Inhaltsverzeichnis

[Abstract 3](#_Toc29543607)

[Danksagung 3](#_Toc29543608)

[1. Einleitung 3](#_Toc29543609)

[2. Arten von Pattern 3](#_Toc29543610)

[3. Die Fourier-Analyse 3](#_Toc29543611)

[3.1. Die diskrete Fouriertransformation 3](#_Toc29543612)

[4. Die Spektralanalyse 6](#_Toc29543613)

[5. Convolutional Neural Network (CNN) 7](#_Toc29543614)

[5.1. Funktionsweise der einzelnen Bestandteile 8](#_Toc29543615)

[5.1.1. Convolutional Schicht 8](#_Toc29543616)

[5.1.2. Pooling Schicht 9](#_Toc29543617)

[5.1.3. Vollständig Vermaschtes Netzwerk 9](#_Toc29543618)

[5.2. Aktivierungsfunktionen 10](#_Toc29543619)

[5.2.1. Backpropagation 10](#_Toc29543620)

[5.2.2. ReLU Funktion 10](#_Toc29543621)

[5.3. Segmentation Problem 10](#_Toc29543622)

[5. Pattern Erkennung mithilfe der Fourier Transformation 10](#_Toc29543623)

[6. Pattern Erkennung mithilfe von Rhythmus und Melodie 10](#_Toc29543624)

[6.1. String basierte Pattern suche 10](#_Toc29543625)

[6.2. Geometrische Pattern suche 10](#_Toc29543626)

[6.2.1. Fünf-Dimensionales Punkt Set 10](#_Toc29543627)

[6.2.1.1. SIA 11](#_Toc29543628)

[6.2.1.2. SIATEC 11](#_Toc29543629)

[6.2.1.3. COSIATEC 11](#_Toc29543630)

[6.3. Pattern Erkennung mithilfe von Matrizen 11](#_Toc29543631)

[7. Pattern Erkennung mithilfe von Neuronalen Netzen 11](#_Toc29543632)

[8. Realisierung der Pattern Erkennung 12](#_Toc29543633)

[8.1. Realisierung des Low-/Band-/High-Pass-Filter 12](#_Toc29543634)

[Abbildungsverzeichnis 12](#_Toc29543635)

[Tabellenverzeichnis 12](#_Toc29543636)

[Literaturverzeichnis 12](#_Toc29543637)

[Eidesstattliche Erklärung 14](#_Toc29543638)

# Abstract

# Danksagung

# 1. Einleitung

# 2. Arten von Pattern

# 3. Die Fourier-Analyse

Für jedes wissenschaftliche Feld, welches mit zum Beispiel mechanischen Schwingungen, elektrischen Schwingungen oder auch mit der Bildverarbeitung in Berührung kommt, ist die Fourier-Analyse ein wichtiges Werkzeug. Diese beruht auf der Grundaussage von Jean Baptiste Joseph Fouriers Forschung aus dem Jahre 1822 welche besagt, dass jede Schwingung mithilfe von unendlich vielen Sinus- und Kosinus-Schwingungen zusammengesetzt werden kann. (Strick, 2012)

Somit kann auch jede periodische Schwingung wieder in ihre Einzelteile zerlegt werden. Dieser Vorgang wird Fourier-Analyse genannt. Dabei gibt es, je nach Eigenschaft der Funktion, vier verschiedene Arten der Fouriertransformation.

* Fourierreihe
* Kontinuierliche Fouriertransformation
* Diskrete Fouriertransformation (DFT)
* Fouriertransformation für zeitdisktrete Signale (DTFT)

In dieser Arbeit werden lediglich die DFT oder die DTFT betrachtet, da diese die einzigen Transformationen sind, die von einem Computer ausgeführt werden können, weil die DFT/DTFT diskrete Werte und eine endliche Länge besitzen (Smith, 1997).

# 3.1. Die diskrete Fouriertransformation

Wie im vorherigen Kapitel schon beschrieben, ist die diskrete Fouriertransformation die einzige Transformation, welche vom Computer berechnet werden kann. Um aber die diskrete und endliche Funktion der DFT zu verstehen, wird an dieser Stelle aufgezeigt wie sich die Formeln der DFT und der kontinuierlichen Fouriertransformation (CTFT) unterscheiden.

Formel : Formel der CTFT (Weisstein, Discrete Fourier Transform, 2015)

Formel : Formel der DFT (Weisstein, Discrete Fourier Transform, 2015)

In Formel 2 ist N die Anzahl der Samples und *k* das *k*-te Sample Bin.

Der Unterschied der beiden Formel liegt bei dem Tausch des Integrals durch die Summe und bei dem Exponenten der e-Funktion. Das Integral in der DFT entfällt und wurde durch ein Sigma ersetzt, da die DFT mit konkreten Zahlenwerten rechnet und nicht mit Flächen. Die Exponenten unterscheiden sich darin, dass die Frequenz *F* aus der CTFT mit substituiert wurde. Aus der DFT ist *n* gleichzusetzten mit dem aus der CTFT stammende *t* (Weisstein, 2015). Die DFT-Formel kann noch vereinfacht werden indem mit und die *e*-Funktion durch die Eulersche Identitätsubstituiert wird. Dadurch erhält man folgende Formel. (Thormählen, 2018)

Formel : Umformulierte DFT-Formel

Somit müssen für die *x* nur noch die einzelnen Samplewerte eingetragen und die Formel ausgerechnet werden. Das Ergebnis aus dieser Formel besteht aus einem Real- und einen Imaginär Anteil. Die Real- und Imaginären Anteile können dann in ein Koordinatensystem eingetragen werden, wobei der Realteil die x-Achse und der imaginäre Anteil die y-Achse darstellt. Der Winkel des eingetragenen Punkt zur positiven x-Achse beschreibt die Phasenverschiebung der Schwingung und die Entfernung des eingetragenen Punktes zum Koordinatenursprung beschreibt die Amplitude der -ten Schwingung.

Formel : Berechnung des Betrages eines Vektors

Welche Frequenz die -te Schwingung hat hängt von der Abtastfrequenz und der Abtastpunkte ab. Die Frequenz des Punktes berechnet sich durch . (Hermann, 2010)

Bevor Formel 4 benutzt werden kann, um die Amplitude zu berechnen, müssen alle Samplewerte deren Index ≥ Anzahl Samples/2 sind gelöscht werden. Dieser Wert wird das Nyquist-Limit genannt. Es besagt das alle Ergebnisse über diesem Limit gelöscht werden und die Ergebnisse der Samples, die unter dem Nyquist-Limit liegen, verdoppelt werden (Weisstein, 2005). Zuletzt müssen die jetzigen Resultate durch die Anzahl der Samples *N* geteilt werden. (Hermann, 2010)

Bei der DFT können jedoch zwei Fehler geschehen. Zum einen *Leakage* und zum anderen *Aliasing*. Leakage tritt auf, wenn die Schwingung in dem betrachteten Zeitabschnitt nicht perfekt periodisch ist.

Ein Bild, das Text, Karte enthält.

Automatisch generierte Beschreibung

Abbildung : Aufzeigen des Leakage-Effektes mit einer Funktion von 2 Hz (oben) und 2.5 Hz (unten)

Wie in der vorgestellten Abbildung zu sehen, ist in dem Zeitfenster eine Sprungstelle zu sehen. Die Sprungstelle befindet sich an den Rändern des Ursprungsignals. Durch den daraus resultierenden ungleichmäßigen Kurvenverlauf kommt es zu einer „*Verschmierung*“ des Spektrums. Der *Leakage*-Effekt kann mit Fenstfunktionen abgeschwächt werden. Wie in **Abbildung 2** zu sehen ist, schwächen diese Funktionen die Ränder des Ursprungsignals so ab, dass diese gegen Null gehen. Dadurch werden bei einem wiederholen des Signals nur noch möglichst kleine Sprünge im Kurvenverlauf vorkommen (Roberts, 2017). Einige Fensterfunktionen sind unter anderem Barlett-, Gauß- oder die Hanning-Festerfunktion (Weisstein, 2015).

Ein Bild, das Himmel enthält.

Automatisch generierte Beschreibung

Abbildung : Anwenden der Hanning-Fensterfunktion auf eine Schwingung von 2.5 Hz

*Aliasing* tritt auf, wenn die Samplerate zu gering für hohe Frequenzen ist. Um Aliasing zu verhindern muss die Samplerate angehoben werden oder das Signal muss vorgefiltert werden, um die zu hohen Frequenzteile zu minimieren (Roberts, 2017).

# 4. Die Spektralanalyse

Die Spektralanalyse eines Musikstücks besteht aus vielen einzelnen Fouriertransformationen, die über den gesamten Verlauf des Stücks durchgeführt werden. Dabei hängt die Länge der einzelnen Fouriertransformationen und die Auflösung der Frequenzen von der Größe des betrachteten Fensters ab. Bei einer Samplerate von 44100 Samples pro Sekunde, was eine Standartgröße bei Musik auf CDs angesehen werden kann (Teufel, 2019), bedeutet dass das der Abstand zwischen 2 Samples Rund 22,67 µs beträgt. Daraus lässt sich wiederum Schlussfolgern das bei einer Fenstergröße von 32 Samples die zeitliche Genauigkeit, durch die Fenstergröße von 0,7 ms, im Mittelpunkt steht aber die Frequenz vernachlässigt wird. Die Vernachlässigung beruht auf dem in Punkt 3.1 beschrieben Nyquist-Limit, wodurch der Frequenzbereich in 15 gleichgroße Bereiche geteilt wird. Bei 44100 Samples pro Sekunde und der daraus resultierenden maximalen Frequenz von 22050 Hz Besitz jeder Bereich 1470 Frequenzen, was für eine Erkennung von Pattern unbrauchbar ist wie **Abbildung 3** aufzeigt.

Ist jedoch das Fenster groß, gibt es eine feine Unterteilung der Frequenz aber eine ungenaue zeitliche Genauigkeit. Analog zu dem obigen Rechenbeispiel aber mit einer Fenstergröße von 32768 Bins bedeutet dies, dass das Fenster circa 0,74 Sekunden groß ist und jeder Frequenzbereich ein Delta von, rein Rechnerisch, 1,345 Hz hat. Um die großen Zeitintervalle auszugleichen können Bereiche auch überlappt werden. Dadurch wird das Spektralbild, je nach Überlappungsgrad, genauer. (Cannam, Landone , & Sandler, 2010)



Abbildung : a) Spektrum mit 32 Bins berechnet  
 b) Spektrum mit 32768 Bins berechnet (herangezoomt)

# 5. Convolutional Neural Network (CNN)

Um die Spektren, welche vergleichbar mit **Abbildung 3.b** sind, automatisch verarbeiten zu können werden neuronale Netze benötigt. Dafür wird ein CNN benutzt, da diese besser geeignet sind als andere Arten von neuronalen Netzwerken. Der wichtigste Vorteil ist die Genauigkeit, mit der das Netzwerk die geforderte Klassifizierung vornimmt. In dem Bereich der Pattern Erkennung sind CNN’s seit circa 9 Jahren besser als andere Typen von neuronalen Netzen. (Imagenet, 2017)

Der zweite Vorteil ist die Effizienz. Durch die Pooling-Schichten der CNN’s wird der Rechenaufwand verringert. Die bekanntesten Pooling Verfahren sind das *max-Pooling* und das *average-Pooling*. Hierbei wird nur ein Wert aus einer vordefinierten *mxn Poolingmatrix* übernomme. (Karpathy, 2018)

Die in diesem Kapitel vorgestellten Netze haben ebenfalls einen anderen Aufbau im Gegensatz zu anderen neuronalen Netzen (NN). NN besitzen mindestens 2 Schichten. Eine eindimensionale Eingabe-Schicht und eine eindimensionale Ausgabe-Schicht. Optional können versteckte Schichten eingefügt werden und diese vollständig oder nur teilweise mit der Nachfolgerschicht verbunden werden. (Beck & Rey, 2018) Durch die Eindimensionalität der Input-Schicht können Daten nur als Vektor an das NN weitergegeben werden, wodurch die Verarbeitung von Bildern nur noch eine eingeschränkte Effizienz aufweist. CNN’s haben den eben beschriebenen Aufbau und Nachteil nicht. Sie besitzen eine bis zu dreidimensionale Input-Schicht, womit RGB-Farbbilder Effizient an das Netz gegeben werden können. (Karpathy, 2018) Die zweite Komponente der CNN’s ist die *Convolutional* Schicht, welche Matrizen über die Pixelmatrix der Schicht schiebt, um gewünschte Eigenschaften stärker hervor zu heben. Komponente drei ist die *Pooling* Schicht. Das Pooling bewirkt, dass gewisse Informationen, innerhalb einer mxn Matrix, verworfen werden. Dadurch wird die Anzahl der Pixel des Bildes verkleinert. Der letzte Baustein ist ein voll vermaschtes neuronales Netz. Das vollvermaschte Netz wird benötigt um die verarbeiteten und verkleinerten Bilder zu Kategorisieren. (Math Works)

Die eben beschriebenen Bausteine müssen nicht in der vorgestellten Reihenfolge vorkommen. Es können mehrere Convolutional Schichten realisiert werden bevor eine oder mehrere Pooling Schichten implementiert werden. Diese Strukturen können sich dabei in ein und demselben Netz auch wiederholen. Die einzigen Strukturen welche einen festen Platz in der Topologie des neuronalen Netzes besitzen ist die Input-Schicht, welche am Anfang implementiert sein muss, und das vollvermaschte neuronale Netz mit der dazugehörigen Output-Schicht, welches das letzte Segment des CNN’s darstellt.

# 5.1. Funktionsweise der einzelnen Bestandteile

# 5.1.1. Convolutional Schicht

Die Convolutional Schicht ist der Hauptteil eines CNN. Bei der in dieser Schicht stadtfindende Faltung (engl. convolution), werden vom Ersteller gewünschten Merkmale extrahiert. **ToDo: In höheren Schichten Einzelne Merkmale in Tieferen Schichten was anderes + Bild mit möglichen Merkmalen.** Die Merkmale werden mit Hilfe von Filterkerneln extrahiert. Der Filterkernel wird auch Filtermatrix genannt, da der Filter in einem Matrixformat vorliegt. Bei zweidimensionalen Datensätzen kann der Filter eine Größe von 1x1 bis hin zur Größe der vorliegenden Matrix besitzen. Bei dreidimensionalen Datensätzen müssen Filter in der dritten Dimension nicht > 1 sein. (Brownlee, How Do Convolutional Layers Work in Deep Learning Neural Networks?, 2019) Die Filtermatrix wird dabei über die Pixelmatrix der vorherigen Schicht geschoben. In welcher Schrittweite das Kernel über die Bildmatrix geschoben wird, kann vom Ersteller frei gewählt werden. Eine größere Schrittweite hat einen kleineren Rechenaufwand, ein kleineres Bild nach Ausführung der Faltung und einen eventuellen Informationsverlust zur Folge. Bei dem schieben des Kernels wird eine pixelweise Multiplikation zwischen dem Pixel des Filters und dem dazugehörigen, vom Filter eingeschlossenen, Pixel der Bildermatrix durchgeführt. Die Ergebnisse der Multiplikation zwischen Filter und Pixelwert werden anschließend addiert. Der ausgerechnete Wert wird in einer *Featuremap* gespeichert, welche die Ausgabe der errechneten Werte nach der Durchführung eines Filters darstellt. Pro Convolution Schicht, können mehrere Filter realisiert werden. Bei der Realisierung von mehreren Filtern der Größe 2x2, wird die dritte Dimension der Folgeschicht, in Abhängigkeit zu der Größe und Anzahl der verwendeten Filterkernel, größer. (Karpathy, 2018)

Da größere Filter jedoch nicht in die Pixelmatrix passen, wenn diese an Pixelposition 1x1 oder an jeder beliebigen Stelle am Rand stehen, existieren zwei Möglichkeiten diese Sonderfälle zu behandeln. Die erste Option ist die Bildermatrix mit einem *zero-padding* so zu vergrößern, dass der Filter an Position 1x1 im Ausgangsbild anfangen kann, was bewirkt das die Matrix nach der Faltung dieselbe Größe wie davor aufweist. Die zweite Möglichkeit ist es nur valide Startpunkte für den Filter zu wählen. Bei einer 3x3 Filtermatrix würde somit die erste Berechnung an Bildpunkt 2x2 anfangen, bei einem 5x5 Bild an Position 3x3 usw. Wenn die zweite Option gewählt wird, verkleinert sich die Featuremap in Abhängigkeit der Größe des Filters. (Karpathy, 2018)

# 5.1.2. Pooling Schicht

Die Pooling Schicht dient dazu, die eben errechneten Featuremaps zu verkleinern. Durch eine Verkleinerung der Featuremap sinkt die Anzahl der zu lernenden Parameter und der Rechenaufwand. Dadurch wird auch ein *Overfitting* des neuronalen Netzes vorgebeugt. (Karpathy, 2018)

Overfitting eines neuronalen Netzes bedeutet, dass das Netz gute Ergebnisse auf den Trainingsdaten erbringt, aber die Ergebnisse von neuen Klassifizierungen eine nicht ausreichende Genauigkeit aufweisen. (Brownlee, 2019)

In der Pooling Schicht, wird abermals die Größe der Matrix angegeben. Jedoch werden keine Zahlenwerte, wie bei der Faltung in Kapitel 5.1.1, für die einzelnen Positionen in der Matrix eingegeben. Es wird lediglich angegeben mit welcher Schrittweite der Poolingkernel über die errechneten Ergebnisse der Convolution geschoben wird und mit welchem Verfahren das Kernel arbeitet. Es existieren drei verschiedene Pooling Verfahren.

* Average Pooling
* Max Pooling
* L2 Pooling

Average Pooling berechnet den Durchschnitt aller in der Matrixregion enthaltenen Zahlenwerten. Max Pooling nimmt sich nur den größten Zahlenwert und verwirft den Rest und L2 Pooling berechnet die Summe der Kernelregion und zieht zusätzlich daraus die Quadratwurzel. (Nielsen, 2015)



Abbildung : Anwendung der Unterschiedlichen Pooling Strategien mit einem 2x2 Filterkernel

In der obigen Abbildung soll die Ausgangsmatrix ein Bild mit schwarzen Rand und immer heller werdenden Inhalt darstellen. In den Ergebnismatrizen ist zu sehen, dass jede Poolingstrategie unterschiedliche Informationen enthält, woraus am Ende auch unterschiedliche Erkennungsraten resultieren.

Des Weiteren kann über das freie Festlegen der Schrittweite ein überlappendes Pooling erzeugt werden. Normalerweise wird die Schrittgröße der Matrixgröße angepasst, was bei einer 3x3 Matrix die Schrittweite drei bedeuten würde usw. Nimmt man aber die Schrittweite zwei, bei dem oben beschriebenen Beispiel, kommt es zu einer Überlappung der Pooling Matrizen. Die Vorteile des Überlappenden Pooling, im Gegensatz zu dem regulären Pooling, sind ein kleinerer Informationsverlust nach dem Pooling, bessere Erkennungsraten bei gleichbleibenden Algorithmen und Struktur und der Overfitting-Effekt ist schwieriger zu erreichen. Die Nachteile des Überlappenden Poolings sind, dass die Größe der Feature Maps nicht so klein werden wie beim regulären Pooling. (Krizhevsky, Sutskever, & Geoffrey, 2012)

# 5.1.3. Vollständig Vermaschtes Netzwerk

Das vollständig vermaschte Netzwerk bildet das letzte Glied in einem CNN. In diesem Teil des neuronalen Netzes wird die Klassifikation, der in den vorangestellten Schichten herausgefilterten Merkmale, durchgeführt. Hierbei sind alle Neuronen einer Schicht mit allen Neuronen der folgenden Schicht Verbunden. Neuronen derselben Schicht sind jedoch unabhängig voneinander. Die Output Schicht des vollständig vermaschten Netzwerkes, stellt auch die Output Schicht des CNN dar. Jedes Output Neuron ist mit einer Klassifikationsgruppe gleich zu stellen, wobei die ausgegebenen Werte für Wahrscheinlichkeiten für die Klassifizierung in diese Gruppe darstellen. (Karpathy, 2018) (Brownlee, 2019)

# 5.2. Die ReLU-Aktivierungsfunktionen

ReLU steht für Rectefied Linear Unit und steht streng genommen nicht für die Aktivierungsfunktion für sich, sondern für den Teil des CNN’s, welches die lineare Korregierung (engl. Rectefied Linear) durchführt. Da Fachliteratur jedoch meist von der ReLU-Aktivierungsfunktion geschrieben wird, wird hierbei diese Ungenauigkeit bewusst eingegangen, um mit den Quellen im Einklang zu bleiben.

Aktivierungsfunktionen bilden den Netzinput auf ein Aktivitätslevel ab. Diese Abbildung geschieht häufig in einem 2-dimensionalen Diagramm in welchem die x-Achse der Netzinput und die y-Achse das daraus folgende Aktivitätslevel darstellt. (Beck & Rey, 2018)

Die am häufigsten benutzten Funktionen, neben der ReLU-Aktivierungsfunktion, sind die Sigmoid- und die Tanh-Aktivierungsfunktion. Der große Unterschied ist die beschreibende Funktion und der daraus resultierenden unterschiedlichen Aktivierungslevels des Netzes.

(Bild einfügen für Funktionen von LeRU,Sigmoid, Tanh)

ReLU bringt allerdings mehrere Vorteile im Bereich des Deep Learnings, gegenüber Sigmoid und Tanh, mit sich. Der erste Vorteil ist, dass die ReLU-Funktion, keinen *vanishing gradiant effect* besitzt. Vanishing gradiant Effekte treten durch den festen Aktivierungsintervall von (0,1) der Sigmoid oder Tanh auf. Sigmoid und Tanh projizieren große Inputs auf 1 und kleine auf 0, selbst wenn diese Inputs wichtige Informationen enthalten würden. Außerdem sind die Funktionen in der Nähe der Extremwerte zu unsensibel. Deshalb sind die möglichen Veränderungen, im Gegensatz zur ReLU, nur marginal möglich. Backpropagation rechnet diese marginalen Fehler auch noch durch jede vorhandene Schicht zurück, wobei der Gradient des Fehlers je Schicht kleiner wird. Dadurch haben die Neuronen in der Nähe der Output Schicht, welche noch vernünftig große Gradienten aufweisen, einen guten Lerneffekt. Neuronen, die weiter im inneren oder in der Nähe der Input Schicht liegen haben nur noch einen sehr kleinen Gradienten. Dadurch lernen diese Schichten nur sehr schlecht bis gar nicht. (Brownlee, 2019)

Hier dann weiter mit der Quelle drüber und dem Paper wieso ReLU diesen fehler nicht machen.

# 5.3. Backpropagation

# 5.4. Segmentation Problem

Hier dann aus Link: <https://stackoverflow.com/questions/42364513/how-to-locate-multiple-objects-in-the-same-image>

# 6. Pattern Erkennung mithilfe von Rhythmus und Melodie

# 6.1. String basierte Pattern suche

# 6.2. Geometrische Pattern suche

# 6.2.1. Fünf-Dimensionales Punkt Set

(Erst ab drittens Interessant!! Zweidimensional Aufbereiten <http://drops.dagstuhl.de/opus/volltexte/2006/652/pdf/06171.MeredithDavid.Paper.652.pdf> Plus Quellen, Chromatische Scala für Musik(Wert 2 der Fünf Dimensionales Punktset)

Algorithmen dafür unter anderen: SIA (MTPs), SIATEC, COSIATEC)

# 6.2.1.1. SIA

# 6.2.1.2. SIATEC

# 6.2.1.3. COSIATEC

# 6.3. Pattern Erkennung mithilfe von Matrizen

<http://hss.ulb.uni-bonn.de/2013/3271/3271.pdf> (darunter alle von dem Link ab 6.2 interessant)

Pattern können auch in Subpattern unterteilt sein

Pattern ranking function?

Ganze Musikstück muss nicht unbedingt ein Pattern sein es können auch „sinnlose“ Filler dabei sein

Arten von Pattern (Seite 98)

Methode: Melody extraction (nimmt an das Melodie in der höchsten Note enthalten ist)

Stringbasierte Erkennung und geometrische Erkennung (Seite 101 Quellen anschauen) und beides zusammen (102 oben)

<https://www.researchgate.net/profile/Jia_Lien_Hsu/publication/221615538_Efficient_Repeating_Pattern_Finding_in_Music_Databases/links/54e614280cf2bff5a4f29302.pdf>

correlative Matrix Verfahren

<http://pdfs.semanticscholar.org/f6ce/8e49f1987d927be91c99e82458c415266c89.pdf>

Baumstruktur (suffix tree) beim auswerten (auch schon in anderen Werken benutzt)

# 7. Pattern Erkennung mithilfe von Neuronalen Netzen

Mit Bildoperation aus Computergrafik bearbeiten <- vllt nicht nötig da Einstellungen im Sonic Visualizer die obsolet machen

Motif Viewer mal anschauen

# 8. Realisierung der Pattern Erkennung

Nach dem in den vorherigen Kapiteln Grundlagen der einzelnen Grundbausteine der Pattern Erkennung erklärt wurden, geht es in diesem Abschnitt um die Realisierung der einzelnen Zwischenaufgaben. Die Umsetzung des Projektes wird in Python geschehen.

Zu aller erst müssen auf das gewünschte Musikstück verschiedene Filter gelegt werden. Die benötigten Filter sind ein Low-Pass-Filter, ein Band-Pass-Filter und ein High-Pass-Filter. Als nächstes werden die Patternstreams der Beats per minute (BPM) und des Rhythmus des Musikstücks extrahiert. Danach werden die Spektrumbilder des Low- und High-Pass-Filter und die Patternstreams der BPM und des Rhythmus an ein neuronales Netz gegeben, um die Pattern des Basses, der Drum und der Clap zu erkennen.

Der darauffolgende Schritt besteht darin, dass die Audiodatei des Band-Pass-Filters so manipuliert wird, sodass die Konvertierung der .wav Datei in ein MIDI-File möglich ist. Wenn dies geschehen ist werden die im Kapitel [am Schluss aktualisieren] verschiedene Algorithmen zur Pattern Erkennung implementiert und bewertet.

Zum Schluss müssen die erkannten Pattern angezeigt werden.

# 8.1. Realisierung des Low-/Band-/High-Pass-Filter

Für die Umsetzung der drei Filter, wird das aus der Pythonbibliothek stammende Packet aubio verwendet. Dieses ist zum einlesen der .wav zuständig. Da es in diesem Packet keine Funktionen für einen Low-/Band-/High-Pass-Filter gibt, muss für dessen Umsetzung ein Biquad-Filter benutzt werden.

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Literaturverzeichnis

Beck, F., & Rey, G. (2018). *Neuronale Netze: Eine Einführung in die Grundlagen, Anwendungen und Datenauswertung.* Deutschland: hogrefe. Abgerufen am 8. Januar 2020 von Neuronale Netze: http://www.neuronalesnetz.de

Brownlee, D. J. (26. September 2019). *How Do Convolutional Layers Work in Deep Learning Neural Networks?* Abgerufen am 16. Januar 2020 von Machine Learning Mastery: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

Brownlee, D. J. (6. August 2019). *How to Avoid Overfitting in Deep Learning Neural Networks*. Abgerufen am 17. Januar 2020 von Machine Learning Mastery: https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/

Cannam, C., Landone , C., & Sandler, M. (2010). *A Brief Reference*. Abgerufen am 2. Januar 2020 von Sonic Visualiser: https://www.sonicvisualiser.org/doc/reference/1.3/en/

Hermann, P. D. (2010). *Die Disktrete Fouriertransformation (DFT)*. Abgerufen am 19. Dezember 2019 von Technische Universität Wien: https://ti.tuwien.ac.at/cps/teaching/courses/dspv/files/DFT-FFT.pdf

Imagenet. (26. July 2017). *Large Scale Visual Recognition Challenge 2017 (ILSVRC2017)*. Abgerufen am 7. Januar 2020 von Imagenet: http://image-net.org/challenges/LSVRC/2017/results

Karpathy, D. A. (17. Oktober 2018). *CS231n Convolutional Neural Networks for Visual Recognition*. Abgerufen am 16. Januar 2020 von Github: http://cs231n.github.io/convolutional-networks/#add

Krizhevsky, A., Sutskever, I., & Geoffrey, H. (2012). *ImageNet Classification with Deep Convolutional.*

Math Works. (kein Datum). *Convolutional Neural Network: Drei Dinge, die Sie wissen sollten*. Abgerufen am 8. Januar 2020 von Math Works: https://de.mathworks.com/solutions/deep-learning/convolutional-neural-network.html#howitworks

Nielsen, M. A. (2015). *Neural Networks and Deep Learning.* Determination Press. Abgerufen am 17. Januar 2020 von http://neuralnetworksanddeeplearning.com

Roberts, P. S. (2017). *University of Oxford.* Abgerufen am 31. Dezember 2019 von Lecture 7 - The Discrete Fourier: http://www.robots.ox.ac.uk/~sjrob/Teaching/SP/l7.pdf

Smith, S. W. (1997). *The Scientist & Engineer's Guide to Digital Signal Processing.* USA, Kalifornien: California Technical Pub.

Strick, H. K. (1. Juli 2012). *Joseph Fourier (1768–1830)*. Abgerufen am 18. Dezember 2019 von Spektrum: https://www.spektrum.de/wissen/joseph-fourier-1768-1830/1156113

Teufel. (9. Juli 2019). *Die Abtastrate – Tastend nach dem besten Sound*. Abgerufen am 2. Januar 2020 von Teufel: https://blog.teufel.de/abtastrate/#chapter2

Thormählen, P. D. (23. April 2018). *Multimediale Signalverarbeitung Frequenztransformation*. Abgerufen am 19. Dezember 2019 von Philipps Universität Marburg: https://www.mathematik.uni-marburg.de/~thormae/lectures/mmk/mmk\_3\_2\_ger\_web.html#1

Weisstein, E. W. (15. April 2005). *Nyquist Frequency*. Abgerufen am 19. Dezember 2019 von Wolfram Math World: view-source:http://mathworld.wolfram.com/NyquistFrequency.html

Weisstein, E. W. (2. Februar 2015). *Apodization Function*. Abgerufen am 31. Dezember 2019 von Wolfram Math World: http://mathworld.wolfram.com/ApodizationFunction.html

Weisstein, E. W. (2. Februar 2015). *Discrete Fourier Transform*. Abgerufen am 19. Dezember 2019 von Wolfram Math World: http://mathworld.wolfram.com/DiscreteFourierTransform.html

# Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich diese Masterarbeit selbstständig ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Alle den benutzten Quellen wörtlich oder sinngemäß entnommenen Stellen sind als solche einzeln kenntlich gemacht.

Diese Arbeit ist bislang keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht worden.

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

Leipzig, den