenhänge erkennen und diese für die Entscheidungsfindung einbeziehen.

- **Information Retrieval/ Suchstrategien.** Basierend auf einer riesigen Wissensbasis muss Watson innerhalb kürzester Zeit Fakten und Tatsachen finden. Hierfür sind spezielle Suchalgorithmen notwendig. Die Analyse und die Erarbeitung von Suchstrategien und Algorithmen zum Lösen eines Problems zählen seit mehreren Jahrzehnten als Teilgebiet der KI. Dabei unterscheidet man grundsätzliche Strategien zum Finden eines Pfades (meist den möglichst kürzesten) und Strategien zum Finden von Informationen oder möglichst ähnlichen Informationen. Derartige Algorithmen sind oft Bestandteil eines lernenden Systems, wie es auch in dieser Studienarbeit entstehen soll.
- **Wissensrepräsentation.** Eng mit dem Suchen und Finden von Wissen ist das Darstellen des Wissens verbunden. Watson besitzt eine intelligente und komplexe Struktur zur Speicherung seines Wissens, sie ist optimal an die Rahmenbedingungen (Zum Beispiel die Hardware) und an die, auf der Speicherstruktur aufzusetzenden, Suchalgorithmen angepasst.  
Die Wissensrepräsentation wird oft als Grundlage oder Kernelement der KI gesehen (Vgl. [Lämmel und Cleve (2008)], S. 30; oder [Russel und Norvig (2004)] S. 397). So ist sie auch für diese Arbeit von großer Bedeutung, denn auch in diesem Projekt muss die Frage geklärt werden, wie erlerntes Wissen – in diesem Fall Musik, die das System “mag” – dargestellt und letztendlich gespeichert werden soll.

Watson soll in dieser Arbeit nicht weiter detailliert betrachtet werden, für weitere Informationen dient beispielsweise [IBM (2011)]. Neben Watson gibt es noch viele weitere Beispiele für intelligente Maschinen. So gibt es Systeme zum autonomen Planen von logistischen oder zeitlichen Plänen, Systeme welche das Ausführen Ihrer Aufgaben automatisch kontrollieren (Zum Beispiel ein selbstständig fahrendes Fahrzeug) oder Systeme, welche Chirurgen bei komplizierten Operationen unterstützen (Robotik). Weitere konkrete Beispiele finden sich in [Russel und Norvig (2004)], Seite 50.

Da viele Systeme die Teilgebiete, welche anhand von Watson erläutert wurden, nutzen, können diese auch als Kerngebiete der KI bezeichnet werden. Folglich sind diese Kerngebiete auch für die Studienarbeit von Bedeutung, deswegen werden im Folgenden verschiedene grundlegende, den Gebieten entsprechende, KI-Techniken erläutert. Ergänzt wird die Reihe durch neuronale Netze. Neuronale Netze dienen ebenfalls oft als Grundlage intelligenter Systeme. Zusätzlich soll bereits kurz argumentiert werden, inwiefern die verschiedenen Techniken zur Klassifizierung von Musik einsetzbar sind.

### 3.1 Darstellung von Wissen mit Hilfe von Logik

An dieser Stelle soll nicht auf die Grundlagen der Logik eingegangen werden, jedoch sind grundlegende Kenntnisse der Logik für Informatiker für das Verstehen der folgenden Ausführungen nötig. Gegebenenfalls können die Grundlagen in anderer Literatur, beispielsweise [Schöning (2000)], nachgeschlagen werden.

Die Logik ist eine sehr einfache, aber dennoch sehr ausdrucksstarke Form der Wissensrepräsentation. Mit Hilfe von Logik können Aussagen und Regeln eindeutig formuliert werden. Das macht es Computerprogrammen möglich mit Hilfe von Resolution Beweise zu automatisieren, also Wissen auch verarbeiten und nutzen zu können. An dieser Stelle sei beispielsweise auf die Programmiersprache PROLOG verwiesen.

Wie bereits im vorherigen Kapitel bei der Analyse von Musik festgestellt wurde, basiert auch Musik zu einem Teil auf klar definierten Regeln. So wäre es zum Beispiel vorstellbar, dass mit Hilfe der Logik ein Musikstück auf seine musik-theoretische Richtigkeit geprüft wird. Hierfür könnten Regeln aufgestellt werden, die Beispielsweise die Stimmigkeit ver-

## Kapitel 3 Verfahren der künstlichen Intelligenz

---

wendeter Noten einer Melodie prüfen - Ein Beispiel hierfür ist im Folgenden Listing als Prolog-Programm dargestellt.

```
stimmig(ton1, ton2, ton3, ton4) :=
    gehoert_zu(ton1, ton2),
    gehoert_zu(ton1, ton3),
    gehoert_zu(ton1, ton4).
gehoert_zu(tonart, ton) := tonart is (ton - 0).
gehoert_zu(tonart, ton) := tonart is (ton - 2).
gehoert_zu(tonart, ton) := tonart is (ton - 4).
gehoert_zu(tonart, ton) := tonart is (ton - 5).
gehoert_zu(tonart, ton) := tonart is (ton - 6).
gehoert_zu(tonart, ton) := tonart is (ton - 8).
[...]
```

---

Listing 1: Musikregeln mit Prolog überprüfen

Im Listing überprüft die Regel `stimmig`, ob die ihr übergebenen Töne zusammen passen. Es wird davon ausgegangen, dass `ton1` der Grundton ist. Im Backtracking-Verfahren werden dann die einzelnen `gehoert_zu`-Regeln probiert. Diese Regeln prüfen, ob ein Ton 0, 2, 4, 5, 7, 9 oder 11 Halbtonschritte vom Grundton entfernt ist und somit zum Grundton passt. Dieses Beispiel erhebt natürlich keinen Anspruch auf Vollständigkeit - Beispielsweise werden Töne nicht zugelassen, wenn sie nicht in der gleichen Oktave sind - Es zeigt jedoch wie man Logik einsetzen kann, um Musik zu klassifizieren und automatisiert zu analysieren.

Jedoch hat die Logik auch Ihre Grenzen: Da sie nur feste Regeln befolgen kann, ist es nicht möglich vage Aussagen zu treffen. In der Logik gibt es nur 0 und 1. So ist es nicht möglich ihr einen "Musikgeschmack" beizubringen. Außerdem kann eine Melodie musiktheoretisch durchaus richtig und vollkommen sein und dennoch nicht besonders schön klingen. Will man neben der Melodie auch noch Interpretation und/ oder Text analysieren, stößt man mit der Logik sehr schnell auf Probleme. Die alleinige Anwendung der Logik würde also nicht ausreichen, um ein menschliches Verhalten glaubhaft zu simulieren, sie kann jedoch durchaus eine Komponente eines intelligenten Systems darstellen.

### 3.2 Darstellung von Wissen in semantischen Netzen

Wie bereits auf Seite 15, bei der Vorstellung von Watson, erwähnt, finden sich zwischen dem Verständnis von Musik und dem Verständnis von Sprache durchaus parallelen. Zumal es durch die Nutzung eines Systems, welches Sprache versteht, möglich wäre, auch den Inhalt eines Liedes zu analysieren. Aber nicht nur das: Neben der Sprache könnte auch der semantische Inhalt der Komposition verstanden werden. Vor allem in klassischer Musik erzählen einzelne Instrumente oder Melodien eigene Geschichten. So werden beim musikalischen Märchen *Peter und der Wolf* von *Sergei Prokofjew* die verschiedenen Figuren durch verschiedene Instrumente dargestellt<sup>4</sup>. Aber nicht nur durch Instrumente, sondern auch durch Spielart oder Tonart-Wechsel können Geschichten modelliert werden. Beispielsweise wechselt in *Antonio Vivaldis* ersten Satz des ersten Abschnitts (*La Primavera – Der Frühling*) des Violinkonzerts *Die Vier Jahreszeiten* die Tonart von einer Moll-Tonart zu einer dominanten Dur-Tonart, wodurch ein aufkommender Frühlingssturm dargestellt wird<sup>5</sup>. Aber auch bei modernen Stücken findet man diverse musikalische Figuren - Zum Beispiel im Song *Seelentherapie* der *Toten Hosen* vom Album *Opium fürs Volk*. Hier werden am Anfang das Gefühl der Verzweiflung und der Eintönigkeit durch abwärts gezupfte Akkorde einer Moll-Tonart verstärkt.

Semantische Netze gehören zu den “am weitesten verbreiteten Schemata zur expliziten Repräsentation von Wissen in sprachverstehenden Systemen” [Sagerer (1990), Seite 85]. Vergleichbare Techniken für *musikverstehende* Systeme sind noch nicht etabliert, jedoch können wahrscheinlich ähnliche Techniken wie beim Verstehen von Sprache angewandt werden.

Semantische Netze sind vergleichbar mit Modellen der objektorientierten Programmierung, sie bestehen aus einer Menge von Knoten, die durch gerichtete und beschriftete Kanten miteinander verbunden sind. Die Knoten repräsentieren begriffliche Einheiten wie Objekte, Vorgänge, Zustände, Orte, Zeitabschnitte, Eigenschaften, Zahlen usw. Die Kanten bringen die Beziehungen zwischen diesen Einheiten zum Ausdruck.

<sup>4</sup>Weitere Informationen [http://de.wikipedia.org/wiki/Peter\\_und\\_der\\_Wolf](http://de.wikipedia.org/wiki/Peter_und_der_Wolf)

<sup>5</sup>Weitere Informationen und Hörproben [http://de.wikipedia.org/wiki/Die\\_vier\\_Jahreszeiten](http://de.wikipedia.org/wiki/Die_vier_Jahreszeiten)

## Kapitel 3 Verfahren der künstlichen Intelligenz

Es gibt verschiedene Arten von Beziehungen:

- A ist Teilmenge von B (is\_a)
- A ist Element/ Exemplar/ Instanz der Klasse B (instance\_of)
- A ist ein Teil von B (part\_of)
- B ist ein Merkmal (Attribut) von A (z. B. Farbe, Größe, Agent, Empfänger...)

Semantische Netze werden in Form eines Graphen dargestellt:

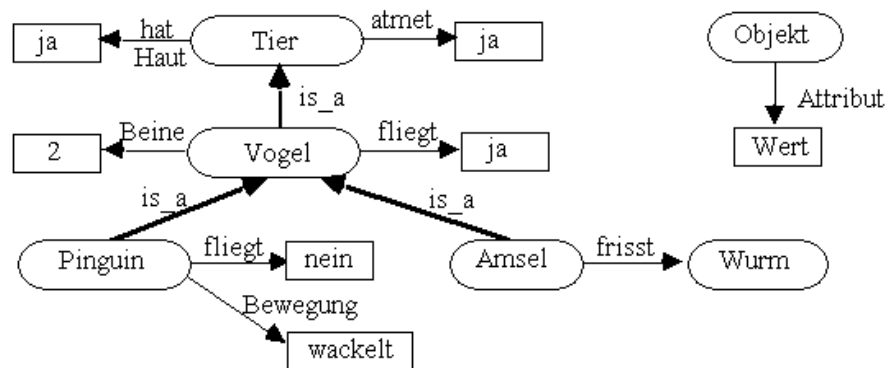


Abbildung 1: Ein Beispiel für ein semantisches Netz. [Kriz (2010)]

Um Sprache wirklich verstehen zu können benötigt das System neben dem semantischen Netz, welches als Speicherstruktur dienen kann, allerdings noch Algorithmen, welche das Netz nutzen können. Auf derartige Algorithmen soll in dieser Arbeit jedoch nicht eingegangen werden, da dies den Rahmen deutlich überschreiten würde. Für weitere Informationen sei auf [Sagerer (1990)] verwiesen. Es wird allerdings festgestellt, dass semantische Netze ebenfalls geeignet sind, um in einem System zur Klassifizierung von Musik eingesetzt zu werden.

### 3.3 Darstellung von vagen Wissen

Bei der Betrachtung der Logik als Mittel zur Darstellung von Wissen auf Seite 17 wurde festgestellt, dass der klassischen Logik durch Ihre Zweiwertigkeit Grenzen gesetzt sind. Aussagen wie

- “Der Mann ist groß.”,
- “Das Auto ist teuer.”, oder (zu dem hier besprochenen Thema passend)
- “Die Musik ist schön.”.

können mit klassischer Logik nicht formuliert werden. Denn hier kann der Wahrheitswert nicht eindeutig bestimmt werden. Sinneswahrnehmungen, wie auch das bewerten eines Musikstückes, werden im Alltag nicht quantitativ bewertet. Derartige Einschätzungen werden in der Regel abstrahiert, um dann eine qualitative Einordnung vorzunehmen. So schätzen wir eine Mahlzeit beispielsweise als *pikant*, *scharf*, *lasch* u.s.w. ein. Ebenso verhält es sich mit der Bewertung von Musik. Aussagen dieser Form werden als unscharfe Aussagen bezeichnet und zählen somit zur Gruppe des vagen Wissens. Neben unscharfen Aussagen existieren noch Aussagen wie

- “Mindestens 30% der Schüler werden später das Abitur machen.“, oder
- “Die Firma wird ihren Umsatz im nächsten Quartal um 2 Millionen Euro steigern.“.

Derartige Aussagen werden als unsichere Aussagen bezeichnet. Sie werden hier aber nicht näher betrachtet, da sie für das Thema der Studienarbeit weniger von Bedeutung sind. Schließlich sollen hier keine Vorhersagen getroffen werden, welche in aller Regel durch unsichere Aussagen formuliert sind.

Für die Darstellung von unscharfen Aussagen eignen sich sogenannte Fuzzy-Mengen ([Lämmel und Cleve (2008), Seite 99]). Grundlage der Fuzzy-Mengen ist die charakteristische Funktion. Diese Funktionen beschreiben die Zugehörigkeit einer Klasse.

## Kapitel 3 Verfahren der künstlichen Intelligenz

Die charakteristische Funktion, welche die Zugehörigkeit der natürlichen Zahlen zu den geraden Zahlen darstellt, ist in der folgenden Abbildung dargestellt.

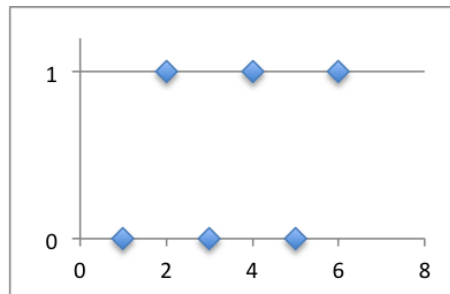


Abbildung 2: Charakteristische Funktion für die Geraden Zahlen

Zwar sind in diesem Beispiel wieder nur die Werte 1 und 0 bzw. Wahr oder Falsch verwendet wurden, jedoch lässt sich dieses Konzept nun erweitern und auf unscharfe Aussagen übertragen. Als Beispiel sollen Aussagen über das *Alter* betrachtet werden. Wann ist eine Person *alt*? Die Antwort wird sehr wahrscheinlich, je nach Alter des Befragten variieren. Jedoch kann davon ausgegangen werden, dass

- eine Person mit einem Alter von mindestens 93 von jedem Befragten als *alt* und
- ein Kind, das noch keine 3 Jahre alt ist, als *jung* bzw. *nicht alt*

bezeichnet wird. Werden diese Annahmen nun auf die charakteristische Funktion abstrahiert, entstehen die folgenden Aussagen:

- Personen mit einem Alter unter 3 Jahren sind nicht alt. Erhalten also einen Zugehörigkeitswert von 0.
- Personen mit einem Alter von 93 oder mehr sind alt. Erhalten also einen Zugehörigkeitswert von 1.
- Personen mit einem Alter von 3 bis 92 bekommen demzufolge einen Zugehörigkeitswert zwischen 0 und 1.

Die Funktion, die nun beschreibt, mit welchem Wert eine Person die Eigenschaft *alt* erfüllt sieht dann in etwa so aus, wie in der folgenden Abbildung dargestellt.

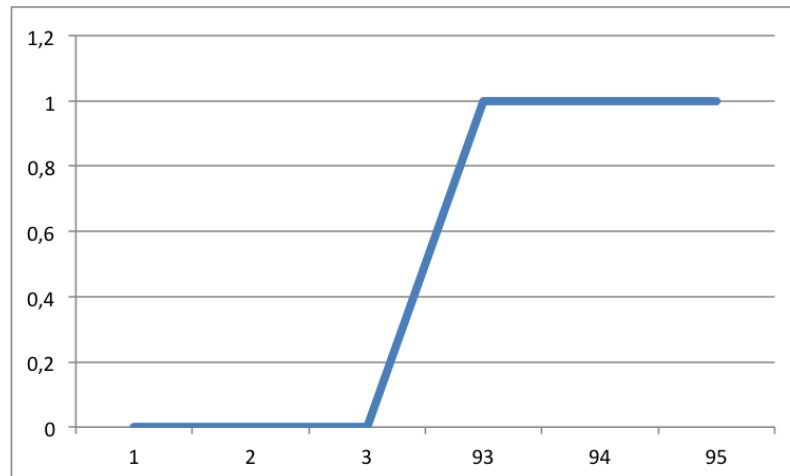


Abbildung 3: Charakteristische Funktion für *alt*

Analog können für andere unscharfe Aussagen derartige Zugehörigkeitsfunktionen erstellt werden. Jedoch nicht für alle. Betrachten wir das Beispiel Musik, fällt es schwer Werte für das Attribut *gut* zu finden, die auf jeden Fall einen Zugehörigkeitswert von 0 oder einen Zugehörigkeitswert von 1 haben. Das Problem liegt sogar schon im Schritt davor: Wie soll *gut* quantitativ bewertet werden, um es in eine Skala zu packen? Für bereits existierende Musikstücke kann hier eventuell die Chart-Position gewählt werden. Dies nützt aber einem System nichts, was neue Musikstücke bewerten soll. Schlussendlich scheint der Einsatz der Fuzzy-Mengen zur Klassifizierung von Musikstücken ungeeignet.

### 3.4 Problemlösung mittels Suche

Wie bereits auf Seite 16 erwähnt, ist die Suche in vielen intelligenten Anwendungen enthalten. Viele Probleme lassen sich offensichtlich auf Suche zurückführen. Aufgaben wie beispielsweise die Tourenplanung oder die Zeitplanung können mittels Suche gelöst werden. Aber auch andere KI-Techniken bauen oft auf verschiedenen Such-Algorithmen auf. So benötigt beispielsweise PROLOG einen geeigneten Suchalgorithmus, um in der vorhandenen Wissensbasis bzw. den gegebenen Regeln einen passenden Beweis zu finden. Die Suche wird in der KI sehr vielseitig eingesetzt - Letztendlich kann theoretisch

## Kapitel 3 Verfahren der künstlichen Intelligenz

---

jedes Problem für welches kein Algorithmus existiert durch eine Suche gelöst werden - Praktisch ist die Einsatzfähigkeit jedoch durch die Größe des Suchraumes begrenzt. Ist dieser zu groß, kann eine Suche nicht mehr sinnvoll eingesetzt werden.

Charakteristisch ist jedes Such-Problem durch die folgenden Eigenschaften ausgezeichnet ([Lämmel und Cleve (2008), Seite 115]):

- Es gibt einen definierten *Ausgangszustand*
- Der gewünschte *Zielzustand* ist vorgegeben
- Die möglichen *Aktionen* zum Übergang von einem Zustand zum nächsten sind bekannt.

Eine Übersicht über die unterschiedlichen Suchalgorithmen, deren Eigenschaften und Einsatzgebiete findet sich in [Russel und Norvig (2004)].

Für ein Musik klassifizierendes Systems könnte das Durchsuchen einer Wissensbasis interessant sein. Viele lernende Systeme, wie Experten- oder Wissensbasierte-Systeme besitzen große Wissensbasen, in denen ihr Wissen gespeichert ist. Sollen die Systeme ein Problem lösen, müssen sie in Ihrer Wissensbasis nach geeigneten Informationen suchen. Je nach Datenstruktur, Art der Daten und des Problems können verschiedenste Suchalgorithmen angewandt werden, um Informationen zu finden. Sollte ein derartiges System genutzt werden, wird die Suche also auch für dieses Projekt eine Rolle spielen.

### 3.5 Wissensbasierte Systeme

Nachdem in den bisherigen Abschnitten KI-Verfahren zur Wissensrepräsentation und mit der Suche der wichtigste Bereich der Verarbeitung von Wissen angesprochen wurde, soll nun der allgemeine Aufbau von Systemen beschrieben werden, welche diese Verfahren einsetzen. Wissensbasierte Systeme, oft auch mit dem Begriff der Expertensysteme gleich gesetzt, sind intelligente Informationssysteme, sie machen Wissen mit Hilfe von Methoden der Wissensrepräsentation und Wissensverarbeitung für einen bestimmten Anwendungsfall nutzbar.

Sowohl zur Repräsentation von Wissens, als auch für das Ziehen von Schlussfolgerungen können unterschiedliche Ansätze verfolgt werden:

- **Fallbasierte Systeme.** In diesen Systemen werden eine Menge von bereits vorhandenen Fällen, inklusive deren Lösungen, gespeichert. Soll das System ein neues Problem lösen, muss in den vorhandenen Fällen nach einem möglichst ähnlichen Fall gesucht werden. Nun wird versucht die Lösung derartig zu modifizieren, dass sie auch für den neuen Fall gültig ist. Wird die Lösung durch einen Benutzer als richtig eingeschätzt, kann dieser Fall ebenfalls in das System aufgenommen werden, sodass das System dazu lernt.
- **Regelbasierte Systeme.** Im Gegensatz zu den Fallbasierten Systemen werden hier keine konkreten Fälle gespeichert, sondern ausschließlich (Logik-)Regeln. Diese Regeln stellen die allgemeinen Vorschriften dar und sind nicht an besondere Begebenheiten angepasst. Das System muss anhand des zu bearbeitenden Falls die entsprechenden Regeln finden und anwenden, um eine Lösung zu finden. Durch die Nutzung von vorhandenen Regeln in neuen Fällen, können neue Regeln inferiert werden. Diese können, nach Bestätigung durch einen Benutzer, ebenfalls in die Regelbasis aufgenommen werden.
- **Entscheidungsbäume.** Mittels Entscheidungsbäumen können Datenmengen automatisiert in Gruppen eingeteilt werden. Dafür untersucht ein Algorithmus eine gegebene Datenmenge auf seine Attribute. Die Attribute stellen im Ergebnis-Baum die Äste dar, ein Attribut, welches den Zielwert der Untersuchung darstellt, ist im Baum als Blatt dargestellt. Der Algorithmus versucht möglichst kleine Bäume zu finden, bei denen möglichst viele Datensätze richtig ausgewertet werden können. Solche Systeme finden zum Beispiel in Data Mining Systemen ihren Einsatz [Vgl. Bollinger (2011), Seite 44].

## Kapitel 3 Verfahren der künstlichen Intelligenz

Alle Arten von Wissensbasierten Systemen haben jedoch einen grundlegend gleichen Aufbau, welcher in der folgenden Abbildung dargestellt ist.

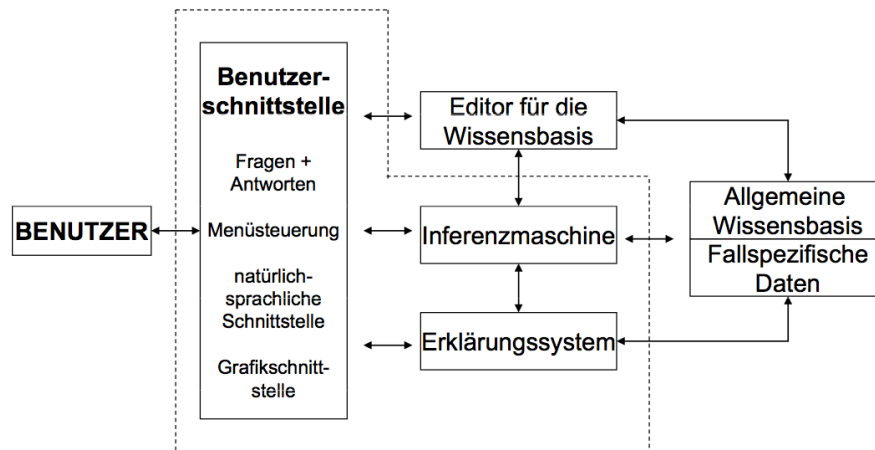


Abbildung 4: Aufbau eines Wissensbasierten Systems [G.F.Luger (2001)]

Die Wissensbasis speichert das gesamte Fakten- bzw. Regelwissen, vergleichbar mit einer Datenbank, ab. Die Wissensbasis kann dabei, wie in der Darstellung, in zwei Bereiche gegliedert sein: In eine allgemeine und eine fallspezifische Wissensbasis. Es ist auch möglich, dass nur eine der beiden Arten vorhanden ist. Fallspezifisches Wissen ist Wissen, welches in Verbindung mit ähnlichen Fällen steht, die bereits vom System bearbeitet wurden oder von Beginn an als Grundlage der Wissensbasis dienen – beispielsweise bei Fallbasierten Systemen. Derartige ähnliche Fälle können in der Wissensbasis mit Hilfe von Ähnlichkeitsmaßen und passenden Algorithmen gefunden werden (Siehe dazu auch [Reichardt (2010), Seite 170]). Die allgemeine Wissensbasis enthält Regeln, welche die allgemeinen Vorschriften in Form von Logik enthalten (Regelbasierte Systeme). Mit Hilfe eines Eingabesystems, in der Abbildung *Editor für Wissensbasis*, ist es möglich die Wissensbasis manuell zu bearbeiten.

Neben der Wissensbasis ist die Inferenzmaschine der wichtigste Bestandteil eines wissensbasierten Systems. Die Inferenzmaschine ist der Teil des Systems, der anhand der Fakten und Regeln der Wissensbasis neue Aussagen schlussfolgern kann und somit auch automatisch die Wissensbasis um neue Aussagen erweitern kann. Die angewendeten Algorithmen können dabei völlig unterschiedlicher Natur sein. Das Erklärungssystem kann die Inferenzmaschine ergänzen.

Diese Komponente dient dazu, dem Benutzer die Begründung bzw. die Lösungsfindung für eine, von der Inferenzmaschine erzeugte, Aussage zu liefern.

Weiterhin besitzt jedes Wissensbasierte System eine Benutzerschnittstelle, über die es mit der Außenwelt bzw. dem Anwender kommunizieren kann. Diese Schnittstelle kann über eine übliche Software-Oberfläche realisiert sein, kann aber auch eine sprachverstehende und sprechende Schnittstelle in Form eines Roboters sein.

Wird ein intelligentes System auf Basis der klassischen Verfahren der KI erzeugt, resultiert in der Regel ein wissensbasiertes System, in dem sich die genannten Bestandteile wieder finden. So sind diese Bestandteile und der generelle Aufbau eines wissensbasierten Systems auch für die Klassifizierung von Musik durchaus interessant.

### 3.6 Künstliche neuronale Netze

Der Ansatz der künstlichen neuronalen Netze unterscheidet sich grundsätzlich von den bisher genannten Techniken. Die zuvor erläuterten Methoden verfolgen alle einen phänomenologischen Ansatz; Sie versuchen stets das Ergebnis bzw. die Ausgabe intelligent erscheinen zu lassen. Im Hintergrund werden die Probleme jedoch stets mit Methoden der klassischen Informatik gelöst. So könnte eine Wissensbasis eines PROLOG-Programms auch in einer gängigen Programmiersprache programmiert werden und gleiche Ausgaben erzeugen. Auch das Lösen von Problemen mit Hilfe von Suchalgorithmen auf der Basis von Wissensbasen obliegt im Hintergrund keiner menschen-ähnlichen Intelligenz. Letztendlich weiß das Programm nicht, ob es Musik oder Rezepte klassifiziert - Es verarbeitet schlicht die Daten, ohne sie wirklich zu verstehen - Auch wenn die Ausgabe dieser Systeme anderes vermuten ließe.

Seit Mitte des letzten Jahrhunderts [Ritter u. a. (1991), Seite 25] wird mit der Forschung im Bereich der Neuroinformatik ein Bottom-up Ansatz verfolgt; Es wird versucht die Fähigkeiten und die Arbeitsweise des Gehirns nachzubilden. Dies wurde auch erst innerhalb der letzten Jahrzehnte durch Fortschritte in der Gehirnforschung möglich. Jedoch ist es aber bisher nicht möglich ein Gehirn in seiner vollen Komplexität nachzubilden - Zum einen ist das Gebiet bei weitem noch nicht ausreichend erforscht und zum anderen übersteigen die Leistungsmerkmale (wie zum Beispiel die Geschwindigkeit) des

## Kapitel 3 Verfahren der künstlichen Intelligenz

---

menschlichen Gehirns noch immer in vielen Bereichen die eines Computers - Auch wenn der Computer in manchen Disziplinen wie dem "Kopfrechnen" den Mensch um Längen schlagen kann. Grundlegende Prinzipien können jedoch bereits heute in Programmen abgebildet werden.

Grundsätzlich besteht das Gehirn aus vielen Millionen von Zellen bzw. Neuronen, die miteinander vernetzt sind. Diese Zellen tauschen untereinander Informationen über elektronische Impulse aus. Empfängt ein Neuron Signale von mehreren anderen Neuronen, produziert es durch chemische Prozesse einen "Ausgabewert" den es wiederum an andere Neuronen weiterleitet. Durch diesen, im kleinen betrachtet, einfachen Prozess werden wahrgenommene Sinnesreize verarbeitet. Durch die vielschichtige Vernetzung zwischen mehreren Millionen Neuronen ist dieser Prozess jedoch sehr komplex und insgesamt kaum durchschaubar.

Künstliche Neuronale Netze versuchen dieses Verhalten nachzubilden. Es soll möglich sein die Netze durch Trainingsszenarien zu trainieren, sodass sie sowohl in der Trainings- als auch in einer neuen Situation angemessen reagieren und somit Aufgaben ähnlich intelligent wie ein Mensch lösen können.

Neuronale Netze haben folgende Vorteile gegenüber den klassischen Techniken ([Reif und Reschenhofer (2005), Seite 7]; [Lämmel und Cleve (2008), Seite 194ff])

- Fehlertoleranz
- Generalisierungsfähigkeit
- Lernen
- Selbstorganisation

Neuronale Netze werden bereits in vielen verschiedenen Bereichen eingesetzt. Zum Beispiel zur Bild- oder Schrifterkennung, zur Steuerung von Robotern oder für Regelungsaufgaben. Außerdem wurden auch Forschungen im Bereich der Spracherkennung mittels neuronaler Netze betrieben. So kommen neuronale Netze also auch zur Analyse von Musik durchaus in Frage.

Die biologischen Grundlagen von Neuronen seien an dieser Stelle aus [Lämmel und Cleve (2008)] und [Reif und Reschenhofer (2005)] zitiert. Ausführlichere Darstellungen finden sich unter anderem in [Ritter u. a. (1991), Seite 25] oder [Zell (1997)].

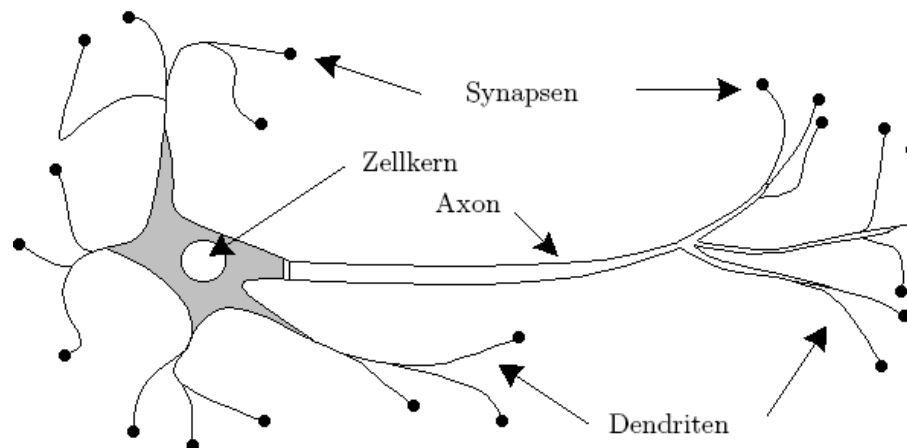


Abbildung 5: Biologisches Neuron. [Reif und Reschenhofer (2005)]

„Für das Verständnis künstlicher neuronaler Netze reicht eine einfache Vorstellung über ein biologisches Neuron [, wie in der Abbildung oben dargestellt,] aus. Die Dendriten nehmen die Erregungen, die von anderen Neuronen ausgehen, auf und leiten sie an den Zellkern weiter. Die Erregung der Zelle wird dann über das Axon an die nächsten Neuronen weitergereicht. Die Synapsen stellen den Übergang vom Axon der einen Zelle zu den Dendriten weiterer Zellen her. Ein biologisches Neuron hat mehrere tausend bis zehntausend Verbindungen zu weiteren Neuronen.“ ([Lämmel und Cleve (2008), Seite 196])

„Künstliche Neuronen sind Prozessoren, die bestimmte Eigenschaften der Signalerzeugung natürlicher Neuronen simulieren. Sie empfangen über Verbindungen die Informationen anderer Neuronen. Durch Gewichte, die für jede Verbindung vorhanden sind, wird bestimmt mit welchem Faktor die empfangenen Informationen auf die neue Signalstärke Einfluss nehmen. Die neue Stärke ist die Summe der Produkte der eingegangenen Informationen mit ihren Gewichten. Durch eine Aktivierungsfunktion wird bestimmt, wie stark das Signal an andere Neuronen weitergeschickt wird.“ ([Reif und Reschenhofer (2005), Seite 6])

Durch die Wahl der Gewichte wird die Funktionsweise des Netzes bestimmt. Ein künstliches neuronales Netz kann seine Gewichte so verändern, dass das Ergebnis verbessert wird. Dieser Vorgang wird als Lernphase des Netzes bezeichnet. Es gibt zwei verbreitete Möglichkeiten des Lernens. ([Reif und Reschenhofer (2005), Seite 7]; [Lämmel und Cleve (2008), Seite 204]).

- **Lernen mit Lehrer/ trainierbare Netze.** Für diese Methode sind zum Lernen Eingabe-Daten und für diese Eingaben entsprechende Ergebnisse bekannt. Das Lernen erfolgt dabei durch das Anlegen der Eingaben an das Netz. Initial werden zuvor zufällige Gewichte an die Verbindungen gelegt. Die vom Netz berechnete Ausgabe wird mit der erwarteten Ausgabe verglichen. Entsprechend des Fehlerwertes der Ausgabe werden dann die Verbindungsgewichte angepasst. Das Anlegen von Eingaben, Vergleichen der Ergebnisse und Anpassen der Gewichte wird solange wiederholt, bis der Fehler einen zuvor festgelegten Maximalwert erreicht.

Dieses Lernverfahren entspricht somit nicht ganz der natürlichen Vorlage, ist jedoch sehr effektiv und wird in vielen Bereichen - Praktisch in allen, in denen man sowohl Ein- als auch Ergebnisdaten zur Verfügung hat - eingesetzt.

Eine Unterform des überwachten Lernens durch einen „Lehrer“ ist das **bestärkte** Lernen. In diesem Fall kann nicht gesagt werden wie richtig bzw. falsch eine Aussage bzw. ein Ergebnis ist. Es kann nur gesagt werden, ob es richtig oder falsch ist. Das Lernen gilt in diesem Fall als etwas schwerer (Vgl. [Lämmel und Cleve (2008), Seite 204]).

- **Lernen ohne Lehrer/ nicht trainierbare Netze.** Grundsätzlich sind auch diese Netze trainierbar - Jedoch erfolgt dieses Lernen autonom. Derartige Lernverfahren werden genutzt, wenn die Ergebnisse zuvor nicht bekannt sind, zum Beispiel bei sogenannten Clusterungen. Clusterungen sind Verfahren bei denen eine Menge von Daten nach vorher nicht bekannten Merkmalen in ebenfalls zuvor nicht bekannte Gruppen eingeteilt werden kann, es werden automatisch Klassifizierungsmöglichkeiten gefunden und auf die Daten angewendet. Diese Verfahren kommen unter anderem in Data Mining Systemen zum Einsatz.

Als nicht trainierbar bzw. unbewacht lernende Netze gelten auch solche, welche ihre Ausgabe direkt wieder an die Eingabe anlegen. Diese Netze obliegen oft auch anderen Paradigmen als den der natürlichen neuronalen Netze. Sie werden zum Beispiel genutzt um physikalische Sachverhalte zu simulieren, so erlernen Sie ein Verhalten nicht, sondern sie schwingen ein - Irgendwann verändert sich die Ausgabe nicht mehr.

In beiden Fällen werden in jedem Lernzyklus die Gewichte entsprechend geändert. Neben dem genutzten Lernverfahren, wird auch oft die Architektur von Netzen als Merkmal zur Einteilung von Neuronalen Netzen gewählt. Dabei spiegeln die Architekturen aber auch oft die genutzten Lernverfahren wieder bzw. sind für diese Lernverfahren optimiert. Man unterscheidet:

- **Vorwärtsgerichtete neuronale Netze.** Diese Netze bestehen aus einer beliebigen Anzahl von Schichten. Jedoch mindestens aus einer Ein- und einer Ausgabeschicht. Dabei ist jedes Neuron einer Schicht mit allen Neuronen der Folgeschicht verbunden. Netze dieser Art werden immer durch ein überwachtes Lernen, meist durch den sogenannten Backpropagation-Algorithmus, trainiert, weswegen sie auch oft als Backpropagation-Netze bezeichnet werden. Typische Anwendungen derartiger Netze sind die
  - **Klassifizierung von Daten.** Zum Beispiel um Kunden in Gruppen einzuteilen (Kreditwürdigkeit von Bankkunden) oder auch zur Qualitätskontrolle anhand von Merkmalen. (Vgl. [Lämmel und Cleve (2008)])
  - **Mustererkennungen.** Sie sind Klassifizierungsproblemen sehr ähnlich und können somit auch mit Hilfe von Netze dieser Architektur gelöst werden.
  - **Steuerung.** In der Robotik werden derartige Netze eingesetzt, um das Verhalten von Robotern zu steuern. (Vgl. [v. Lienen (2002)]; [Lämmel und Cleve (2008), Seite 228])
- **Partiell rückgekoppelte Netze.** Diese Architektur ist eine Erweiterung der vorwärtsgerichteten Netze. Das normale vorwärtsgerichtete Netz wird dabei um einen Rückkoppelungs-Mechanismus erweitert. Ein Beispiel für ein solches Netz ist das

*Jordan-Netz.* Das Jordan Netz besitzt sogenannte Kontext-Neuronen, welche die Ausgabe wieder an die Eingabe leiten. Eine schematische Darstellung findet sich in Abbildung 7. Durch diese Architektur wird das Netz um eine Art Erinnerungsvermögen erweitert, denn durch diese Rückkoppelung werden neue Eingaben und deren Ergebnisse stets durch die vorangegangenen Ergebnisse beeinflusst. Dies ist sinnvoll, wenn ein Netz zur Vorhersage von Werten eingesetzt werden soll. Partiiell rückgekoppelte Netze werden somit zum Beispiel zur Prognose von Aktienverläufen genutzt.

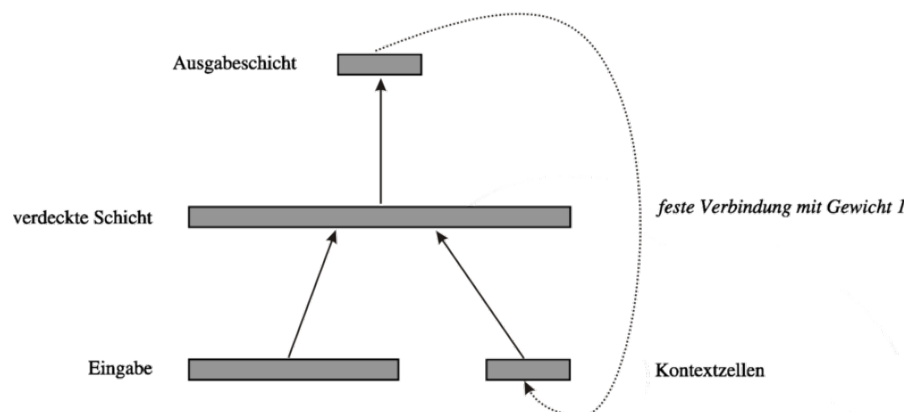


Abbildung 6: Schematische Darstellung des Jordan-Netzes. [Elmenreich (2011)]

- **Selbstorganisierende Karten.** Liegen für ein Problem keine Eingabedaten und zugehörige Ergebnisse vor, muss die Netzarchitektur entsprechend anders gestaltet werden. Ein Beispiel für eine alternative Architektur stellen die selbstorganisierenden Karten dar. Sie werden auch, nach ihrem Entwickler, als Kohonen-Netze oder Kohonen-Karten bezeichnet. Netze dieser Architektur lernen nicht überwacht, sondern finden autonom Lösungen. Eine typische Anwendung dieser Netze ist das bereits auf Seite 30 erwähnte Clustering.

Die Kohonen-Netze orientieren sich bei Ihrer Funktionsweise noch mehr am natürlichen Vorbild als die Backpropagation-Netze. Wie im menschlichen Gehirn nutzen selbstorganisierende Karten die räumliche Anordnung und die Nachbarschaftsbeziehung zwischen den Neuronen aus. Die Architektur besteht aus zwei Schichten: Einer *Eingabe-* und einer *Kartenschicht*. Jedes Neuron der Eingabeschicht ist mit jedem Neuron der Kartenschicht verbunden. Auch die Kartenschicht ist untereinander

vollständig vernetzt. Das Training eines solchen Netzes erfolgt durch das mehrmalige Anlegen der Muster in ungeordneter Reihenfolge. Im Laufe der Zeit bilden sich zu jedem Muster gewisse Erregungsfelder heraus. Weitere Informationen über den Lernalgorithmus und Kohonen-Netze im Allgemeinen finden sich in [Reif und Reschenhofer (2005), Seite 9]. Eine schematische Skizze der Architektur ist in der folgenden Abbildung dargestellt.

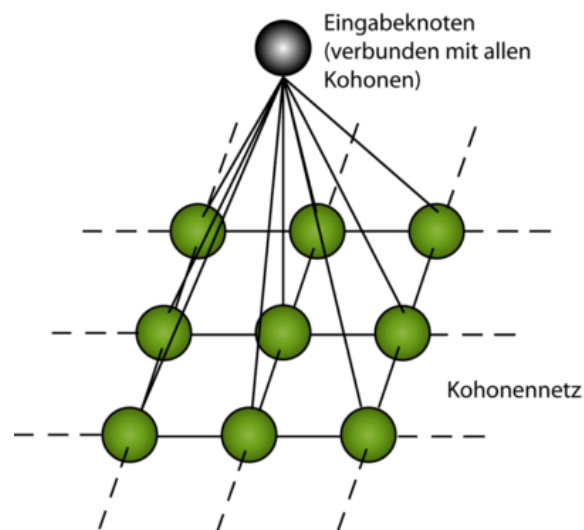


Abbildung 7: Schematische Darstellung des Kohonen-Netzes. [Reif und Reschenhofer (2005), Seite 10]

- Neuronales Gas.** Diese Form der neuronalen Netze ist eine aktuelle Weiterentwicklung der selbstorganisierenden Karten. Streng genommen ist das neuronale Netz kein Netz im eigentlichen Sinne, da die Neuronen nur mit der Eingabeschicht, jedoch nicht untereinander verbunden sind. Für viele Anwendungen von neuronalen Netzen besteht ein Problem darin, die geeignete Anzahl von Neuronen zu finden. Neuronales Gas behebt dieses Problem, da es in der Lage ist dynamisch zu wachsen (Vgl. [Fritzke (1998), Seite 74]). So kann neuronales Gas effektiver für die gleichen Aufgabengebiete wie selbstorganisierende Karten eingesetzt werden. Ebenfalls findet es auch in der Bildverarbeitung, sowie in der Spracherkennung Verwendung. Da das Feld der Neuronalen Netze jedoch noch relativ neu ist, findet sich in diverser Literatur noch kein umfassender Überblick. Es sei jedoch auf [Fritzke (1998)], [Kriesel (2005)] und [Lämmel und Cleve (2008)] verwiesen, die dieses Thema beleuchten.

## Kapitel 3 Verfahren der künstlichen Intelligenz

---

In den verschiedenen Publikationen zum Thema der neuronalen Netze lassen sich auch andere Einteilungen und weitere Architekturen finden. Grundlegend bildet die hier dargestellte Einteilung einen guten Überblick über die existierenden Ansätze. Letztendlich sind alle Architekturen mit der Backpropagation-Architektur oder dem Aufbau selbstorganisierender Karten verwandt - neuronales Gas und partiell rückgekoppelte Netze sind Beispiele für derartige Weiterentwicklungen. Sie wurden jedoch explizit erwähnt, da partiell rückgekoppelten Netze weitere Einsatzgebiete erschließen und neuronales Gas den aktuellen Stand der Forschung im Gebiet der neuronalen Netze repräsentiert.

Betrachtet man die unterschiedlichen Einsatzgebiete, kommt ein vorwärtsgerichtetes Backpropagation-Netz am ehesten für den Einsatz in einem Musik-klassifizierenden-System in Frage. Schließlich ist die die Einordnung von Musik zu einer Güteklasse durchaus mit der Klassifizierung von Daten bzw. mit der Qualitätskontrolle von Daten zu vergleichen - Auf Seite 31 wurden diese Beispiele bereits als Einsatzgebiete von vorwärtsgerichteten Netzen erwähnt. Die anderen Architekturen dienen durch ihrer Lernverfahren und ihren Aufbau eher zur Lösung anders ausgeprägter Aufgaben, wie beispielsweise der Klassifizierung von Daten, ohne Kenntnis über mögliche Einteilungen. Allenfalls neuronale Gase könnten auch für die Klassifizierung von Daten, wie auch Musik, genutzt werden. Der erfolgreiche Einsatz neuronalen Gases in der Spracherkennung belegt, dass neuronales Gas auch in der Lage ist typische Aufgabenfelder von Backpropagation-Netzen zu bearbeiten.

# 4 Digitale Verarbeitung von Musik

Ausgehend von den Kenntnissen über die Komponenten der Musik (Kapitel 2) und die grundlegenden Verfahren der künstlichen Intelligenz (Kapitel 3), sollen nun verschiedene Möglichkeiten der digitalen Speicherung und Verarbeitung von Musik erörtert werden. Es soll also die Frage geklärt werden mit welchen Datenformaten ein intelligentes System arbeiten kann.

Wie bereits in den voran gegangenen Kapiteln angesprochen, besitzt Musik und die Erkennung von Musik durchaus parallelen zur Erkennung von Sprache. Aus diesen Grund sollen nun auch zunächst die Verfahren der Spracherkennung kurz untersucht werden. Danach wird die Nutzung des MIDI-Formats in Betracht gezogen. Sie erlauben es Musik nach ihren einzelnen Bestandteilen, wie sie in Kapitel 2 beschrieben wurden, zu beurteilen.

## 4.1 Musikanalyse nach dem Vorbild der Spracherkennung

Alle Spracherkennungssysteme nutzen für die Untersuchung der Sprache das Frequenzspektrum der gesprochenen Sprache. Aus diesem Spektrum lassen sich die verschiedenen Frequenzbestandteile ablesen. Aus diesem digitalen Sprachsignal wird ein sogenannter Merkmalsvektor erzeugt, der dann mit Vektoren bekannter Signale (Wörter) verglichen werden kann (Vgl. [Schukat-Talamazzini (1995)]). Eine weitere Möglichkeit ist es, das Frequenzspektrum in ein neuronales Netz einzuspeisen. In der Praxis wurden hierfür sogenannte *Time Delay Neural Networks* (zu deutsch etwa *Zeitverzögertes Neuronale Netze*) eingesetzt. Diese Netze machen es möglich zeitlich voneinander abhängige Eingaben zu verarbeiten. Das Frequenzspektrum eines Musikstückes aus üblichen Audio-Aufnahmen, wie zum Beispiel mp3-Dateien, gewonnen werden. In diesen Formaten liegt das Musikstück in digitaler Form, durch Abtastung des des akustischen Signals, vor. Ähnlich wie bei der Spracherkennung könnte dieses Signal mit Hilfe der diskreten *Fourier-Transformation (DFT)* bzw. der schnellen *Fourier-Transformation (FFT)* in ein digitales Signal umgewandelt werden.

Nutzt man das Frequenzspektrum, hat man ein sehr großes Spektrum an möglichen Testdaten, da alle handelsüblichen Datenformate verwendet werden könnten. Diese Datenformate haben allerdings auch den Nachteil, dass sie bereits zu einem Teil komprimiert sind und damit nicht mehr alle Informationen des ursprünglich abgetasteten Signals besitzen. Dies könnte das Ergebnis verfälschen, es wäre aber auch denkbar, dass sich der Qualitätsverlust nicht auf ein mögliches Ergebnis der Analyse niederschlägt.

Ohne auf weitere Details der Spracherkennung und der Analyse von akustischen Signalen einzugehen, kann an dieser Stelle bereits festgestellt werden, dass digitale Signale als Input von intelligenten Systemen genutzt werden kann. Das Format eignet sich vor allem für neuronale Netze, als auch für Such-/ Vergleichsbasierte Systeme. Weitere Informationen und Details über die Verarbeitung von Sprache finden sich in [Schukat-Talamazzini (1995)].

### 4.2 Nutzung des MIDI-Formats

Ein anderer Ansatz der Verarbeitung kann mit dem *MIDI* Formaten verfolgt werden. MIDI, ausgesprochen *Musical Instrument Digital Interface*, ist ein bereits in den 80er Jahren entwickeltes Datenübertragungs-Protokoll zur Übermittlung musikalischer Steuerungsinformation zwischen elektronischen (digitalen) Musikinstrumenten oder Computern. Eine MIDI-Datei speichert im Gegensatz zu digitalen Audio-Aufzeichnungen keine akustischen Signale, sondern nur Anweisungen, welche von einem Klangerzeuger genutzt werden können, um Töne zu erzeugen. Dabei enthält die Datei Befehle wie:

- *Note-on* - Spiele Note  $x$
- *Velocity* - Anschlagsstärke  $y$
- *Note-off* - Beende das Spielen der Note  $x$

So lassen sich aus einer MIDI-Datei die Komposition, also die Melodie, sowie Teile des Arrangements, zum Beispiel die Instrumentation, Spielweise, etc., herauslesen. Es kann also als Grundlage für eine Analyse der einzelnen Bestandteile dienen. Diese Daten würden sich eignen um die Daten über ein Logik-System oder ein Such-/ Vergleichsbasiertes System auszuwerten. Ebenfalls ist die Eingabe der Daten in ein neuronales Netz denkbar.

Da MIDI ein weit verbreitetes Format, gibt es bereits viele Bibliotheken, welche die Programmierung von MIDI-verarbeitenden Systemen erleichtern. Außerdem gibt es zahlreiche frei verfügbare Musikstücke im MIDI-Format, welche als Testdaten dienen könnten. Ein Nachteil von MIDI ist, dass der Text eines Musikstückes außen vor gelassen wird. Wollte man neben der Komposition und dem Arrangement auch den Inhalt analysieren, müsste der Text getrennt in einem anderen Format - Zum Beispiel einer einfachen Textdatei - verwaltet werden. Das hat jedoch den Nachteil, dass Textpassagen nicht in Verbindung mit Melodie-Passagen gebracht werden können. Soll auch dieses Problem umgangen werden, muss ein neues Datenformat, welches Text- und MIDI-Datei kombiniert entworfen werden.

Neben dem MIDI Format gibt es anderer Formate, die ebenfalls das Prinzip der Steuersignale verwenden. So nutzt die Java API *JFugue*<sup>6</sup> beispielsweise einen sogenannten *MusicString*: Eine einfache Textdatei in welcher die Notennamen und die Notenlänge der zu spielenden Melodie einfach sequentiell notiert sind. Auch können hier Steuersignale wie die Wahl des Instrumentes erzeugt werden. Über eine etwas kompliziertere Syntax ist es auch möglich mehrere Töne zeitgleich zu spielen. Der *MusicString* unterstützt fast alle gebräuchlichen Funktionen des MIDI-Formats, hat aber den Vorteil, dass die Dateien auch menschlich lesbar sind und nicht als reine Binärdaten vorliegen. Auch die Umwandlung des Musikstückes in die Notenrepräsentation ist etwas einfacher, da hier nur die aus der Musik üblichen Notenlängen möglich sind. Im MIDI Format ist die Länge eines Tons nur als Zeit-Dauer vorgegeben. Aufgrund dieser Vorteile kann es durchaus sinnvoll sein dieses Format, statt des üblichen MIDI-Formates einzusetzen.

### 4.3 Komprimierte Musik

Ein weiterer interessanter Ansatz sei an dieser Stelle kurz erläutert: Wissenschaftler des Dutch National Research Institute in Amsterdam haben Experimente angestellt, ob sich Datenkompressions-Software zur Unterscheidung von Musikrichtungen einsetzen lässt. Dazu haben sie Stücke von Beethoven, Miles Davis und Jimi Hendrix durch das Linux-Programm Bzip2 geschickt. Erstaunlicherweise war diese Methode überaus erfolgreich.

---

<sup>6</sup>Mehr Informationen: <http://www.jfugue.org/>

## Kapitel 4 Digitale Verarbeitung von Musik

---

So lassen sich nicht nur verschiedene Stilrichtungen unterscheiden, sogar der Komponist des Werkes kann somit ermittelt werden (Vgl. REDAKTION (2003)).

Analysiert man dieses zunächst überraschende Ergebnis, kommt man jedoch auf eine schlüssige Begründung. Das Bzip-Verfahren basiert auf dem einfachen System, dass sich wiederholende Sequenzen einer Datei zusammengefasst werden. Durch eine Liste von Zeigern auf die entsprechenden Abschnitte, kann dann die Original-Datei verkürzt repräsentiert werden. Ähnliche Musik, zum Beispiel gleicher Stilrichtungen, oder gleicher Interpreten, weisen - wie das Attribut ähnlich bereits verrät - übereinstimmende Abschnitte vor. Daraus resultiert in den komprimierten Musik-Dateien ein gewisser Stamm von Sequenzen, der in allen komprimierten Dateien einer Stilrichtung oder eines Komponisten vorhanden ist.

Diese Eigenschaft von Musik bzw. komprimierter Musik könnte eventuell ebenfalls für ein Such-/ Vergleichsbasiertes intelligentes System genutzt werden.

# 5 Definition des Prototyps

Nach der Betrachtung der Grundlagen und erster Ansätze soll nun erörtert werden, welche der vorgestellten Möglichkeiten, die in den vorangegangenen drei Kapiteln vorgestellten Bereiche, sinnvoll bzw. erfolgsversprechend erscheinen. Daraus soll wiederum die Entscheidung der Merkmale des, in dieser Arbeit zu erstellenden, Prototypen abgeleitet werden. Aufgrund der vielen verschiedenen Ansätze, wird von Beginn an auch die Erstellung mehrere Prototypen in Betracht gezogen.

## 5.1 Auswahl des Datenformates

Zunächst soll die Wahl des Datenformates betrachtet werden. In Kapitel 4 wurden zwei grundsätzlich mögliche Arten von Formaten besprochen:

- Audiodateien und
- MIDI-Dateien (bzw. MIDI-ähnliche Formate)

Aus beiden Arten können Daten gewonnen werden, welche als Eingabe in neuronale Netze oder in Systeme der klassischen KI genutzt werden können. Weiterhin wurde die Nutzung von komprimierten Musik-Dateien besprochen. Da dieses Prinzip jedoch auf der Ähnlichkeit von einzelnen Teilen der Dateien basiert, ist es nicht vom eigentlichen Datenformat abhängig und wird deswegen zunächst ausgeblendet.

Die Nutzung von Audiodateien hat den Vorteil, dass aufgrund der Verbreitung derartiger Formate im Alltag, eine große Auswahl an möglichen Lern- bzw. Testdaten zur Verfügung steht. Außerdem enthält eine Audiodatei bereits alle möglichen Informationen eines Musikstückes (Melodie, Spielweise und Text). Allerdings liegen die einzelnen Teile nicht einzeln vor und können somit nicht getrennt betrachtet werden. Auch ist eine inhaltliche Analyse des Textes nur schwer möglich. Selbst mit bereits existierenden komplexen Systemen der Spracherkennung ist es praktisch unmöglich den Text aus einem Lied vollständig zu erkennen - Nebengeräusche der Instrumente und die unnatürliche Aussprache der Wörter durch das Singen der Melodie verhindern eine eindeutige Filterung und Erkennung des Textes. So ist ein sprachverstehendes System zur Analyse

des Textes, nach dem Stand der heutigen Technik, auf Basis von Audiodateien, nicht möglich. Werden die Aussagen zur Motivation dieses Projektes, welche in der Einleitung auf Seite 1 formuliert wurden, mit einbezogen, stößt man auf ein weiteres Problem: Wenn später ein System entwickelt werden soll, welches Musik selbst komponiert, muss dieses System in der Lage sein Daten zu erzeugen, die vergleichbar mit den Lern- und Testdaten sind. Das Erzeugen einer Audio-Datei, welche vergleichbar mit einer richtigen Tonaufnahme ist, ist zwar nicht unmöglich, jedoch sehr aufwendig.

Im Vergleich zum Audio-Format enthalten MIDI-Dateien grundsätzlich weniger Informationen über ein Musikstück. So fehlt hier der Gesang, also der Text, und auch der Sound und die Dynamik der Instrumente ist im Audio-Format deutlicher dargestellt. Allerdings ist es hier möglich jede einzelne Melodie-Linie jedes Instruments auszuwerten. Ebenso wie beim Audio-Format, liegen viele Musikstücke bereits im MIDI-Format vor. Die Verarbeitung der Daten ist im Vergleich zu Audio-Daten auch einfacher. So werden keine Filter-Verfahren und keine Fourier-Transformation gebraucht, um nutzbare Eingabe-Daten zu gewinnen. Die Auswertung von MIDI-Dateien ist in den meisten Programmiersprache sehr einfach, da aufgrund der weiten Verbreitung von MIDI sehr viele Bibliotheken existieren.

Insgesamt bietet das Audio-Format mehr Möglichkeiten. Ein intelligentes Musik verstehendes und analysierendes System wird um die Nutzung eines solchen Formates in der Zukunft nicht vorbei kommen - Schließlich stellt das Audio-Format auch das Format dar, welches der Mensch nutzt, um Musik zu klassifizieren. Jedoch ist der Nutzung eines solchen Formates mit sehr großen Aufwänden verbunden, um aus dem Format die Nutzdaten zu extrahieren. Teilweise ist es auch überhaupt noch nicht möglich einzelne Informationen zu gewinnen (Beispiel Sprache). Aus diesem Grund wird für diese Arbeit zunächst das MIDI-Format genutzt. Dies ermöglicht den Fokus der Arbeit auf dem eigentlichen Thema der Arbeit: Der Musik bzw. ihrer Komposition. Für die grundlegenden Überlegungen und Forschungen ist dieses Format völlig ausreichend. Sollte ein Prototyp auch den Text zu einem Musikstück analysieren, kann dies durch Text-Format-ähnliche Dateien erfolgen, die neben dem Text als normalen ASCII-Code auch Steuer Informationen erhalten, zu welchen Melodie-Abschnitten der entsprechende Text gehört. Im Verlauf der Arbeit wurde die Nutzung des eigentlichen, „richtigen“ MIDI-Formates

überdacht und durch den auf Seite 37 erwähnten MusicString, der im Zusammenhang mit der jfugue-API verwendet werden kann, ersetzt. Dieser vereinfacht die Arbeit noch weiter. Da das Dateiformat auf ASCII-basiert, ist es auch ohne zusätzliche Software leicht (für Menschen) lesbar, so können bei der Entwicklung der Prototyp(en) die Ergebnisse leichter analysiert werden. Im Folgenden Kapitel, in dem die Realisierung beschrieben wird, wird die ursprüngliche Verwendung des MIDI-Formates nicht weiter betrachtet. Es sei jedoch an dieser Stelle darauf hingewiesen, dass sich erst im Verlaufe der Realisierung der Nutzen einer Verwendung der jfugue-Bibliothek auszahlt. Für spätere Weiterentwicklungen ist es auch leicht möglich MIDI-Dateien in das jfugue-Format umzuwandeln, da dieses auf den gleichen Prinzipien arbeitet. Ein derartiger Adapter konnte während des Projektzeitraums dieser Arbeit nicht umgesetzt werden.

## **5.2 Auswahl der zu untersuchenden Merkmale von Musik**

In Kapitel 2 wurde analysiert aus welchen Bestandteilen ein Musikstück besteht und inwiefern diese Einfluss auf die Bewertung durch den Zuhörer haben. Dabei wurden drei wichtige Säulen eines Liedes herausgestellt: Die Komposition, die Interpretation, sowie der Text. Für alle drei Komponenten konnte, anhand von Beispielen und wissenschaftlichen Begründungen aus verschiedener Literatur, nachgewiesen werden, dass sie die Bewertung der Musik beeinflussen. So wäre es sinnvoll alle Bestandteile zu untersuchen. Mit Hilfe der verschiedenen KI-Techniken, wie sie in Kapitel 3 vorgestellt wurden, ist dies auch durchaus möglich:

- Mit Hilfe der Logik bzw. unter Nutzung von wissensbasierten Systemen oder neuronalen Netzen wäre es möglich die Komposition zu analysieren. Zum einen auf ihre Richtigkeit, zum anderen auf ihren Rhythmus. Es könnte die Frage geklärt werden, ob ein neues Musikstück dem Geschmack des Systems entspricht. Finden sich etwaige parallelen zu bereits bekannten Stücken? Wenn ja ist davon auszugehen, dass auch das neue Stück dem System bzw. einem Benutzer gefällt.
- Die Nutzung von semantischen Netzen und entsprechenden Algorithmen könnte es erlauben Sinnzusammenhänge zwischen dem Text, den Melodien oder ihrer Interpretation zu ihren Komponisten bzw. Künstlern her zu stellen. Die semantischen Netze könnten in diesem Fall ebenfalls in einem wissensbasierten System integriert

sein, welches, neben den eigentlichen Informationen über die Lieder, auch Wissen über die Welt, zum Beispiel über die Komponisten und das umgebende Weltgeschehen und der Situation des Benutzers, speichert. So könnte es möglich sein ein all-umfassendes System zu schreiben, welches Musik wirklich interpretieren kann und somit erschließen kann, ob ein Musikstück einen Benutzer emotional berührt – was zwangsläufig zum Gefallen eines Songs, wie in Kapitel 2 beschrieben, führt.

- Würde das Audio-Format als Grundlage der Daten genutzt werden, ergäben sich unter Verwendung von neuronalen Netzen weitere Möglichkeiten, ähnlich zu den Verfahren des Sprachverstehens mit neuronalen Netzen.

Weiterhin sind noch viele weitere Ansätze denkbar. Die Möglichkeiten die verschiedenen Bestandteile der Musik einzeln oder in ihrer Gesamtheit zu analysieren sind also breit gefächert. Im Rahmen dieser Arbeit wurde zunächst der Fokus auf die Komposition und die Melodie gelegt. Die anderen Bereiche sind es aber dennoch Wert in anderen Arbeiten betrachtet und genauer analysiert zu werden. Da im Rahmen dieser Arbeit nicht alle Bestandteile mit einbezogen werden konnten, musste eine Entscheidung zugunsten eines Bereiches fallen. Die Entscheidung für Komposition und Melodie ist darin begründet, dass die Komposition die Grundlage eines jeden Musikstückes ist. Außerdem wurde als Motivation, in der Einleitung auf Seite 9 zu finden, das Fernziel genannt, ein intelligentes System zu schreiben, welches selbst komponieren kann. Für ein solches System ist das Grundverständnis einer Komposition bzw. das Verstehen von Kompositionen ein wichtiger Basis-Bestandteil.

### **5.3 Auswahl von Ansätzen der künstlichen Intelligenz**

In Kapitel 3 wurden die Grundlagen der verschiedenen Basis-Technologien der künstlichen Intelligenz vorgestellt. So wurden die klassischen Verfahren der KI zur Wissensrepräsentation und Wissensverarbeitung dargestellt, als auch die „modernen“ Ansätze aus dem Bereich der künstlichen neuronalen Netze. Dabei wurde jeweils bereits eine kurze Einschätzung formuliert, ob sich die Verfahren für die Verwendung in einem intelligenten System zur Analyse von Musikstücken eignen. Diese Schlussfolgerungen seien an dieser Stelle nochmals zusammengefasst und um einige Gedanken erweitert:

- **Logik.** Die Logik hat zwar durch ihre Zweiwertigkeit ein stark beschränktes Einsatzfeld. Kann jedoch zur Überprüfung von grundlegenden Regeln der Musiktheorie effektiv genutzt werden.
- **Semantische Netze.** Da sie vor allem im Bereich von sprachverstehenden Systemen zum Einsatz kommen, könnten sie sich durchaus eignen auch inhaltliche Zusammenhänge im Bereich der Musik zu erkennen. Eventuell müssten die etablierten Verfahren des Sprachverstehens in Richtung der Merkmale von Musik modifiziert werden. Zwar wurden in der Literatur keine Hinweise auf derartige Forschungen verbunden. Doch die Parallelen zwischen Sprach- und Musikverständnis lassen den Schluss zu, dass semantische Netze für die Verwendung zu diesem Zweck ebenfalls sinnvoll sein kann.
- **Fuzzy-Logik.** Als Methode zur Darstellung von vagen Wissen wurde die Fuzzy-Logik dargestellt. Zwar handelt es sich bei Aussagen wie „Das ist ein gutes Lied“ um vages Wissen, jedoch können die Methoden der Fuzzy-Logik nicht auf die Musik übertragen werden, da keine Möglichkeiten zur Quantifizierung gefunden werden können. Die Fuzzy-Logik kann somit für den Einsatz in dieser Arbeit ausgeschlossen werden. Eine weitere vergleichbare Methode zur Verarbeitung von vagen Wissen wurde in der Literatur nicht gefunden.
- **Suche.** Ebenfalls wurde die Bedeutung von Suchalgorithmen in der KI beleuchtet. Als Grundlage von vielen wissensverarbeitenden Algorithmen ist die Suche ein wichtiger Aspekt. Sollte für den Prototyp auf Methoden der klassischen KI zurückgegriffen werden, wird die Suche mit Hoher Wahrscheinlichkeit ebenfalls ein Thema beim Entwurf des Prototyps sein.
- **Wissensbasierte Systeme.** Ähnlich wie die Suche folgt aus der Verwendung der grundlegenden Techniken der künstlichen KI oft der Aufbau eines wissensbasierten Systems. So wird auch die Architektur eines solchen Systems von Bedeutung sein, wenn ein Prototyp auf Basis der klassischen Verfahren konstruiert werden soll.
- **Künstliche neuronale Netze.** Neben den klassischen Verfahren wurde auch der Ansatz der künstlichen neuronalen Netze vorgestellt. Grundsätzlich ist dieser Ansatz ebenfalls erfolgsversprechend. Ähnliche Arbeiten im Bereich der sprachver-

stehenden Systeme wurden bereits im Bereich der neuronalen Netze durchgeführt. Bei der Vorstellung der neuronalen Netze wurden auch die verschiedenen Arten des Trainings, sowie die verschiedenen Netzarchitekturen betrachtet. Dabei stellte sich heraus, dass vor allem die Backpropagation Netze für den Einsatz im Bereich der Musik-Analyse geeignet sein könnten. So wurden bereits ähnliche Aufgaben, zum Beispiel das Erkennen von Mustern oder die automatisierte Qualitätskontrolle, mit Hilfe dieser Netze durchgeführt.

Auch die Erweiterung der Backpropagation Netze, die partiell rückgekoppelten Netze, könnte eventuell sinnvoll eingesetzt werden. Zwar wird diese Architektur hauptsächlich zur Prognose von zukünftigen Werten (beispielsweise Aktienkurse) eingesetzt, jedoch lässt sich diese Fähigkeit auch im Bereich der Qualitätskontrolle von Musik einsetzen: So könnte ein Teil einer zu prüfenden Melodie als Eingabe des Netzes dienen. Prognostiziert es den gleichen Ton, welcher auch in der Melodie als nächstes kommt, kann davon ausgegangen werden, dass die Wahl des Tons richtig bzw. gut ist. Ein derartiger Lösungsansatz in einem vergleichbaren Gebiet wurde jedoch bei der Recherche nicht gefunden.

Als dritter interessanter Ansatz wurde das neuronale Gas vorgestellt. Diese Form der neuronalen Netze wurde erst in den letzten Jahren entwickelt und ist noch wenig erforscht. Jedoch ergaben die Ergebnisse der bisherigen Forschungen bereits interessante Ergebnisse. So sind neuronale Gase in vielen Gebieten effektiver als ihr „Vorfahre“, die selbstorganisierenden Karten. Zwar haben die selbstorganisierenden Karten andere Einsatzfelder, als die Backpropagation Netze. Ihre Weiterentwicklung, das neuronale Gas, konnte aber bereits auch in Gebieten, welche normalerweise durch Backpropagation Netze bearbeitet werden, Erfolge erzielen – beispielsweise in der Spracherkennung. So scheint der Einsatz von neuronalen Gas zur Klassifizierung von Musik auch nicht ausgeschlossen.

Die Verwendung von selbstorganisierenden Karten kann jedoch weitestgehend außer acht gelassen werden, da ihre Einsatzbereiche auf Aufgaben beschränkt ist, in denen Datenmengen automatisiert gruppiert werden sollen. Diese Problemfelder finden sich in der Regel im Bereich von Data-Warehouse-Systemen wieder.

Die Palette der möglichen Ansätze im Bereich der KI ist also ebenfalls breit gefächert. Unter Einbeziehung der Festlegungen, welche im Hinblick auf das zu nutzende Daten-

format und die zu analysierenden Teile der Musik, bereits getroffen wurden, scheidet die Nutzung von semantischen Netzen aus. Da zunächst keine inhaltlichen Zusammenhänge hergestellt werden sollen, ist ihr Einsatz nicht sinnvoll. Jedoch kann es eine Aufgabe für Folgeprojekte sein mit Hilfe von semantischen Netzen die inhaltlichen Zusammenhänge zwischen Melodie, Komponist und der Umwelt zu analysieren. Da diese Idee bereits eine hohe Stufe der musikalischen Analyse darstellt, passt er auch noch nicht in den Rahmen dieser eher grundlegenden Arbeit. Bleiben also die Nutzung der Logik, der Suche und den Erkenntnissen über wissensbasierte Systeme, sowie neuronale Netze. Da es sich dabei auf der einen Seite um klassische Verfahren und auf der anderen Seite um eine ganz andere Annäherung an das Thema handelt und da auf Grundlage der bisherigen Kenntnisse keine Entscheidung pro oder kontra einer der beiden Ansätze formuliert werden kann, sollen in dieser Arbeit zwei verschiedene Prototypen erstellt werden.

Der Ansatz der klassischen KI wird in einem Prototyp unter Verwendung von Verfahren der Logik und der Suche implementiert. Letztendlich soll ein einfaches wissensbasiertes System konstruiert werden. In Abschnitt 3.5 wurden verschiedene Arten von wissensbasierten Systemen vorgestellt. Eine Kombination aus einem Fallbasierten und einem Regelbasierten-System erscheint sinnvoll. So kann die musiktheoretische Richtigkeit eines Liedes mit Hilfe einer Regelbasis überprüft werden, zusätzlich kann ein Geschmack erlernt werden - Dieser Geschmack wird durch das Fall-Archiv im System repräsentiert. Der ebenfalls in Abschnitt 3.5 erwähnte Ansatz der Suchbäume, passt nicht zur Problemstellung dieser Arbeit.

Der zweite Prototyp soll ein neuronales Backpropagation Netz implementieren. Diese Netzarchitektur wurde gewählt, da sie aufgrund ihrer gewöhnlichen Einsatzfelder am ehesten zum Thema dieser Arbeit passt. Der Einsatz von partiell rückgekoppelten Netzen, als Erweiterung, ist sicherlich interessant, soll aber zunächst nicht betrachtet werden. Jedoch besteht die Möglichkeit, das zu entwickelnde backpropagation Netz später weiter zu entwickeln. Der Ansatz der neuronalen Gase soll ebenfalls nicht betrachtet werden. Zwar könnte auch dieser zum Erfolg zwingen - Die Forschungen auf diesem Gebiet sind jedoch nicht weit genug voran geschritten, um darüber eindeutige Aussagen zu treffen. Sicherlich könnte auch diese Ansatz in späteren Arbeiten näher betrachtet werden. Die Entscheidung für das Backpropagation-Netz basiert also vor allem auf den besseren

Erfolgsaussichten und der Möglichkeit es später zu einem partiell rückgekoppelten Netz zu erweitern.

## 5.4 Weitere Vorüberlegungen

In den voran gegangenen Abschnitten wurden bereits Festlegungen getroffen und die zu nutzenden Verfahren gewählt. Diese Festlegungen sollen nun zusammengefasst werden und durch weitere Merkmale der Prototypen ergänzt werden.

Als Eingabe-Format für Lern- und Testdaten soll das MusicString-Format der jfugue-API genutzt werden. Hierzu wurde bereits vor Beginn der Implementierung eine Menge von Testdaten erstellt. Dabei wurden die Noten der gewählten Musikstücke mit Hilfe eines einfachen Texteditors in das MusicString-Format in „Handarbeit“ umgesetzt. Die Auswahl der Songs erfolgte ohne bestimmte Einschränkungen. Um jedoch eine möglichst breite Wissensbasis, trotz einer nicht all zu hohen Anzahl an Testdaten, zu erreichen, wurde darauf geachtet, dass Lieder unterschiedlicher Genre, Interpreten und Entstehungsjahre in den Pool der Testdaten aufgenommen werden. Gleichzeitig wurde aber auch versucht einen möglichst einheitlichen Musikgeschmack zu repräsentieren, der sich auf mitteleuropäische bzw. amerikanische Rock- und Pop-Musik, sowie Volkslieder und Schlager beschränkt. Die Auswahl der Musikstücke ist in Anhang A auf Seite 73 aufgelistet. Die Menge der Lern- und Testdaten muss für beide Prototypen später noch um einen Anteil von Musikstücken erweitert werden, die nicht *gut*, also keine richtigen Musikstücke, sind. Dazu soll in der Realisierungsphase ein einfaches Programm zur zufälligen Generierung von MusicString-Dateien erstellt werden. Dieses Programm soll jedoch nicht nur auf Zufall basierende Kompositionen erzeugen, sondern auch einige Kompositions-Regeln beachten. Würden rein zufällige Dateien erzeugt werden, könnte die Unterscheidung zwischen echter und erzeugter Musik zu leicht fallen, sodass die Prototypen zu wenig Schwierigkeiten haben, die Musik zu unterscheiden.

Ziel der Prototypen ist es, nach der Lern- und der Testphase, neue Stücke mit *gut* oder nicht *gut* zu bewerten. Der Wert der Einschätzung kann dabei zwischen 0 und 1 liegen. Wünschenswert sind natürlich Ergebnisse, die möglichst nah an 0 oder 1 liegen und gleichzeitig von einem Mensch gleichwertig eingeschätzt werden. 0 steht bei dieser

Klassifizierung für *nicht gut*, 1 für *gut*. So werden alle echten Musikstücke in der Lern- bzw. Testmenge mit 1 eingestuft, alle vom Generator erzeugten sollen mit 0 eingestuft werden.

Für die Implementierung der Prototyp soll ausschließlich Java verwendet werden. Während der Recherche zu dieser Arbeit wurden auch andere Möglichkeiten in Betracht gezogen:

- **SNNS/JNNS.** Der Stuttgarter Neuronale Netze Simulator (SNNS) ist ein in den 90er Jahren an der Universität Stuttgart entwickeltes Software-Paket, welches zahlreiche Funktionen zur Arbeit mit Neuronalen Netzen zur Verfügung stellt. Zahlreiche Werkzeuge nutzen die Funktionen des SNNS, so zum Beispiel der Java Neuronale Netze Simulator (JNNS), der ebenfalls in Stuttgart entwickelt wurde. Der JNNS stellt eine grafische Benutzeroberfläche für den SNNS zur Verfügung. Die Möglichkeiten des JNNS bzw. des SNNS wurden durch kleine Tests vor Beginn der Implementierung untersucht. Dabei stellte sich heraus, dass der Umgang mit dem Werkzeug sehr komplex ist und eine längere Einarbeitungszeit nötig wäre, um Nutzen für das Thema der Studienarbeit aus den Programmen zu ziehen. Aufgrund der relativ kurzen Projektzeit, wurde die Nutzung dieser Systeme ausgeschlossen. Zwar beinhalten diese zahlreiche Funktionen zum modellieren von Netzen, zur Modifikation der Lernalgorithmen und viele Analyse-Möglichkeiten. Die Möglichkeiten einer einfachen Implementierung in einer üblichen Hochsprache wurde für dieses Projekt, in dem es ausschließlich zunächst um grundlegende Tests und Analysen geht, aber auch als ausreichend betrachtet. Sollte sich der Ansatz der Nutzung von neuronalen Netze als erfolgreich herausstellen, kann man in Folge-Arbeiten durchaus auf derartige Werkzeuge zurückgreifen, um die Arbeit und den Aufbau der Netze genauer analysieren zu können.

Weitere Informationen über SNNS finden sich auf den Servern der Uni Tübingen, welche das Projekt heute leitet (<http://www.ra.cs.uni-tuebingen.de/SNNS/>), ebenfalls finden sich dort die aktuellen Daten zum JNNS Projekt (<http://www.ra.cs.uni-tuebingen.de/software/JavaNNS/>).

- **PROLOG.** Der Einsatz von Prolog zur Realisierung des Logik-Teils wurde ebenfalls in Betracht gezogen. Die Integration von PROLOG in andere Programmiersprachen wie Java oder auch C/ C++ stellt keine große Hürde dar, da für die meisten

verbreiteten Sprachen Bibliotheken für die Integration von Prolog existieren.<sup>7</sup> Es wurde jedoch der Entschluss getroffen, den kompletten Logik-Teil ebenfalls in Java zu implementieren. Ausschlaggebend für diese Entscheidung war, dass die Regeln welche umgesetzt werden nicht besonders umfangreich sind und deswegen auch einfach durch normale Bedingungs-Konstrukte üblicher Programmiersprachen zu realisieren sind. Die Stärken von PROLOG im Hinblick auf das Verarbeiten von großen Wissensbasen oder die Fähigkeit durch Resolution Schlüsse zu ziehen werden in diesem Projekt nicht genutzt. Der Mehraufwand, der durch die Einbindung von PROLOG betrieben werden müsste, würde sich also nicht lohnen.

Die Entscheidung für Java als Implementierungssprache fiel aufgrund der weiten Verbreitung unter der Studentenschaft. Praktisch jeder Informatik-Student beherrscht diese Sprache, so ist es später leicht für andere Studenten möglich sich in die hier entstandene Arbeit einzuarbeiten. Zudem unterstützt Java das objektorientierte Programmierparadigma nahezu perfekt. So kann der Aufbau der Prototypen sehr strukturiert, logisch und überschaubar erfolgen, sodass das Prinzip leicht zu verstehen ist.

---

<sup>7</sup>Zum Beispiel: <http://www.gnu.org/software/gnuprologjava/>

## 6 Realisierung

In diesem Kapitel soll nun die Umsetzung der beiden Prototypen, sowie des MusicString-Generators beschrieben werden. Beide Prototypen, und der Generator wurden innerhalb eines Gesamtprojektes entwickelt. Deswegen wird zunächst eine kurze Einführung in die Gesamtarchitektur erfolgen. Danach wird jede Komponente einzeln, im Hinblick auf Ihre Architektur und wichtige Algorithmen vorgestellt. Das Kapitel wird durch die Beschreibung der Tests und einer kurzen Auswertung der Ergebnisse komplettiert.

Die Erläuterung der Algorithmen soll in einer eher abstrakten Art und Weise erfolgen; Ohne konkreten Java-Quellcode. Dies soll die Lesbarkeit der Ausführungen fördern und somit zum Verständnis beitragen. Bei Interesse kann der Quellcode der Programme betrachtet werden. Die Projekt-Dateien befinden sich im Dateianhang der Arbeit bzw. können nach der Veröffentlichung der Arbeit auch aus dem Internet heruntergeladen werden.<sup>8</sup>

### 6.1 Architektur-Überblick

Da beide Prototypen auf der Basis der gleichen Daten arbeiten, und somit auch auf gleiche Funktionen zurückgreifen müssen, wurden beide in einem Gesamtprojekt integriert. Das Projekt besteht aus fünf Hauptkomponenten:

- **Hauptprogramm.** Das Hauptprogramm dient als Benutzerschnittstelle für alle anderen Komponenten. Es enthält die *main()*-Methode des Programms. Es implementiert die Auswahl der möglichen Optionen, sodass das Programm von einem Benutzer bequem über eine Konsole ausgeführt werden kann. Die Umsetzung dieser Komponente wird nicht näher beschrieben, da sie nicht mehr als Standard-Verfahren üblicher Konsolen-Anwendungen besitzt.
- **Song.** Diese Komponente stellt das Musikstück als Objekt innerhalb der Anwendung dar. Eine Instanz dieser Komponente wird erzeugt, um eine MusicStream-Datei einzulesen und die enthaltenen Daten für die anderen Komponenten der Anwendung zur Verfügung zu stellen. Die Komponente enthält außerdem Funktionen,

---

<sup>8</sup>Nach Veröffentlichung der Arbeit, steht diese inkl. aller Projekt-Dateien unter <http://files.michaelwellner.de> zur Verfügung

welche das Musikstück auf bestimmte Merkmale untersuchen. Diese Funktionen und die entsprechenden Algorithmen werden im Abschnitt 6.6 erläutert.

- **MusicStream-Generator.** Der MusicStream-Generator erzeugt zufällige Musikstücke. Wie auf Seite 46 bereits erwähnt, soll dieser nicht rein zufällige Sequenzen erzeugen, sondern bestimmte Regeln beachten. Dabei greift der Generator auch auf Funktionen der Song-Komponente zurück. Die Umsetzung des Generators ist in Abschnitt 6.3 beschrieben.
- **Neuronales Netz/ wissensbasiertes System.** Schließlich enthält das Projekt auch die beiden Kernsysteme, welche die beiden Prototypen darstellen. Beide Prototypen nutzen die Song-Komponente zum Verarbeiten der MusicStream-Dateien. Ihre Umsetzung wird in den Abschnitten 6.4 bzw. 6.6 beschrieben.

Die Gesamtarchitektur ist in der folgenden Abbildung noch einmal schematisch, inklusive der Beziehungen zwischen den Komponenten, dargestellt.

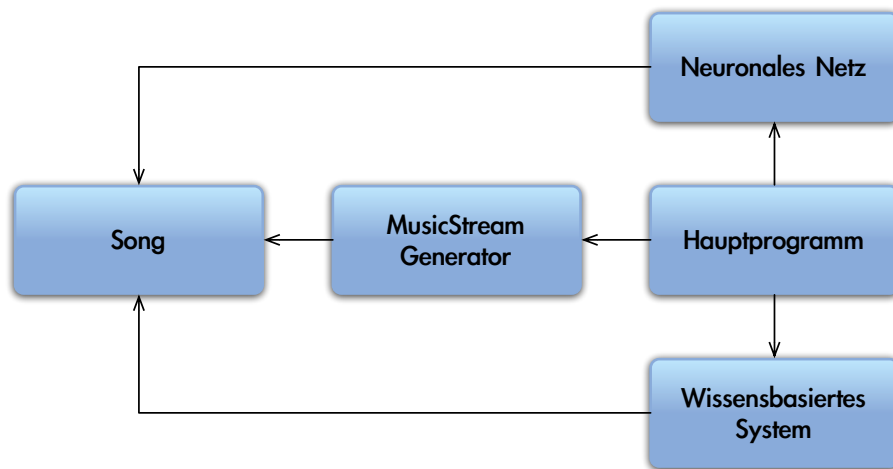


Abbildung 8: Hauptbestandteile des Prototyps

## 6.2 Einlesen und Analyse eines Musikstückes

Grundsätzlich dient die Komponente Song den anderen Komponenten als Data-Access-Objekt. Diese Architektur macht die anderen Komponenten unabhängig vom verwendeten Datenformat. So ist es später leicht möglich das verwendete Datenformat auf MIDI-,

oder sogar auf Audio-Formate umzustellen. Neben dieser grundlegenden Aufgabe enthält die Komponente jedoch noch weitere Funktionen, welche bestimmte Merkmale eines Musikstückes untersuchen: So ist die Komponente in der Lage grundlegende statistische Angaben aus dem Musikstück zu extrahieren. Der Tonumfang, die Häufigkeitsverteilung der verschiedenen Töne, sowie die Anzahl von Tönen, die außerhalb der Tonart liegen, werden ermittelt. Für den Tonumfang und die Häufigkeitsverteilung wird einmal durch alle vorhandenen Noten iteriert. Während dieser Iteration wird das Vorkommen der Noten in einer Tabelle notiert, diese Tabelle stellt danach die Häufigkeitsverteilung dar, außerdem werden der höchste und der niedrigste Ton gemerkt um den Tonumfang zu berechnen. Weiterhin transponiert die Komponente beim Einlesen der Dateien alle Lieder auf eine einheitliche Tonart (C-Dur), um sie später besser vergleichen zu können.

Die Auswahl der Merkmale basiert auf den Erkenntnissen, die in Kapitel 2 formuliert wurden. Demnach bewegt sich ein Lied stets um ein tonales Zentrum (Tonumfang). Außerdem wurde festgestellt, dass die Häufigkeitsverteilung von Tönen vergleichbar mit der Häufigkeitsverteilung von Buchstaben in natürlicher Sprache ist - Deswegen wird auch diese analysiert. Außerdem wurde bereits in den Grundlagen beschrieben, dass sich Melodie-Sequenzen in Musikstücken oft wiederholen, neben wiederholenden Melodie-Sequenzen finden sich in Liedern auch immer wiederkehrende Rhythmus-Muster wieder. Aus diesem Grund implementiert die Komponente Algorithmen, welche derartige Wiederholungen finden. Weiterhin kann die Komponente das von ihr verwaltete Lied mit einem anderen vergleichen und eine Aussage darüber treffen wie ähnlich die Stücke untereinander sind. Die Algorithmen zum Finden von Wiederholungen und zum Vergleichen zweier Lieder werden im Folgenden näher beschrieben. Zunächst zeigt Abbildung 9 jedoch noch einmal den Aufbau der Komponente.

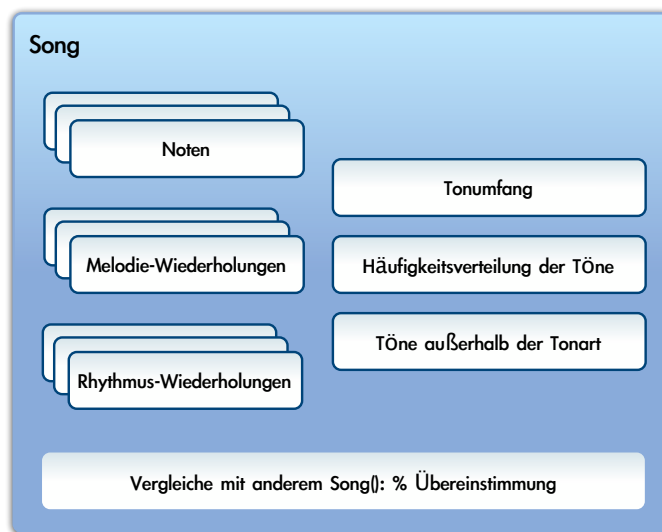


Abbildung 9: Die Komponente Song mit ihren wichtigsten Bestandteilen.

Da nach längerer Recherche kein vergleichbarer, bekannter Algorithmus zum Finden von Wiederholungen in einer Datei gefunden wurde, soll hier eine Lösung vorgestellt werden, die im Rahmen dieser Arbeit entstanden ist. Dieser einfache Algorithmus stellt keinen Anspruch an Performance oder Laufzeit. Die Ziele bei der Entwicklung dieses Verfahrens seien im Folgenden kurz zusammengefasst:

- Eine Sequenz besteht aus mindestens 4 Tönen.
- Es sollen die längst möglichen, sich wiederholenden Stücke gefunden werden. Dabei soll gespeichert werden, wie oft und an welchen Stellen sie auftreten.
- Eine sich wiederholende Sequenz gilt nur als solche, wenn sie nicht bereits komplett Bestandteil einer Sequenz ist. Liegt nur der Anfang oder das Ende in einer bereits erkannten Sequenz ist das erlaubt. Dieses Verhalten ist besonders wichtig, da nur so sich aufbauende Sequenzen erkannt werden können. In einer ersten Version des Algorithmus führte das zu Problemen und fehlerhaften Analysen.

Der Algorithmus arbeitet zunächst mit 4 Zeigern, die auf einzelne Noten im Musikstück zeigen. Sie markieren dadurch zwei Abschnitte: Es handelt sich also jeweils um einen Start- und einen End-Zeiger. Am Anfang zeigen alle Zeiger auf die erste Note. Dieser Zustand ist in Abbildung 10 dargestellt.

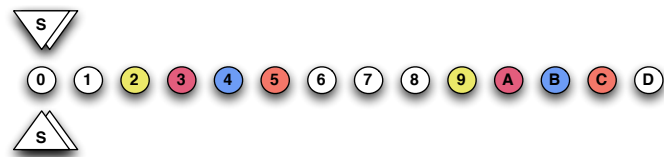


Abbildung 10: Wiederholungen finden; Schritt 1

In dem Beispiel sind die Stellen 2 bis 5 und 9 bis C identisch. Der Algorithmus besteht aus zwei verschachtelten Schleifen. Die äußere Schleife durchläuft mit dem ersten Start-Zeiger das gesamte Lied. Für jede Note wird nach der gleichen Note im Restlied gesucht. Wird sie noch einmal gefunden, wird der zweite Zeiger auf diesen Ton gesetzt. So wandert der erste Zeiger solange durch das Lied, bis er auf einem Ton steht, der auch ein zweites mal im Lied auftritt. Zu diesem Zeitpunkt stehen die beiden Start-Zeiger auf zwei unterschiedlichen Stellen. Die beiden End-Zeiger werden ebenfalls auf diese Stellen gesetzt. Die Situation ist in Abbildung 11 dargestellt.

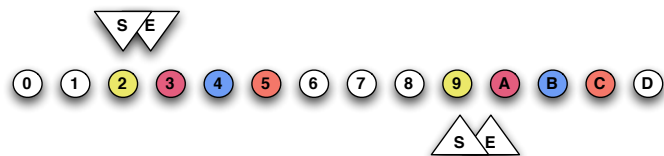


Abbildung 11: Wiederholungen finden; Schritt 2

An diesem Punkt beginnt die innere Schleife. In dieser Schleife werden die beiden End-Zeiger solange weiter bewegt, bis der Ton, welcher nach ihrer jeweiligen aktuellen Position folgt, nicht mehr gleich ist. Handelt es sich also um einen einzelnen Ton der doppelt auftritt, verbleiben die End-Zeiger auf der gleichen Position wie die Start-Zeiger. Handelt es sich jedoch tatsächlich um eine Sequenz wandern die Zeiger bis an das Ende dieser Sequenz, so wie es in Abbildung 12 dargestellt ist.

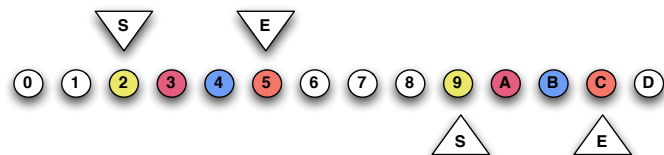


Abbildung 12: Wiederholungen finden; Schritt 3

Nach diesem Prinzip werden zunächst alle sich wiederholenden Sequenzen gefunden. Im zweiten Teil des Algorithmus wird geprüft, ob die Sequenz in die Sammlung der Wiederholungen aufgenommen werden kann. Dies geschieht, falls die oben festgelegten Bedingungen erfüllt sind. Durch die Aufteilung des Algorithmus in die zwei Teile, wird dieser überschaubar und leicht nachvollziehbar. In einer ersten Version wurde versucht die Bedingungen im Schleifen-Konstrukt unterzubringen. Dies führte zu einem komplexen und leider auch fehlerhaften Ergebnis. Im Programm ist der Algorithmus mit weiteren kleinen Details implementiert, so muss beispielsweise auch erkannt werden, wann sich die zwei Sequenzen beim durchlaufen des Liedes überschneiden.

Das finden von Ähnlichkeiten in zwei unterschiedlichen Liedern funktioniert prinzipiell nach dem gleichen Prinzip. Zu dem vorgestellten Algorithmus müssen hierfür nur kleine Modifikationen vorgenommen werden. Aus diesem Grund wird dieser Algorithmus nicht explizit beschrieben.

### **6.3 Entwicklung eines einfachen MusicStream-Generators**

Die MusicStream-Generator Komponente ist für die Erstellung von Lern- und Testdaten zuständig. Mit den erzeugten Dateien soll die Leistungsfähigkeit der Prototypen getestet werden. Um es den Systemen möglichst schwer zu machen, sollen die generierten MusicStreams bereits einige Regeln beachten. Deswegen wurde die standardmäßige Zufallsfunktion, wie sie bereits in den Standard Paketen von Java enthalten ist, erweitert. Die Erweiterung kombiniert die normal gebräuchliche *Math.random()*-Methode mit einer Funktion, sodass die Häufigkeitsverteilung der eigentlich gleichmäßig verteilten Resultate der *Math.random()*-Methode in verschiedenen Häufigkeiten auftreten. Dieses vorgehen soll im Folgenden an einem Beispiel, wie es im MusicStream-Generator auch angewandt wurde, veranschaulicht werden. Zuvor soll als Grundlage für diese Ausführungen noch einmal die normale Erzeugung einer Zufallszahl vorgestellt werden.

In den allermeisten Programmiersprachen existieren Methoden zum Erzeugen einer Zufallszahl. Dabei hat man jedoch selten die Möglichkeit einen Minimal und einen Maximalwert festzulegen. Auch die Anzahl der Nachkommastellen kann selten bei der Bestimmung einer Zufallszahl angegeben werden. Normalerweise gibt die Zufallsfunktionen

einen Double-Wert zwischen 0 und 1 zurück. So auch bei Java. Um nun beispielsweise eine zufällige Zahl zwischen 3 und 11 zu erzeugen, multipliziert man den Zufallswert mit der Werte-Spannweite und addiert den Minimalwert. Sollen nur Ganzzahlen erzeugt werden, muss das Ergebnis noch gerundet werden. Um die folgenden Ausführungen verstehen zu können, bleibt festgehalten, dass immer ein Wert zwischen 0 und 1 erzeugt wird.

Wie bereits bei der Analyse von Musik in Abschnitt 2 festgestellt, bewegen sich die Melodie-Linien aller Lieder um ein tonales Zentrum herum. Um dies auch in den zufällig erstellten Stücken zu realisieren, wird die Länge der Tonsprünge, sowie die Richtung der Tonbewegung durch eine ungleichmäßige Häufigkeitsverteilung der Zufallsfunktion angepasst. Bei der Länge der Tonsprünge sollten so überwiegend kurze Tonsprünge erreicht werden. Um dies zu realisieren wird zunächst eine Funktion gesucht, die einen zufälligen  $x$ -Wert als Eingabe in einen ungleichmäßig verteilten Wert umwandelt. Dabei muss darauf geachtet werden, dass der neu zugeordnete Wert nicht den Wertebereich zwischen 0 und 1 verlässt. Die Gruppe der Potenzfunktionen der Form  $f(x) = x^n$  eignet sich dafür hervorragend, da für alle  $n$  gilt:  $f(1) = 1$ , sowie  $f(0) = 0$ . Durch den Wert von  $n$  wird der Verlauf der Kurve bestimmt. Im Fall der Tonsprünge eignet sich die Funktion  $f(x) = x^2$ . Sie ist in Abbildung 13 dargestellt. Wie an der Darstellung erkennbar ist, steigt der Funktionswert für kleine Werte von  $x$  zunächst langsam an. Nutzt man nun diese Funktion mit einem zufällig zwischen 0 und 1 liegendem  $x$ , bekommt man als Ergebnis mit einer höheren Wahrscheinlichkeit einen kleineren „Zufallswert“ um damit weiter zu arbeiten.

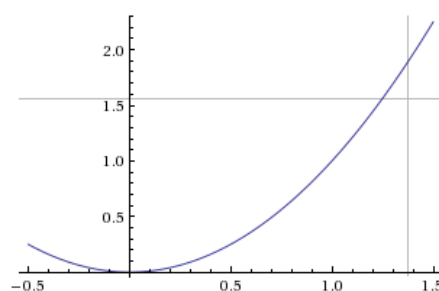


Abbildung 13: Die Funktion  $f(x) = x^2$

Durch dieses Konzept wird erreicht, dass die Tonsprünge eher klein sind und die Melodie nicht alternierend auf der Spannweite der möglichen Töne herumspringt. Weiterhin wird diese Methode benutzt, um auf- oder absteigende Tonfolgen und Folgen von gleichbleibenden Tonlängen zu erzeugen, wie sie in Liedern oft vorkommen. Hierzu wird die Potenz-Funktion abhängig von der Länge der bereits erzeugten Sequenz (gleicher Tonlängen bzw. auf-/ absteigender Töne) gemacht. Das heißt, das  $n$  entsprechend der Sequenzlänge angepasst wird. Dadurch kann erreicht werden, dass bei kurzer Sequenzlänge möglichst kein Wechsel der Richtung bzw. der Tonlänge erfolgt. Je länger die Tonlänge, desto wahrscheinlicher wird ein Wechsel. Dabei wurde der Wert, bei dem die Wahrscheinlichkeit gleich verteilt ist, auf eine Sequenzlänge von 4 festgesetzt. Dieser Wert resultiert aus statistischen Analysen der vorhandenen Lerndaten.

Ohne auf die Details der Erstellung der zufälligen Generate einzugehen, sollen in der folgenden Aufzählung noch einmal die Schritte der Erzeugung abgebildet werden:

1. Auswahl einer zufälligen Länge zwischen 20 und 50 Noten.
2. Auswahl eines zufälligen Grundtons.
3. Zufällige Auswahl der Tonrichtung zu Beginn. Dabei wurde eine ca. 70%-ige Wahrscheinlichkeit auf einen Auftakt gelegt. Da dies in den meisten Liedern der Fall ist.
4. Zufällige Auswahl der ersten Tonlänge.
5. Hinzufügen des Tons zum Lied.
6. Auswahl der Tonlänge des Folgetons. Dabei wird die Wahrscheinlichkeit von der Länge der Sequenz mit Tönen gleicher Länge vor dem aktuellen Ton beeinflusst.
7. Auswahl der Tonrichtung. Diese ist ebenfalls Abhängig von der Länge der bereits erzeugten Frequenz von auf- bzw. absteigenden Tönen.
8. Auswahl des Tonsprunges. Dieser wird durch eine statische Funktion, bei der kleine Tonsprünge wahrscheinlicher sind, beeinflusst.

9. Die Höhe des Tons wird berechnet: Höhe letzter Ton + Tonsprung \* Tonrichtung (1; -1)
10. Schritte 5 bis 9 werden wiederholt, bis die festgelegte Länge erreicht ist.

Durch dieses Verfahren können nicht besonders schlecht klingende Musikstücke erstellt werden und als Lern- bzw. Test-Daten in den Pool für die Prototypen aufgenommen werden.

## **6.4 Implementierung eines neuronalen Netzes**

Die Entwicklung des ersten Prototyps unterteilt sich in zwei Abschnitte: Zunächst wird die Entwicklung eines Java-Paketes beschrieben, mit dem es möglich ist Backpropagation Netze zu erstellen. Man kann dieses Paket durchaus als kleines Framework betrachten, da die Klassen des Paketes verschiedene Funktionen zur Erstellung eines Netzes zur Verfügung stellen. So ist es später leicht möglich ein Netz für einen bestimmten Anwendungsfall zu konfigurieren. Genau dies wird im zweiten Abschnitt auch getan. Denn dann wird das Framework genutzt, um ein Netz zu erstellen, welches die Daten aus den Song-Objekten verarbeiten kann und daraus lernen kann. Dabei werden wiederum zwei Variationen von möglichen Eingaben betrachtet, welche später in den Tests erläutert werden sollen.

### **6.4.1 Entwicklung eines Frameworks zur Erstellung neuronaler Netze**

Im Folgenden soll die Entwicklung des Frameworks schrittweise erläutert werden. Bereits in Kapitel 3.6 wurden der Aufbau und das Lernverfahren derartiger Netze kurz skizziert. Dabei wurde bereits erwähnt, dass ein vorwärtsgerichtetes Netz aus mehreren Schichten von Neuronen besteht. Es muss mindestens eine Ein- und eine Ausgabeschicht existieren. Die Anzahl der Schichten zwischen der Ein- und der Ausgabe ist theoretisch frei wählbar. Jedoch hat sich in der Praxis gezeigt, dass Netze mit mehr als zwei Schichten kaum praktikabel sind. Ohnehin können die meisten Probleme mit Netzen mit einer Zwischenschicht gelöst werden (Vgl. [Lämmel und Cleve (2008), Seite 227]). Der Aufbau eines Backpropagation Netzes ist in der folgenden Abbildung dargestellt.

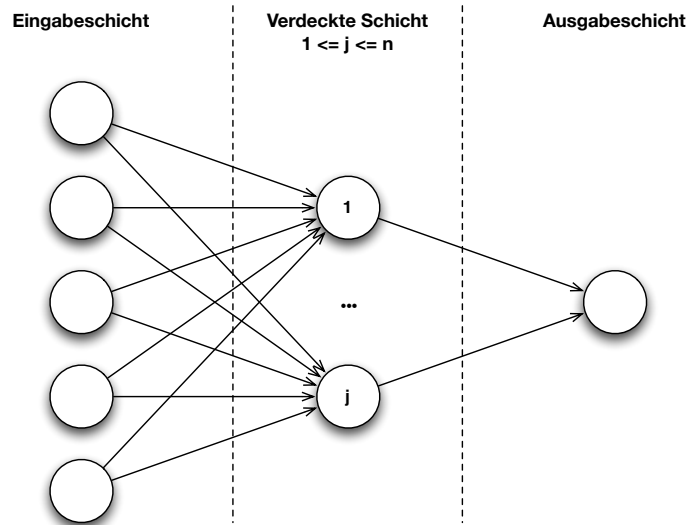


Abbildung 14: Architektur eines Backpropagation Netzes

Es zeigt sich, dass jedes Neuron mit den Neuronen seiner Folgeschicht verbunden ist. Dabei besitzt jede Verbindung ein Gewicht, welches während des Lernprozesses angepasst werden kann. Diese Architektur wurde zunächst in einer objektorientierten Software-Architektur umgesetzt. Abgeleitet aus der, in der Abbildung dargestellten, Architektur ergibt sich, die im folgenden UML-Diagramm dargestellte, Objekt-Struktur. Das Diagramm enthält dabei bereits die Methoden, welche zum Aufbau des Netzes nötig sind. Auf diese soll nicht näher eingegangen werden, da die Namen selbsterklärend sind und die Funktionsweise sehr simpel ist.

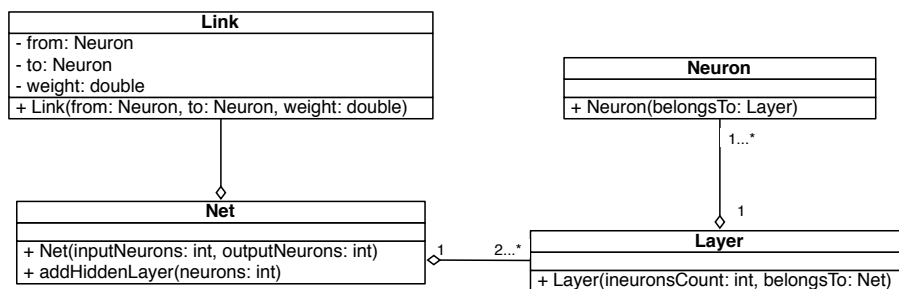


Abbildung 15: Ein neuronales Netz als objektorientiertes Klassen-Modell

Mit den vorhandenen Klassen kann bereits ein einfaches Netz aufgebaut werden. Dabei besitzt der Entwickler die Freiheit, Netze mit beliebig vielen Zwischenschichten, mit

jeweils beliebig vielen Neuronen zu erzeugen. Auch die Anzahl der Ein- und Ausgabe-Neuronen ist frei wählbar. Im nächsten Schritt muss diese Architektur nun um Funktionalität erweitert werden, sodass das Netz arbeiten kann.

In den Grundlagen in Kapitel 3.6 wurde bereits die Funktionsweise eines natürlichen Neurons aufgezeigt. Diese muss jetzt in den künstlichen Neuronen nachgebildet werden. Wie die eines natürlichen Neurons, basiert die Ausgabe eines künstlichen Neurons auf einer Menge von Eingabewerten. Das natürliche Neuron besitzt zum Empfangen der Eingabewerte die Dendriten (Siehe Abbildung 5, Seite 29). Beim künstlichen Neuron werden die Dendriten durch die gewichteten Verbindungen von den Neuronen der Vorgängerschicht zum Neuron repräsentiert. Um die eingehenden Werte zu bearbeiten wird dem Neuron eine *propagate()*-Methode hinzugefügt. Diese Methode kombiniert die Propagierungs- und die Aktivierungsfunktion und stellt damit das Pendant zum chemischen Prozess der Natur her. Die Propagierungsfunktion fasst alle Eingabewerte zusammen. In den meisten Fällen geschieht das durch einfaches summieren der Eingabewerte, multipliziert mit den jeweiligen Gewichten (Vgl. [Kruse u. a. (1991), Seite 99; Lämmel und Cleve (2008), Seite 198]) - Diese Arbeitsweise soll auch hier eingesetzt werden. Die Propagierungsfunktion bzw. ihr Ergebnis wird üblich mit der Netzeingabe gleich gesetzt. Die Aktivierungsfunktion nutzt die Netzeingabe, um den Ausgabewert des Neurons zu berechnen. Das Ergebnis der Aktivierungsfunktion (und damit der Ausgabewert des Neurons) sollte zwischen 0 und 1 liegen, inklusive der Grenzwerte. Dies kann durch eine einfache Schwellwert-Funktion realisiert werden. Eine solche Funktion kann sehr einfach formuliert werden:

```
if (Netzeingabe > 1.5) Ausgabe = 1; else Ausgabe = 0;
```

---

Listing 2: Eine einfache Schwellwert-Funktion

In der Praxis macht eine solche Funktion jedoch selten Sinn (Vgl. [Kriesel (2005), Seite 71]). Neben einer solchen Schwellwert-Funktion kommen daher noch eine Lineare-Funktion oder eine sigmoide Funktion in Frage. In aller Regel wird eine sigmoide Funktion als Aktivierungsfunktion genutzt. In nahezu allen neuronalen Netzen kommt die logistische Funktion, als Form einer sigmoiden Funktion, zum Einsatz (Vgl [Kriesel (2005), Seite 71; Lämmel und Cleve (2008), Seite 199; Kruse u. a. (1991), Seite 101]. Die Funktion ist in der folgenden Abbildung dargestellt.

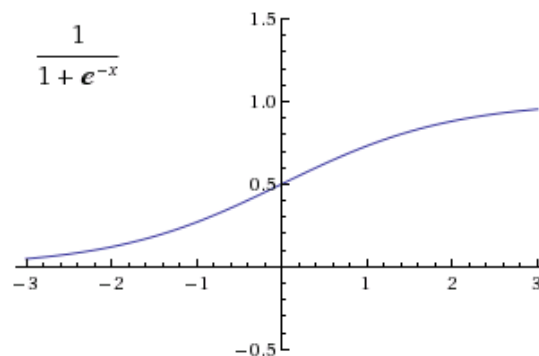


Abbildung 16: Eine sigmoide Funktion als Aktivierungsfunktion.

Im Diagramm kann man erkennen, warum sich diese Funktion besonders gut für den Einsatz als Aktivierungsfunktion eignet: Die Funktion nähert sich in Richtung minus unendlich dem Wert 0 und in positiver Richtung dem Wert 1. Die Funktionswerte liegen also stets zwischen 0 und 1. Da diese Funktion offensichtlich am besten geeignet ist, kommt sie auch in der *propagate()*-Methode des Neuron-Objektes zum Einsatz. Weiterhin erhält das Netz-Objekt unter anderem eine *apply()*-Methode, welche die Neuberechnung aller Neuronen anstößt. So ist das Netz in diesem Status bereits in der Lage einen Ausgabewert zu berechnen.

Im letzten Schritt muss der Lern-Mechanismus in das Framework implementiert werden. Für den Lernmechanismus wurde ebenfalls auf verbreitete Verfahren zurück gegriffen. Ohnehin gibt ein Backpropagation-Netz den Lern-Algorithmus bereits vor, schließlich basiert die Bezeichnung des Netzes auf dem Name des Algorithmus. Der Backpropagation-Algorithmus ist vereinfacht im folgenden Listing formuliert:

```
backpropagate() {
    nZyklen = 0;
    wiederhole {
        fehler = 0;
        nZyklen++;
        for (i .. AnzahlDerDatensaetze) {
            Eingabewerte aus Datensatz[i] anlegen;
            for (j .. AnzahlDerAusgabeNeuronen) {
                Bestimme Ausgabewert oj;
                Bestimme tj = Erwartetes Ergebnis;
```

```

        Bestimme Fehlerwert  $f_j = t_j - o_j$ ;
        Bestimme Fehlersignal  $fsig_j = o_j * (1 - o_j) * f_j$ ;
        fehler +=  $f_j^2$ ;
    }

    for (s = Ausgabeschicht - 1 .. ErsteSchicht) {
        for (k .. AnzahlDerNeuronenInSchicht) {
            fsum = 0;
            for (m .. AnzahlDerNeuronenInSchicht[k+1])
                fsum = fsum + w[k,m] * fsig[s+1,m];
            fsig[s,k] = o[s,k] * (1 - o[s,k]) * fsum;
            for (m .. AnzahlDerNeuronenInSchicht[k+1])
                w[k,m] = w[k,m] + Lernrate * o[s,k] * fsig[s+1,m];
        }
    }
} while (fehler > tolerierterFehler || nZyklen erreicht)
}

```

Listing 3: Der Backpropagation-Algorithmus

Der Algorithmus verändert die Gewichte der Verbindungen zwischen den Neuronen, ausgehend von der Ausgabeschicht des Netzes. Dort kann der Fehler initial ermittelt werden, da das Ergebnis der Ausgabeschicht mit dem erwarteten Ergebnis verglichen werden kann. Nachdem der initiale Fehler berechnet wurde, können die Fehlersignale der davor liegenden Schichten, Schicht für Schicht, erschlossen werden und die Gewichte entsprechend angepasst werden.

Im Framework wurde der Lern-Algorithmus in einem neuen Objekt, dem *Trainer*-Objekt implementiert. Das Trainerobjekt verwaltet außerdem die Lern-Datensätze; Es ist auch in der Lage das trainierte Netz mit Test-Datensätzen zu testen. Komplettiert wird das Framework durch ein Interface, welches als Schnittstelle zur Anwendung dient, welche das Netz nutzt. Das gesamte Paket ist vereinfacht im nachstehenden UML-Diagramm abgebildet.

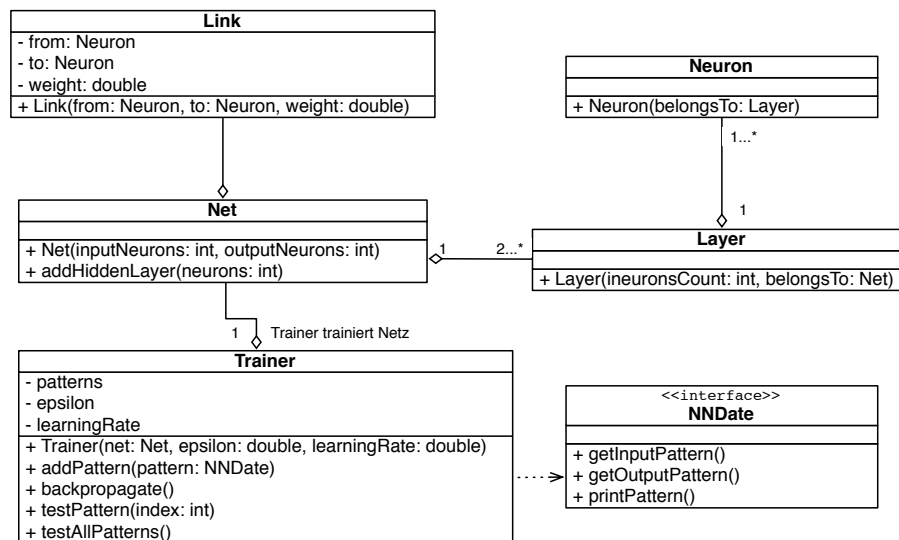


Abbildung 17: Eine vereinfachte Darstellung des Frameworks

Die genaue Umsetzung des Backpropagation-Algorithmus in der objektorientierten Architektur und die Funktionsweise der einzelnen Methoden soll hier nicht weiter beschrieben werden. Das würde den Rahmen der Arbeit sprengen. Ebenfalls sei für weitere Erläuterungen des Algorithmus, sowie mathematische Beweise der Arbeitsweise, auf [Lämmel und Cleve (2008)], [Kriesel (2005)] oder [Ritter u. a. (1991)] verwiesen.

#### 6.4.2 Aufbau eines Netzes zur Analyse von Musik

Bevor ein Netz mit Hilfe des Frameworks aufgebaut werden kann, müssen die Eingabewerte für das Netz definiert werden. Als mögliche Eingaben wurden hauptsächlich zwei Ansätze betrachtet:

- Eingeben einzelner Melodie-Fragmente
- Eingeben von vorher analysierten Merkmalen des Musikstücks.

Beide Ansätze werden im folgenden genauer erläutert.

Das Nutzen der Melodie als Eingabe erscheint sinnvoll, da auch bei sprachverstehenden Systemen ähnlich vorgegangen wird. Zwar werden bei sprachverstehenden Systemen

Audio-Signale, also letztendlich das Frequenzband, genutzt. Jedoch ist die Notendarstellung einer Melodie, wie sie im MusicStream-Format vorliegt, durchaus mit einem Frequenzband vergleichbar. Beide Formate stellen die Veränderung der Tonhöhe über einen gewissen Zeitraum dar. So ist es also durchaus denkbar, dass auch hier die Tonhöhe zu verschiedenen Zeitpunkten in das Netz eingegeben wird. In diesem Zusammenhang spricht man auch von *Time Delay Neutral Networks*, da ein über die Zeit veränderlicher Messwert eingegeben wird (Siehe auch Seite 35).

Da das Netz nur Eingabewerte zwischen 0 und 1 akzeptiert, muss für die Umsetzung des Ansatzes ein Weg gefunden werden, die Tonhöhe auf einen entsprechenden Wert zu transformieren. Hierfür eignet sich der numerische Wert der Note. Sowohl im MIDI, als auch im MusicString-Format hat jede Note einen Wert zwischen 0 und 127. Dieser Wert kann für eine einfache Umrechnung, wie im folgenden Listing dargestellt, genutzt werden.

```
if (Note == Pause)
    Netzeingabe = 0;
else
    Netzeingabe = 1 / (Notenwert + 1)
```

Listing 4: Normalisieren des Notenwertes

Damit läge der Eingabewert für Tonhöhe stets zwischen 0 und 1. Er wäre also für die Eingabe in ein Netz geeignet. Um die Melodie als Netzeingabe zu vervollständigen, muss auch die Notenlänge in das Netz eingegeben werden. Bei dieser kann direkt der mathematische Ausdruck der Notenlänge verwendet werden. Schließlich ist jede mögliche Notenlänge ein Teil von 1 (Ganze Note =  $1/1$ , Halbe Note =  $1/2$ , u.s.w.). So ergeben sich für jeden Zeitpunkt, der in das Netz eingegeben werden soll, zwei Eingabewerte. Sinnvollerweise werden als Zeitpunkte die jeweiligen Anschläge eines neuen Tons gewählt, sodass das Netz eine komplette Notenfolge als Eingabe erhält. Alternativ könnte auch ein regelmäßiger Zeitabstand, zum Beispiel jedes Achtel, genutzt werden. Zwar könnte man hier die explizite Tonlänge als Netzeingabe weglassen, jedoch erscheint dieser Ansatz als weniger praktikabel.

Bevor das Netz erzeugt werden kann, muss noch eine einheitliche Sequenzlänge festgelegt werden. Da das gleiche Netz für alle Lieder genutzt werden soll und alle Lieder unterschiedlich lang sind, kann nicht die gesamte Melodie eingegeben werden. Außerdem

zeigen Experimente, dass eine zu große Anzahl an Eingabeneuronen dazu führt, dass das Netz nicht terminiert, also keine brauchbaren Gewicht-Verhältnisse findet (Siehe auch [Reichardt (2010)]). Da im Vornherein keine Aussage darüber getroffen werden kann, wird die Länge ohne weitere Überlegungen auf 4 Töne festgesetzt. 4 Töne ergeben bereits eine Eingabeschicht mit 8 Eingabeneuronen.

Nachdem alle Merkmale des Netzes feststehen kann das Netz mit Hilfe des Frameworks erstellt werden. Hierzu wird zur Laufzeit eine Instanz der *Net*-Klasse erstellt. Das Netz wird mit einer Eingabeschicht aus 8 und einer Ausgabeschicht aus einem Neuron initialisiert. Danach wird eine weitere Schicht als versteckte Schicht hinzugefügt. Die versteckte Schicht bekommt zunächst 6 Neuronen. Dieser Wert kann bei späteren Testläufen verändert werden. Ist das Netz angelegt, kann eine Instanz der Klasse *Trainer* erstellt werden. Dieser Instanz werden die vorbereiteten Lern- und Testdaten bereitgestellt. Zur Vorbereitung der Daten, zum Beispiel dem extrahieren der ersten Töne aus einem *Song*-Objekt, wird eine Hilfsklasse genutzt, welche das Interface *NNDat*e implementiert. Sind dem Trainer die Daten bekannt, kann das Netz angelernt und getestet werden.

Auch an dieser Stelle soll nicht auf die technischen Details eingegangen werden.

Bevor im nächsten Abschnitt die Tests erläutert werden, soll noch kurz auf die Modellierung des Netzes eingegangen werden, welche die Merkmale eines Musikstückes als Eingabewerte annimmt. Die Nutzung von zuvor analysierten Merkmalen als Netzeingabe ist auf Basis der vorhandenen Komponenten sehr einfach. Da die Song-Komponente bereits in der Lage ist Musikstücke zu analysieren und statistisch auszuwerten, können die ermittelten Werte der Komponente einfach normalisiert und als Eingabe verwendet werden. Dieser Ansatz erscheint deswegen erfolgsversprechend, weil sich bei den ersten statistischen Analysen, welche während der Entwicklung der Song-Komponente gemacht wurden, durchaus wiederkehrende Muster in den analysierten Merkmalen wieder fanden.

### 6.5 Tests am neuronalen Netz

Im Datei-Anhang der Arbeit finden sich einige Auszüge der Testausgabe. An dieser Stelle sollen die Erkenntnisse lediglich umschrieben werden.

Die ersten Tests mit der Melodie als Netzeingabe wurden mit einem Epsilon von 0,5 durchgeführt. Epsilon beschreibt dabei den akzeptierten Fehler. Das heißt, dass der Lernprozess beendet wird, sobald der Gesamtfehler beim Test aller genutzten Lern-Daten den Wert nicht übersteigt. Dabei muss davon ausgegangen werden, dass der Fehler bei neuen Daten höher sein wird. Normalerweise ist ein Epsilon von 0,5 für das Lernverfahren eines Netzes sehr hoch. Da aber die Ausgabewerte auf 0 und 1 beschränkt sind, liegt hier eine große Spannweite vor, sodass bereits ein Ergebnis von 0,8 als eindeutig angesehen werden kann. Als Lernrate wurde ebenfalls zunächst ein Wert von 0,5 gewählt. Die Lernrate beeinflusst die Geschwindigkeit des Lernverfahrens, ist sie zu klein gewählt, dauert der Lernprozess sehr lange. Ist sie zu groß, „schwingen“ die Gewichte der Verbindungen und das Netz terminiert nicht.

Für die Tests wurden zunächst 40 Musikdateien als Lerndaten verwendet. 10 weitere als Test-Daten. Beide Gruppen bestanden zu gleichen Teilen aus echter und generierter Musik.

In den ersten Versuchen terminierte das Lernverfahren des Netzes nicht. Die Veränderung der Lernrate, sowie des Epsilon-Wertes führten zwar zu funktionierenden Lernprozessen. Das Ergebnis war jedoch unbefriedigend. So konnte nur knapp die Hälfte der Testdaten richtig erkannt werden. Nach Analyse des Lernprozesses wurde das Normalisierungsverfahren der Melodie als Grund für dieses unbefriedigende Ergebnis vermutet. Die Eingabewerte der Melodie lagen alle sehr nah bei einander. Durch die Normalisierung der Tonhöhe von 127 Werten auf den Bereich zwischen 0 und 1 waren die Werte kaum zu unterscheiden. Für das Netz waren bei den Eingaben also nur kleine Unterschiede im Muster zu erkennen. Aus diesem Grund wurde die Netzeingabe verändert: So wurde nun nicht mehr der Tonwert normalisiert, sondern nur noch die Differenz zum vorherigen Ton. Hierfür wurde ein Verfahren angewendet, welches ähnlich zum Verfahren der Fuzzy-Logik, wie es auf Seite 21 beschrieben wurde, ist.

## Kapitel 6 Realisierung

So wurde festgelegt, dass

- Einem Tonsprung von 22 oder mehr Tönen nach „oben“ der Funktionswert 1,
- einem Tonsprung von 22 oder mehr Tönen nach „unten“ der Funktionswert 0 zugeordnet wird.

Alle dazwischen liegenden Tonsprünge teilen sich entsprechend auf den verbleibenden Wertebereich auf. So wurde der zu normalisierende Wertebereich um rund zwei Drittel gesenkt. Außerdem kann so mit Hinzufügen eines weiteren Neurons ein weiterer Ton der Melodie als Eingabe verwendet werden: Da das „Tonsprung“-Neuron des ersten Tons immer 0 ist, kann es vernachlässigt werden und somit nur die Länge des Tones eingegeben werden. Die umgebaute Struktur des Netzes ist noch einmal in der unten stehenden Abbildung dargestellt.

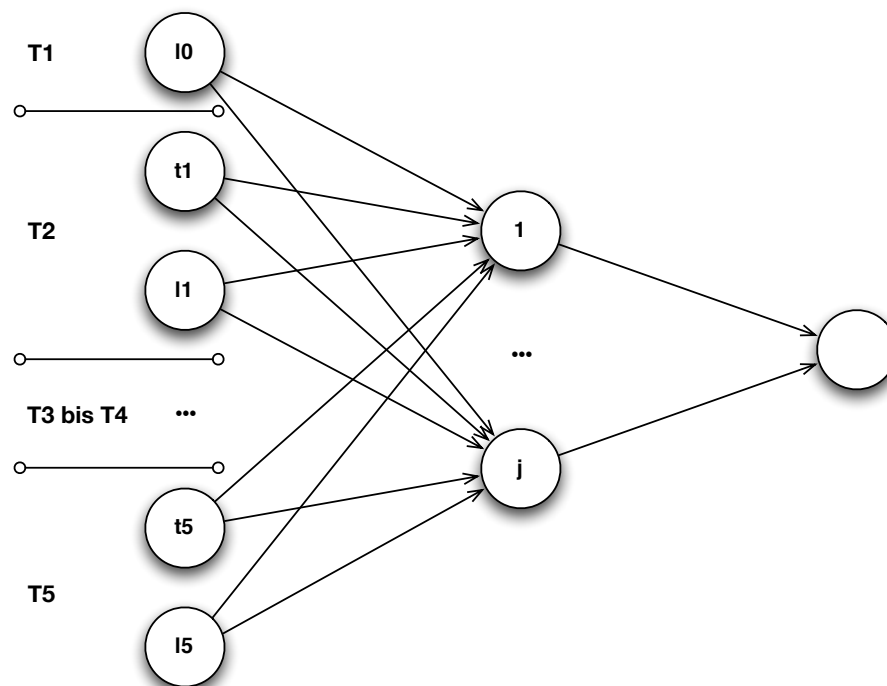


Abbildung 18: Die veränderte Architektur des Netzes

Der Umbau der Netzarchitektur brachte bestechenden Erfolg. Bereits im ersten Versuch lernte das Netz innerhalb von wenigen tausend Zyklen. Im Testlauf schätzte es nur eines der 10 Lieder falsch ein. Dieses war *Yesterday*, was aber tatsächlich auch ein

ungewöhnlichen Melodieverlauf in den ersten Tönen nimmt - Es enthält zwei nicht zur Tonart gehörende Töne und beginnt nicht mit dem Grundton. Möglicherweise ist dies der Grund für die Fehleinschätzung.

In weiteren Versuchen wurden die Werte für die Lernrate und Epsilon verändert. Dies änderte jedoch kaum etwas am Ergebnis. Auch die Reduzierung der Lern-Daten auf 25 Datensätze und die damit verbundene Erhöhung der Test-Daten verlief erfolgreich. So wurden von 25 Test-Daten 3 Lieder nicht entsprechend den Erwartungen bewertet. Darunter wiederum *Yesterday*, sowie zwei weitere generierte Songs, welche eventuell durchaus als gut einzuschätzen sind - Zumindest in den ersten Tönen.

Die Analyse der Melodie war also durchaus erfolgreich. Ein vergleichbares Ergebnis wurde mit dem zweiten Eingabemuster erreicht, welches vorab erarbeitete Merkmale in das Netz eingab. Der Grund hierfür dürfte in der statistisch klaren Verteilung der Werte liegen. So ergab sich für die richtigen Musikstücke ein klares Muster und die Mustererkennung gehört schließlich zu den Haupteinsatzgebieten von Backpropagation Netzen. Warum jedoch die Melodie-Analyse anhand von nur 5 Tönen so gut funktioniert, kann an dieser Stelle nicht begründet werden. Genauere Analysen des Ergebnisses waren im Rahmen dieser Studienarbeit nicht möglich. Eventuell kann dieses Ergebnis jedoch Ausgangspunkt für weitere Arbeiten sein.

### 6.6 Implementierung eines einfachen wissensbasierten Systems

Aufgrund der bereits vorhandenen Werkzeuge zur Analyse von Merkmalen eines Musikstückes, ist die Implementierung des wissensbasierten Systems im Vergleich zur Realisierung des Neuronalen Netzes, deutlich weniger aufwendig. Wie die einzelnen Komponenten umgesetzt wurden, sei im Folgenden umschrieben:

- **Benutzerschnittstelle** Für die Benutzerschnittstelle konnte die bereits vorhandene Hauptkomponente der Prototypen Umgebung verwendet werden.
- **Wissensbasis** Die Wissensbasis ist in zwei Teile unterteilt: Eine Regel- und eine Fallbasis. Die Fallbasis nutzt im Ausgangszustand die Sammlung der vorhandenen Lern-Daten. Auf Basis dieser Daten werden zunächst Statistiken über die verschiedenen Merkmale wie Tonumfang, Anzahl und Verteilung der verschiedenen Töne,

Anzahl und Länge von wiederholenden Sequenzen und anderen Eigenschaften, erstellt. Außerdem kann ein Song-Objekt erstellt werden, welches alle Songs vereint. Dieses Objekt kann später genutzt werden, um leicht parallelen zwischen einem zu testenden Musikstück und der gesamten Fallbasis zu erkennen.

Die Wissensbasis wird ergänzt durch eine Funktionalität, welche den Song auf Zugehörigkeit der Töne zur Tonart kontrollieren kann. Da auch diese Funktion bereits bei der Entwicklung des MusicStream-Generators umgesetzt wurde, kann sie auch hier ohne großen Aufwand genutzt werden.

- **Inferenzmaschine** Die Inferenzmaschine wird durch einfache Algorithmen dargestellt. Diese vergleichen die statistischen Werte der Wissensbasis mit den Werten des eingegebenen Musikstücks oder untersuchen die Musikstücke hinsichtlich ihrer Ähnlichkeit - Hierzu wird der in Abschnitt vorgestellte Algorithmus eingesetzt – Bei der Beschreibung der Tests, werden später verschiedene Ansätze betrachtet. Nachdem ein Lied abgearbeitet wurde und mit dem erwarteten Ergebnis abgeglichen, ggf. berichtigt wurde, kann das Lied in die Fallbasis aufgenommen werden. Bisher wurde allerdings noch keine Möglichkeit gefunden bzw. umgesetzt, die neue Regeln aufgrund der Wissensbasis ableiten kann.
- Ein Editor für die Wissensbasis ist für diesen simplen Prototyp nicht nötig, da er alle Daten beim Programmstart neu einließt. Soll die Wissensbasis vergrößert werden, kann das einfach durch Hinzufügen neuer Daten im Dateisystem erfolgen.
- Auch ein Erklärungssystem wurde nicht implementiert.

Dieser Prototyp stellt damit lediglich rudimentär die Komponenten eines Wissensbasierten-Systems dar. Auch konnte der Prototyp während des Projektzeitraums nicht zu einem kompletten lauffähigen Programm umgesetzt werden. So basieren die Tests hauptsächlich auf einfachen Debugging-Ausgaben. Dennoch konnten durch diesen simplen Prototypen bereits einige Erkenntnisse gewonnen werden, welche im nächsten Abschnitt erläutert werden sollen.

### 6.7 Tests am wissensbasierten System

Die Tests, welche mit den Komponenten des wissensbasierten Systems durchgeführt wurden, waren sehr rudimentär und wurden nicht genauer aufgezeichnet. Im Dateianhang der Arbeit finden sich aber auch zu diesen Tests einige Beispiel-Ausgaben.

Während der Tests hat sich herausgestellt, dass einige Merkmale deutlich aussagekräftiger sind als andere. Beispielsweise kann der Zugehörigkeit der Töne zur Tonart, zumindest beim bisher genutzten Analyse Verfahren, keine große Bedeutung zugesprochen werden. So enthielten die richtigen Musikdateien durchaus einen großen Anteil an Noten, die scheinbar nicht zur Tonart dazu gehören. Dieses Phänomen kann verschiedene Ursachen haben. Unter anderem basiert die Tonart-Prüfung auf der Annahme, dass der erste Ton der Grundton ist. Dies ist jedoch in vielen Fällen, wie sich im Verlauf der Testreihen herausgestellt hat, nicht der Fall. So sollten nachfolgende Versionen besser die gegebenen Vorzeichen des Notensatzes zur Bestimmung der Tonart heranziehen. Der Verzicht der Tonart-Kontrolle bei der Erstellung der generierten Musik hat dennoch nicht dazu geführt, dass das System größere Fehleinschätzungen getroffen hat.

Vielmehr haben sich die „undefinierten“ Regeln, die nicht durch die Musiktheorie beschrieben werden, als entscheidende Faktoren herausgestellt. So stimmt die prozentuale Häufigkeitsverteilung, der am meisten auftretenden Töne in den richtigen Musikstücken, prozentual stark überein. Auch die Anzahl an redundanten Rhythmus-Stellen der echten Lieder liegt im Mittel bei 60% und mehr.

So kann Häufig bereits auf Basis von einfachen statistischen Analysen eine Entscheidung getroffen werden. Der Vergleich eines Test-Musikstückes mit den in der Wissensbasis vorhandenen Stücken brachte zunächst im Einzelvergleich wenig Übereinstimmungen. Dies kann daran liegen, dass die breite der Musikstücke sehr vielfältig gewählt wurde. Wäre ausschließlich Pop-Musik in der Wissensbasis oder wären die Daten alles Songs eines Künstlers, hätte man wahrscheinlich größeren Erfolg erzielen können. Der Vergleich eines Musikstückes mit einem Musikstück, welches aus allen in der Basis vorhandenen Stücken besteht, erwies sich jedoch als sehr erfolgreich. So konnten bei richtigen Liedern große Übereinstimmungen gefunden werden, bei erzeugten Liedern jedoch nicht - Trotz der Mechanismen, welche regelmäßige und natürlich aussehende Sequenzen erzeugen.

Zusammenfassend kann man behaupten, dass der Prototyp in der Lage ist, anhand von

## Kapitel 6 Realisierung

---

einfachen Analysen auf einer Wissensbasis und durch unterschiedliche Vergleiche, generierte Musik von richtiger Musik zu unterscheiden - Hierfür müssen jedoch die richtigen Faktoren betrachtet werden. Ein Mechanismus, der diese Faktoren selbst erkennt, wäre in einem „richtigen“ wissensbasierten System jedoch Pflicht. Im hier vorgestellten Prototyp wurde diese Entscheidung durch programmatische Festlegungen und durch starre Algorithmen getroffen.

# 7 Zusammenfassung und Ausblick

Im Verlauf des Projektzeitraums wurde ein breites Spektrum an verschiedenen Forschungsgebieten beleuchtet. Angefangen von Grundlagen der Musiktheorie über verschiedene Gebiete der künstlichen Intelligenz bis hin zu technischen Fragestellung über die digitale Verarbeitung von Musik. Ein Großteil der Erkenntnisse und Gedankengänge wurde in dieser Arbeit erläutert, oder zumindest angeschnitten.

Aufgrund der großen Breite des Themengebietes und der beschränkten Laufzeit des Projektes, war es nicht möglich noch tiefere Einblicke in verschiedene Bereiche zu finden. Dennoch wurde das Ziel der Arbeit, welches auch in der Einleitung formuliert wurde, erreicht: Mit einem neuronalen Netz und einem einfachen wissensbasierten System stehen nicht nur ein, sondern gleich zwei Prototypen eines intelligenten Systems zur Unterscheidung zwischen automatisch generierter und von Menschen komponierter Musik als Ergebnis da. Damit wurden die Grundlagen für weitere Forschungen auf diesem Gebiet gestellt.

Die Funktionsfähigkeit des regel- und fallbasierten wissensbasierten Systems beweist, dass „gute“ Melodien oft ähnlich zu bekannten Melodien sind; Es wurde damit belegt, dass Melodien, die „gefallen“, bestimmte Regeln befolgen. Regeln die aber auch über normale Musiktheorie hinaus gehen - Die automatisch generierte Musik, welche diese Regeln befolgt, wurde von beiden Systemen förmlich „entlarvt“. Dieses Ergebnis lässt auf Parallelen zu ähnlichen Forschungsfeldern schließen. Untersuchungen zum Schönheitsempfinden von Menschen untereinander führte ebenfalls zu dem Ergebnis, dass das bekannte und der absolute Durchschnitt als schön empfunden wird.

Vor allem in der Grundlagenarbeit dieses Projektes wurden noch viele weitere Ansätze betrachtet, die in dieser Arbeit später nicht weiter bearbeitet wurden. So resultieren aus dieser Arbeit auch neue Herausforderungen für fortführende Arbeiten. So könnte beispielsweise der Bezug zwischen Melodie und Inhalt analysiert werden. Auch ist es vorstellbar, dass originale Audio-Daten verarbeitet werden, anstatt stark vereinfachte MIDI bzw. MusicString-Daten. Nur durch die Erweiterung des Analyse-Feldes können letztendlich weitere Schritte unternommen werden, um die Ziele zu erreichen welche als Motivation zu dieser Arbeit gedient haben. So muss neben der eigentlichen Komposition

mindestens auch noch die Spielweise, sowie der Inhalt eines Musikstückes untersucht und analysiert werden, um zu entscheiden, ob einem Menschen der Titel wirklich gefällt. Denn wie in den Grundlagen herausgefunden wurde, spielen diese zwei Komponenten ebenfalls eine wichtige Rolle.

Aber nicht nur in Richtung neuer Projekte kann weitere Forschung betrieben werden, sondern auch die konkreten Ergebnisse dieser Arbeit könnten noch weiter analysiert werden. So konnten bis zum Ende der Arbeit keine ausführlichen Testreihen mit den Prototypen durchgeführt werden. Die Einschätzungen und Analysen können deswegen nicht eindeutig durch ausreichende Messreihen belegt werden. Für Folgeprojekte, die direkt auf den in dieser Arbeit erstellten Prototypen basieren, ist eine genaue Analyse der Funktionsweisen und eine Auswertung von breiteren Testreihen jedoch unabdingbar. Aus diesem Grund werden auch alle bisherigen Ergebnisse, sowie der Quellcode veröffentlicht.

Zusammenfassend kann gesagt werden, dass die Arbeit durchaus erfolgreich verlaufen ist und zu einigen neuen Erkenntnissen geführt hat. Aber auch die Sammlung der Grundlagen kann für ähnliche Projekte von großer Bedeutung sein.

## Anhang

### Anhang A - Lern- und Testdaten

In der folgenden Tabelle sind die Musikstücke aufgeführt, welche als Lern- bzw. Testdaten bei der Entwicklung der Prototypen genutzt wurden. Als Vorlage diente [Grote (1999)].

<b>Titel</b>	<b>Komponist (der Melodie)</b>
Blowin' in the wind	Bob Dylan
Das Lied der Deutschen	Joseph Haydn
Das Wandern ist des Müllers Lust	Carl Zöllner
Der Mond ist aufgegangen	Johann Abraham Peter Schulz
Freude, schöner Götterfunken	Ludwig van Beethoven
Greensleeves	Volkswaise
Hejo, spann den Wagen an	Volkswaise
Hey Jude	John Lennon, Paul Mc Cartney
I like the flowers	Volkswaise
In Frühtau zu Berge	Volkswaise
In the summertime	Ray Dorset
Lady in black	Ken Hensley
Love me tender	Elvis Presley, Vera Matson
Mein kleine grüner Kaktus	Bert Reisfeld, Albrecht Marcuse
Morning has broken	Cat Stevens (Yusuf Islam)
My Bonny is over the ocean	Volkswaise
Oh du fröhliche	Volkswaise
Ohne dich	Stefan Zauner, Aron Strobel
Sah ein Knab ein Röslein stehn	Heinrich Werner

<b>Titel</b>	<b>Komponist (der Melodie)</b>
Sailing	Gavin Sutherland
Santa Claus is coming to town	J. Fred Coots
Süßer die Glocken nie klingen	Volkswaise
This land is your land	Woody Guthrie
Wann wird's mal wieder richtig Sommer?	Steve Goodman
Yesterday	John Lennon, Paul McCartney

## **Anhang B - Projektdateien**

Zu dieser Arbeit existiert ein Datei-Anhang, dieser enthält zum einen den Java-Quellcode der Prototypen, sowie einige Auszüge der Testdaten. Die Dateien stehen unter

<http://files.michaelwellner.de>

zur Verfügung. Alle Dateien sind in einem Zip-Archiv verpackt, welches wiederum als Eclipse<sup>9</sup>-Projekt importiert werden kann. Im Ordner *res* befinden sich die Lern- und Testdaten, sowie die Testergebnisse.

---

<sup>9</sup>Infos und Download von Eclipse unter <http://www.eclipse.org>

## Literatur

- [zu Bexten 2008] BEXTEN, E. M. zu: *Vorlesungs-Skript Interaktive Systeme*. 2008
- [Bollinger 2011] BOLLINGER, T.: *Data Warehouse, Data Mining (Vorlesungsskript)*. 2011
- [Dewald 2004] DEWALD, U.: *Mathematik des Wohlklangs*. 2004. – URL <http://www.wissenschaft.de/wissenschaft/news/242155.html>
- [Elmenreich 2011] ELMENREICH, W.: *Jordan-Netze*. 2011. – URL [http://de.wikipedia.org/w/index.php?title=Datei:DiagramJordanNet\\_deutsch.png&filetimestamp=20070710121749](http://de.wikipedia.org/w/index.php?title=Datei:DiagramJordanNet_deutsch.png&filetimestamp=20070710121749)
- [Fritzke 1998] FRITZKE, B.: *Vektorbasierte neuronale Netze*. Shaker, 1998
- [G.F.Luger 2001] G.F.LUGER: *Künstliche Intelligenz*. Addison-Wesley, 2001
- [Grote 1999] GROTE, Manfred: *Liederbuch für die Schule*. 2. Auflage. Volk und Wissen, 1999 (ISBN-13: 978-3-06-150525-7)
- [IBM 2011] IBM: *Watson - Antworten mit System*. IBM Systems and Technology. 2011
- [ics 2011] ICS: *Watson lässt Quizkönige alt aussehen*. 2011. – URL <http://www.spiegel.de/netzwelt/gadgets/0,1518,745831,00.html>. – Zugriffsdatum: 16. Februar 2011
- [Kriesel 2005] KRIESEL, D.: *Ein kleiner Überblick über Neuronale Netze*. 2005
- [Kriz 2010] KRIZ, J.: *KI Script: Grundlegende Konzepte*. 2010. – URL [http://www.nosco.ch/ai/script\\_08\\_01\\_01.php](http://www.nosco.ch/ai/script_08_01_01.php)
- [Kruse u. a. 1991] KRUSE, H. ; MANGOLD, R. ; MECHLER, B. ; PENDER, O.: *Programmierung Neuronaler Netze*. Addison-Wesley, 1991 (ISBN 3-89319-293-X)
- [Lämmel und Cleve 2008] LÄMMEL, U. ; CLEVE, J.: *Künstliche Intelligenz*. 3. Auflage. Carl Hanser Verlag München, 2008 (ISBN 978-3-446-41398-6)

- [v. Lienen 2002] LIENEN, E. v.: *Neuronale Netze in der Robotik*. 2002
- [REDAKTION 2003] REDAKTION: *Komprimierte Musikdateien verraten den Komponisten*. 2003. – URL <http://derstandard.at/standard.asp?id=1271917>
- [Reichardt 2010] REICHARDT, D.: *Expertensysteme*. 2010
- [Reif und Reschenhofer 2005] REIF, M. ; RESCHENHOFER, P.: *Kartenerstellung und Navigation mit einem autonomen mobilen Roboter mit Hilfe künstlicher Intelligenz*. 2005. – URL <http://fbim.fh-regensburg.de/~saj39125/Diplomarbeiten/ReschReif/node6.html>
- [Ritter u. a. 1991] RITTER, H. ; MARTINETZ, T. ; SCHULTEN, K.: *Neuronale Netze - Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke*. Addison-Wesley, 1991 (ISBN 3-89319-131-3)
- [Russel und Norvig 2004] RUSSEL, S. ; NORVIG, P.: *Künstliche Intelligenz - Ein moderner Ansatz*. 2. Auflage. Pearson Education Deutschland, 2004 (ISBN 3-8273-7089-2)
- [Sagerer 1990] SAGERER, G.: *Automatisches Verstehen gesprochener Sprache*. Mannheim ; Wien ; Zürich : BI-Wiss.-Verl., 1990 (ISBN 3-411-14391-6)
- [Schöning 2000] SCHÖNING, Uwe: *Logik für Informatiker*. Spektrum Akademischer Verlag, 2000 (ISBN 978-3-8274-1005-4)
- [Schukat-Talamazzini 1995] SCHUKAT-TALAMAZZINI, E.G.: *Automatische Spracherkennung*. Vieweg Verlag, 1995 (ISBN 3-528-05492-1)
- [Zell 1997] ZELL, A.: *Simulation Neuronaler Netze*. 3. Auflage. Oldenburg, 1997 (ISBN-13 978-3486243505)