Hochschule Düsseldorf
University of Applied Sciences
HSD

Fachbereich Medien
Faculty of Media

WISSENSCHAFTLICHE VERTIEFUNG

Der Fortschritt automatischer Musiktranskriptionssysteme durch Künstliche Intelligenz

Autor:
Benedikt Kolodziej
878007
Medieninformatik (B. Sc.)

Betreuender Professor:
Prof. Dr. Dennis Müller
dennis.mueller@hs-duesseldorf.de

August 26, 2025

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Automatische Musiktranksription

Musik war schon immer ein zentraler Bestandteil unserer Gesellschaft. Während früher musikalische Werke meist mündlich überliefert wurden, entwickelte sich während des Mittelalters die Notenschrift, welche wir auch heutzutage noch nutzen. Diese Notenschrift ermöglichte es, Musikstücke einfacher zu erlernen und Musik einem breiteren Publikum zugänglich zu machen. Durch die Digitalisierung erhielten „Digital Audio Workstations“ zunehmend Einzug in die Musikproduktion, wodurch Notenblätter irrelevanter wurden und somit weniger Musikstücke in Notenschrift übersetzt wurden.

Ungefähr gleichzeitig begann zudem die Forschung an AMT-Systemen (Automatic Music Transcription). AMT ist ein Prozess, zur automatischen Transkription von Audiospuren. Als Input wird ein Audiosignal gegeben, welches durch verschiedene Prozesse umgewandelt wird. Diese Prozesse analysieren die Eigenschaften der Noten, zum Beispiel die Frequenz, Lautstärke und vieles mehr. Dadurch entsteht als Output eine MIDI-Datei, ein Notenblatt oder andere Daten, die das gegebene Musikstück in Schriftform darstellen.

1.2 Grundlegende Begriffe in AMT-Systemen

Bevor wir tiefer in das Forschungsfeld der automatischen Musiktranskription eintreten, sollen zunächst einige zentrale Begriffe erläutert werden. Diese sind für das grundlegende Verständnis von AMT-Systemen unverzichtbar, da sie in nahezu allen Ansätzen Anwendung finden, mit Ausnahme der Polyphonie, welche in frühen Systemen oftmals noch nicht berücksichtigt wurde. Darüber hinaus stellen sie die wesentlichen Bausteine dar, wie etwa der Beginn und das Ende von Noten, die Behandlung mehrstimmiger Musikstücke und zuletzt der Input und Output.

Onsets und Offsets

Onsets beschreiben den exakten Zeitpunkt, an dem eine Note beginnt. Dementsprechend zeigen Offsets den Zeitpunkt, an dem diese endet. Onsets werden in der Signalverarbeitung typischerweise durch plötzliche Änderungen im Audiosignal erkannt. Durch einen starken Anstieg der Amplitude, Änderungen im Spektrum oder plötzliche Energiezunahmen im Frequenzbereichen. Offsets markieren den Zeitpunkt, an dem die im Onset erfassten Merkmale nicht mehr vorhanden sind und die Note endet. Die präzise Erkennung von Onsets und Offsets ist essenziell, um die Dauer einer Note korrekt zu bestimmen und eine vollständig korrekte Notenschrift zu erzeugen. Fehler in der Erkennung können zu falscher Notenlänge, fehlenden Noten oder zusätzlichen Phantomnoten führen. In späteren Modellen, wie RNNs, können falsche Onsets und Offsets auch zu Folgefehler führen.

Die folgende Darstellung verdeutlicht, wie musikalische Ereignisse anhand von Änderungen in der Amplitude erkannt werden können. Ein deutlicher Anstieg der Amplitude markiert den Onset einer Note.

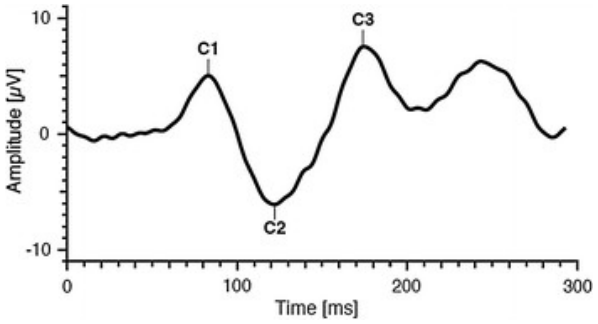

Abbildung 1: 3 Noten Onsets mit Abhängigkeit von Amplitude und Zeit [Brown, S. G. Crewther und D. P. Crewther 2016].

Polyphonie

Polyphonie beschreibt in der Musik die Mehrstimmigkeit. Das bedeutet, dass ein Musikstück

mehrere Stimmen gleichzeitig wiedergibt. In der automatischen Musiktranskription muss das System diese gleichzeitig erklingenden Stimmen erkennen, voneinander unterscheiden und den jeweiligen Stimmen zuordnen. Je mehr Stimmen ein Musikstück besitzt, desto schwieriger ist die Transkription. Ein klassisches Beispiel für polyphone Musik sind Klavierstücke, Orchesterwerke oder Chormusik, bei denen oft viele Noten gleichzeitig erklingen.

Spektrogramme

In der automatischen Musiktranskription wird das Input Audiosignal immer zunächst in ein Spektrogramm umgewandelt. Spektrogramme zeigen den zeitlichen Verlauf des Frequenzspektrums eines Audiosignals. Es gibt verschiedene Arten von Spektrogrammen. Zwei Spektrogrammtypen, die häufig genutzt werden, sind CQT-Spektrogramme und Log-Mel-Spektrogramme. CQT steht für Constant-Q Transform und in diesem Spektrogramm werden die Frequenzachsen logarithmisch aufgelöst. Zudem bleibt der Q-Faktor konstant, dieser beschreibt das Verhältnis von Frequenz zu Brandbreite. Das Log-Mel-Spektrogramm wird zunächst mit einer Short-Time Fourier Transform (STFT) gebildet. STFT ist eine Methode, um ein Signal als ein Frequenzspektrum darzustellen. Anschließend wird die lineare Frequenzachse auf eine Mel-Skala projiziert. Das menschliche Gehör kann zum Beispiel 200–400Hz feiner als 5000–5200Hz wahrnehmen. Die Mel-Skala sorgt dafür, das sich das Spektrogramm dem menschlichen Gehör anpasst. Insgesamt ist ein CQT-Spektrogramm besser für die automatische Musiktranskription, da in diesem die Töne feiner unterschieden werden können. Jedoch werden in neueren KI-Modellen zum Großteil Log-Mel-Spektrogramme genutzt. Diese sind robuster und funktionieren, vor allem in Transformer-basierte KI-Modellen, zuverlässiger.

Im Folgenden sind vier verschiedene Spektrogramme zu sehen, welche alle auf den identischen Ausschnitt aus Bachs WTK I, Nr.5 [Heinemann 1992] basieren. Oben links ist ein normales lineares Spektrogramm zu sehen, welches ausschließlich auf dem STFT basiert und eine lineare Frequenzskala besitzt. Rechts daneben ist ein Log-Spektrogramm, bei dem die Amplitudenwerte des STFT logarithmisch in Dezibel skaliert wurden, wodurch der Dynamikumfang im niedrigen Amplitudenbereich erhöht wird. Das „MelSpec“ steht für Log-Mel-Spektrogramm und unten rechts wird ein CQT-Spektrogramm angezeigt.

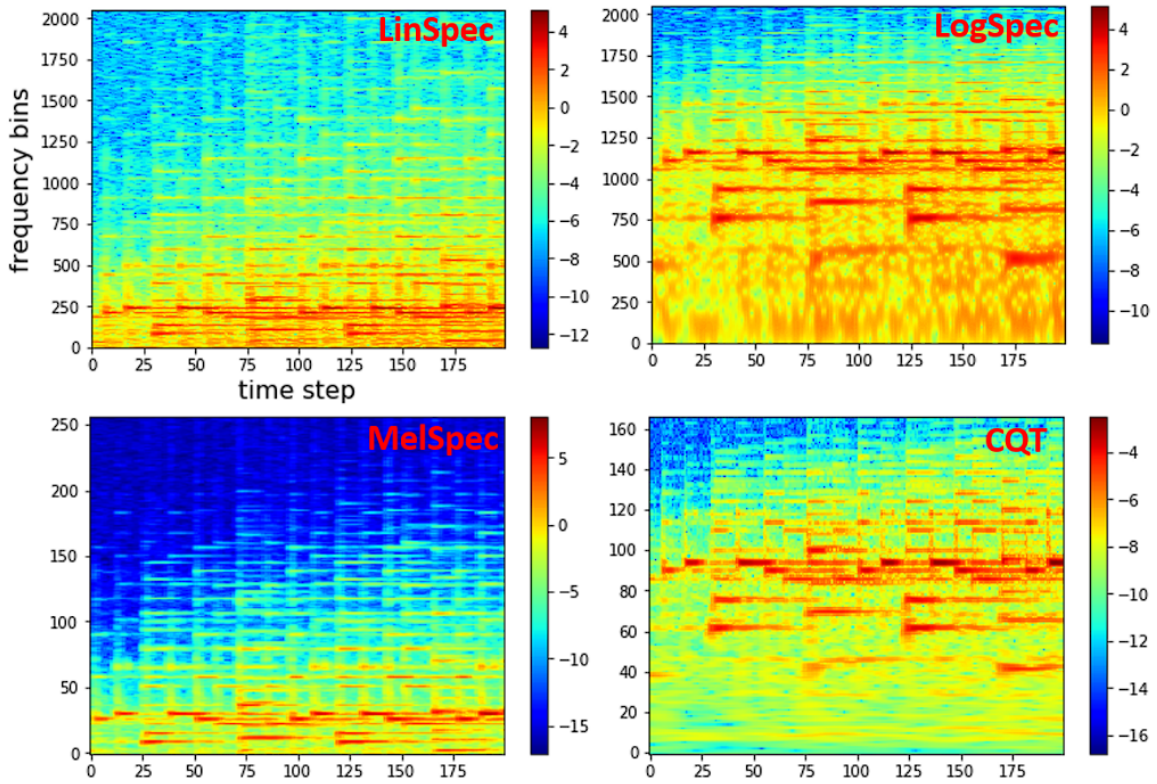

Abbildung 2: Vergleich von vier Spektrogrammen basierend auf [T. Cheuk, Shan und Raphael 2020].

MIDI-Dateien

Notenschrift als Input für ein Computerprogramm, Synthesizer oder ähnliches ist unhandlich. Zunächst müssten die gespielten Noten stets in Notenschrift umgewandelt und anschließend in anderen Programmen analysiert werden. Dieser Prozess ist sehr aufwändig. Eine bessere Lösung wäre eine Datenschreibweise, bei der alle wichtigen Informationen der Noten übersichtlich aufgeschrieben sind. MIDI-Dateien sind dafür perfekt geeignet.

MIDI (Musical Instrument Digital Interface) ist ein Standartprotokoll zur Kommunikation zwischen elektronischen Musikinstrumenten, Computern und anderen Geräten wie zum Beispiel Synthesizern. Dieses Protokoll wurde erstmals 1983 eingeführt und wurde schnell zu einem Standard in der digitalen Musikindustrie [Dave und Chet 1981]. In MIDI-Dateien werden Daten von Tönen gespeichert, welche zum Beispiel zuvor von einem elektronischen Instrument gespielt wurden oder durch AMTs erfasst wurden.

In MIDI-Dateien werden folgende Daten gespeichert:

1. **MIDI Header-Chunk (MThd):** Enthält grundlegende Informationen zur Struktur der Datei:
   - Dateiformat (Formattyp: 0 = eine Spur, 1 = mehrere synchrone Spuren, 2 = unabhängige Spuren)
   - Anzahl der folgenden MTrk-Blöcke
   - Zeitauflösung (Ticks pro Viertelnote)
2. **MIDI Track-Chunks (MTrk):** Jede Spur enthält eine zeitlich sortierte Liste von MIDI-Events:
   - **MIDI-Events:**
     - Onset / Offset
     - Lautstärkeänderung (Control Change)
     - Angabe des spielenden Instruments (Program Change)
     - Tonhöhenänderung (Pitch)
     - Druckstärke pro Taste (Aftertouch / Polyphonic Key Pressure)
   - **Meta-Events):**
     - Tempo in Mikrosekunden pro Viertelnote (Set Tempo)
     - Taktart (Time Signature)
     - Tonart (Key Signature)
     - Spurname (Track Name)
     - Liedtext (Lyrics)
     - Markierungen (Markers)
     - Ende der Spur (End of Track)
   - **Systemexklusive Ereignisse (SysEx):**
     - Herstellerspezifische Daten wie Synthesizer-Voreinstellungen oder Spezialbefehle
3. **Delta-Time:** Gibt die Zeit in Ticks an, die seit dem letzten Ereignis vergangen ist:
   - Ermöglicht die genaue zeitliche Positionierung jedes MIDI-Ereignisses

- Grundlage für das Timing und die rhythmische Struktur der Datei

Am wichtigsten sind dabei die MTrk-Blöcke, in denen die Daten der einzelnen Noten gespeichert werden. Dabei stellt ein Track die Eventliste einer ganzen Stimme dar, wie zum Beispiel die Melodiestimme, eine Violine, die Pedalsteuerung eines Klaviers oder Metadaten. Es fällt auf das diese vier Beispiele alle sehr unterschiedliche Aufgaben und Bedeutungen haben. Das liegt daran, dass in MIDI-Dateien zusammenhängende Funktionen gespeichert werden und nicht nur Musiknoten.

MIDI-Dateien kamen auch der Forschung für AMT-Systeme sehr gelegen, da nun ein standardisiertes Output-Format für diese Programme zur Verfügung stand. Später werden diese zudem sehr essenziell bei dem Training KI basierter AMT-Systeme [Telila, Cucinotta und Bacciu 2025].

Im folgenden Bild wird ein Auszug einer MIDI-Datei, mit zwei Noten, angezeigt. Darunter ist ein Piano-Roll abgebildet, zugehörig zu oben genannten Noten. Ein Piano-Roll ist eine grafische Darstellung von Musiknoten über die Zeit. Dabei gibt die Position die Tonhöhe an und die Länge die Spieldauer.

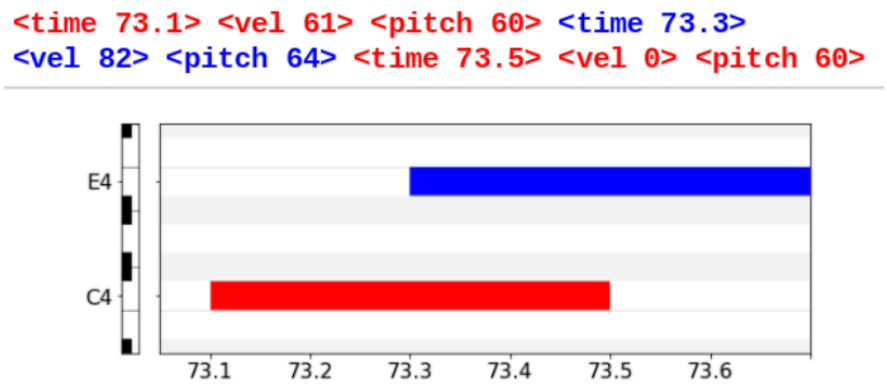

Abbildung 3: MIDI-Datensatz mit zugehörigem Piano-Roll. Ausschnitt aus der Varbeitungspipeline des MT3-Modells [Gardner u. a. 2022].

1.3 Herausforderungen und Hindernisse

Anstatt durch sein Gehör selber diese Musikstücke in Notenschrift zu übertragen, würde ein AMT-System diese Aufgabe übernehmen. Dieses Ziel war jedoch schwer zu erreichen, da Musik mehrdimensional, durch Aspekte wie zeitlicher Abfolge, Tonhöhe und Polyphonie, ist. Vor allem bei polyphonen Musikstücken haben herkömmliche Algorithmen häufig Schwierigkeiten. In diesen Fällen müssen AMT-Systeme mehrere verschiedene Stimmen gleichzeitig analysieren und im späteren die jeweiligen Töne voneinander differenzieren um diese dann eindeutig einem Instrument zuordnen zu können. Ein weiteres Problem ist die Individualität jedes Musikstückes. In realen Aufnahmen können leichtes Rauschen, kleine Spielfehler oder stilistische Mittel wie Vibrato auftreten, die je nach Interpreten unterschiedlich klingen. Zudem sind die meisten AMT-Modelle auf die westlichen Tonleiter trainiert. Dies kann zu Problemen führen, wenn zum Beispiel arabische oder indische Musikstücke transkribiert werden sollen.

In folgender Tabelle sind einige Regionen aufgelistet und deren Tonleiter, die diese benutzen. Die Tabelle ist nicht vollständig, auf der Welt gibt es noch viele weitere Regionen, welche auch ihre ganz eigenen Tonleiter nutzen.

Region	Typische Skalen
Europa	Dur, Moll, Kirchentonarten, Pentatonik
Arabische Welt	Maqām Rast, Maqām Bayati, Maqām Hijaz, Maqām Kurd, Maqām Nahawand
Türkei	Hicaz, Nihavent, Hüseyni, Segah, Uşşak
Persien / Iran	Shur, Mahur, Segah, Homayun, Nava
Indien	Rāga Yaman, Bhairav, Todi, Kafi, Bhupali
China	Gong, Shang, Jue, Zhi, Yu
Japan	Hirajoshi, In-sen, Yo, Kumoi
Indonesien	Sléndro, Pélog
Afrika	Pentatonik, Hexatonik, Balafon-Skalen

Tabelle 1: Übersicht typischer Skalen/Tonleiter verschiedener Weltregionen

1.4 Anwendungsfelder und Vorteile

AMT kann auch häufig bei anderen Problemen helfen oder in zahlreichen Bereichen „Quality-of-Life-Changes“ bringen. Zum einen kann der Musikunterricht spannender und interaktiver gestaltet werden. Es würde eine breitere Auswahl von Musikstücken geben, die den Schülern angeboten werden können, wodurch diese durch individuell angepasste Musikstücke mehr Spaß und Ehrgeiz beim Lernen entwickeln. Zudem können die gespielten Musikstücke der Schüler während des spielens transkribiert werden, wodurch frühzeitig erkannt wird, wo der jeweilige Schüler noch Verbesserungsmöglichkeiten aufweist und Feedback erhalten muss. Grundsätzlich können deutlich mehr Musikstücke transkribiert werden, wodurch sich große Archive aufbauen lassen. Ein größeres Interesse an Musik wird geweckt, da Musikstücke von beliebten Serien, Filmen oder Spielen leichter für deren Musikbegeisterte Zielgruppe zugänglich sind. Alleine dadurch, dass KI-Modelle Musikstücke besser verstehen, können darauf aufbauend weitere Tools für die Musikproduktion entwickelt werden. Auch andere Anwendungsfelder von Künstlicher Intelligenz würden dadurch profitieren. KI-generierte Musik würde verbessert werden, da die KI selber ein besseres Verständnis der Musik entwickelt. Audiobasierte Suchmaschinen könnten gewünschte Musikstücke oder bestimmte Videos präziser finden.

1.5 AMT und Künstliche Intelligenz

Um diese musikalische Vielfalt zu bewältigen, wird heutzutage Künstliche Intelligenz und Machine Learning eingesetzt. Im Gegensatz zu Algorithmen ist KI flexibler und kann sich dadurch besser auf kleine Abweichungen in Musikstücken einstellen. Ein Audiosignal kann entweder im Studio produziert oder im Alltag aufgenommen werden. Alltag aufgenommene Signale werden, in dieser Arbeit, als reale Audiosignale bezeichnet. Reale Audiosignale beinhalten immer eine gewisse Menge an Rauschen. Das kann bei der automatischen Musiktranskription für Schwierigkeiten sorgen, da Algorithmen diese als zusätzliche Noten ansehen oder richtige Noten dadurch nicht erkennen könnten. KIs können sich besser an die fehlerhaften Geräusche von Rauschen anpassen, da neuronale Netzwerke mit genau diesen fehlerbehafteten Audiosignalen trainiert werden können. Somit können AMT-Systeme besser angepasst werden an einen realistischen Alltagsgebrauch.

Auch die Mehrdimensionalität von Musik kann KI deutlich besser bewältigen als Algorithmen. Neuronale Netze besitzen eine mehrdimensionale Struktur, die es ihnen ermöglicht, verschiedene Muster, Stimmen und Eigenschaften zu erlernen [Graves, Fernández und Schmidhuber 2007]. Auf der anderen Seite müssen klassische Algorithmen diese verschiedenen Dimensionen explizit modellieren und sind nicht in der Lage, Muster selbstständig zu erkennen. Sie folgen nur den fest einprogrammierten Anweisungen und sind somit nicht anpassungsfähig.

Es gibt verschiedene KI-Modelle, welche für AMT-Systeme in Frage kommen. In diesem Abschnitt werden die am weitesten verbreiteten KI-Modelle aufgegriffen. Weitere potenzielle KI-Modelle und der heutige „State of the Art“ werden im Abschnitt-(4) ausführlich erläutert. Die meisten AMT-Systeme nutzen Recurrent Neural Networks (RNN) und Convolutional Neural Networks (CNN)

[Böck und Schedl 2012]. Diese Module bilden keine eigenständigen Systeme, sondern lassen sich flexibel innerhalb eines Systems kombinieren. Da es keine objektiv richtigen oder falschen Notenfolgen gibt, ist der Einsatz von Supervised Learning am effizientesten. Dementsprechend ist es notwendig ein zuverlässiges und großes Datenset aus Audiodateien und deren zugehörigen MIDI-Dateien zu besitzen.

CNNs können gut räumliche Strukturen erkennen. Das ist sehr hilfreich bei der Analyse von Spektrogrammen. Da das Input Audiosignal in ein Spektrogramm umgewandelt wird, kann das CNN diese bildliche Darstellung analysieren. Durch die Analyse von dem Spektrogramm können gewisse Frequenzmuster erkannt werden, die darauffolgend einem bestimmten Instrument zugeordnet werden können. Das ist vor allem dabei hilfreich, die verschiedenen Stimmen der jeweiligen Instrumente voneinander zu differenzieren [Han, Kim und K. Lee 2016].

RNNs hingegen spezialisieren sich darauf, mithilfe eines Gedächtnis, zeitliche Abläufe zu verstehen. In Musikstücken werden zahlreiche Noten hintereinander gespielt, welche harmonisch aufeinander aufbauen. Das RNN verarbeitet die jeweiligen Sequenzen und merkt sich die Informationen der schon gespielten Noten, um die darauffolgenden Noten besser einordnen zu können. So lässt sich die harmonische Struktur des Musikstückes analysieren. Auch der Rhythmus und vorhandene Vorzeichen lassen sich dadurch interpretieren [Böck und Schedl 2012].

In den meisten AMT-Systemen werden RNNs und CNNs kombiniert. Zunächst verarbeitet das CNN das Spektrogramm und extrahiert dabei musikalische Merkmale. Darauf folgend verarbeitet das RNN diese Merkmale und bildet konkrete Noten mit deren einzelnen Eigenschaften.

1. **CNN**
   Extrahiert folgende Merkmale aus dem Spektrogramm:
   - Frequenzverteilungen und spektrale Muster
   - Tonhöhenlage und damit verbundene Obertöne
   - Klangfarbe einzelner Instrumente
   - Energieverteilung, unter anderem zur Erkennung von Toneinsätzen
   - Harmonische Strukturen wie Akkordfolgen
2. **RNN**
   Verarbeitet auf Basis dieser Merkmale die zeitliche Abfolge und erkennt dabei folgende Eigenschaften:
   - Reihenfolge und Übergänge musikalischer Ereignisse
   - Beginn und Ende einzelner Töne zur Bestimmung der Notendauer
   - Rhythmische Muster und zeitliche Gruppierungen
   - Musikalische Phrasen mit zusammenhängender Struktur
   - Wiederholungen, Themen oder längere Abhängigkeiten im Verlauf
3. **Output**
   Gibt das transkribierte Musikstück in strukturierter Form aus:
   - Als MIDI-Datei mit exakten Noteninformationen

Nachdem die KI-Modelle die Daten verarbeitet haben, kann die erhaltene MIDI-Datei mit einem externen Programm zu standardisierten Musiknoten transkribiert werden.

1.6 Motivation, Zielsetzung und Ablauf

Künstliche Intelligenz ist momentan eins der aktivsten Forschungsfelder. Durch KIs wurden und werden noch viele grundlegende Probleme und Algorithmen komplett umstrukturiert, verworfen oder

gelöst. Ein perfektes Beispiel dafür ist AlphaDev von DeepMind [Mankowitz u. a. 2023], welches durch Reinforcement Learning ein neues, effizienteres Sortier- und Hashverfahren entwickelt hat. Ein Forschungsgebiet, das unter anderem auch sehr stark durch die Einführung von KI profitiert hat, ist die automatische Musiktranskription. Während früher noch Modelle und Algorithmen genutzt worden, um AMT-Systeme umzusetzen, werden jetzt überwiegen verschiedene KI-Modelle genutzt, um unter anderem Spektrogramme zu analysieren und um den Aufbau des Musikstückes zu identifizieren.

Durch die integration von KI in AMT-Systemen rückt die Vorstellung von AMT-Systemen, welche im Alltag genutzt werden auch immer näher. Sobald das erste AMT-System praxisgeeignet ist, werden sich, wie im Abschnitt-(1.4) angeschnitten, viele neue Einsatzmöglichkeiten ermöglicht. Dadurch werden andere Forschungsgebiete gefördert, momentane Aufgaben vereinfacht und das Bildungswesen verbessert.

Deshalb wird der Schwerpunkt dieser wissenschaftlichen Vertiefung darin liegen, auf die Funktion von KI-Modellen in AMT-Systemen. Diese Analyse wird sich aus drei Teilen zusammenfügen. Zunächst werden die heutigen Schwachstellen moderner AMT-Systeme aufgelistet. Als Nächstes werden die KI-Systeme beschrieben, welche grundlegend in AMT-Systemen genutzt werden und welche Vorteile diese mit sich bringen. Zuletzt werden zwei neu zeitige AMT-Systeme vorgestellt, welche beide eine starke Auswirkung auf dieses Forschungsgebiet haben.

Um das Thema automatische Musiktranskription zu vervollständigen, wird im nächsten Kapitel zunächst auf die Geschichte von AMT-Systemen eingegangen. Außerdem wird erläutert, wie frühere AMT-Systeme funktionierten und aussahen.

2 Geschichtliche Entwicklung von AMT-Systemen

2.1 Das erste AMT-System

Eine der ersten Arbeiten über automatische Musiktranskription wurde im Jahr 1977 geschrieben [Moorer 1977]. In dieser beschreibt Moorer seinen Ansatz, polyphone Musikaudiospuren direkt über Computerprogramme in Notenschrift zu übertragen. Während dieses Prozesses fallen ihm schon große Probleme auf, die auch in späteren Arbeiten eine ausschlaggebende Rolle spielen.

2.1.1 Grundlegende Probleme

Eines dieser Probleme wird von ihm als das „Cocktail-Party-Problem“ bezeichnet. Dieses stellt die Schwierigkeit dar, auf einer Party bestimmten Stimmen zu folgen, während viele verschiedene Stimmen gleichzeitig erklingen. Das gleiche Problem existiert in der Noten transkription. Die meisten Musikstücke haben mehrere Instrumente, welche gleichzeitig spielen. Oft gibt es auch Musikstücke, bei denen es mehrere Stimmen für ein Instrument gibt. Ein Beispiel dafür ist die Violine, die zum Beispiel in Orchesterstücken oft auf mehrere Stimmen (erste, zweite, dritte) aufgeteilt wird. Dies erschwert die Zuordnung bestimmter Noten zu einer gewählten Stimme. Der Großteil aller Menschen scheitert daran große Musikstücke richtig zu transkribieren. Noch schwieriger wird es für Computerprogramme. Anfangs identifizieren diese bestimmte Töne anhand der Frequenz des Tons. Moorer stellte fest, dass sich allein anhand der Frequenz nicht eindeutig bestimmen lässt, welche Töne zu einem bestimmten Zeitpunkt gespielt werden.

Das nächste Problem ist die Frequenzüberladung durch Obertöne. Jeder Ton hat Obertöne. Diese Obertöne sind jeweils das Vielfache des gespielten Grundtons. Wird beispielsweise auf einem Klavier der Ton C3 (130,81 Hz) gespielt, so entstehen die Obertöne C4 (261,62 Hz), G4 (392,42 Hz) und weitere. Auch wenn nur C3 gespielt wird, erkennt das Computerprogramm zusätzlich die jeweiligen Obertöne, was als Frequenzüberlagerung bezeichnet wird. Diese Frequenzüberlagerung sorgt dafür, das zum Beispiel ein Klavier anders klingt als eine Violine. Daraus resultiert die Klangfarbe (Timbre) eines bestimmten Instrumentes [Goswami und Velankar 2013]. Moorer konnte das Problem der Obertöne nicht lösen. Zum einen war es ihm aufgrund des damaligen technischen Stands nicht möglich, die Klangfarbe eines Instruments zu erkennen. Zum anderen waren die Algorithmen und Verfahren

jener Zeit nicht in der Lage, die Grundfrequenz von den Obertönen zu trennen. Deshalb fokussierte er sich ausschließlich auf zweistimmige polyphone Musikstücke.

Ein weiteres Problem war Rauschen in unter realen Aufnahmebedingungen erstellten Audiospuren sowie stilistische Mittel in der Musik, wie etwa Vibrato. In real aufgenommenen Audiospuren gibt es immer ein gewisses Hintergrundrauschen [iZotope n.d.]. Dieses kann von einem Computerprogramm auch als Note erkannt werden oder verhindern, dass bestimmte Noten richtig erkannt werden. Moorer nahm das Musikstück selbst analog auf und digitalisierte es anschließend mit einem 14-Bit-A/D-Wandler. Durch die eigene Aufnahme konnte er die Aufnahmeumgebung, die Mikrofonposition und die genutzte Technik kontrollieren, wodurch sich das Rauschen im Vergleich zu unkontrollierten Aufnahmen verringerte. Zudem wies er die Musiker an auf stilistische Mittel zu verzichten. Stilistische Mittel wie Vibrato verursachen kleine, aber kontinuierliche Veränderungen der Frequenz. Dadurch kann das Computerprogramm nicht korrekt erkennen, das eigentlich eine einzelne Note gespielt wurde. Die Folge davon wäre eine falsche Zuordnung des Onsets und Offsets der Noten. Ohne diese erhielt Moorer klarere und besser verwertbare Daten für die Transkription.

Das letzte Problem, was Moorer angesprochen hat, ist die Nutzung von nicht harmonischen Instrumenten wie Trommeln oder einem Schlagzeug. Diese Instrumente haben keinen eindeutigen Pitch. Sie sind abhängig von dem Rhythms und der Lautstärke. Da Moorers AMT sich jedoch auf das Frequenzmuster der Noten fokussiert, konnten diese Musikinstrumente nicht berücksichtigt werden.

2.1.2 Der Aufbau des ersten AMT-Systems

Moorers automatische Musiktranskriptionssystem war eins der ersten seiner Art. Viele weitere AMT-Systeme leiten sich von diesem ab.

Zunächst wird ein analoges Musiksignal mit einem 14-Bit-A/D-Wandler digitalisiert. Dieses digitale Musiksignal wurde anschließend genutzt, um mithilfe von Bandpassfiltern, bestimmte Frequenzbereiche zu isolieren. Ein Bandpassfiltern ist ein Filter welcher nur bestimmte Frequenzen durchlässt. Dadurch konnte Moorer die gespielte Note und deren Dauer, wie deren Onset und Offset, feststellen.

Nun mussten die bestimmten Noten einer gewählten Stimme zugeordnet werden. Dies wurde durch melodische Gruppierung verwirklicht. Zunächst wurden Inseln gebildet. Inseln sind Noten die sich Zeitlich vollständig überlappen. Wir gehen davon aus das jede Stimme nur eine Note gleichzeitig spielt, wodurch diese Noten nicht der gleichen Stimme zugeordnet werden können. Als Nächstes müssen die anderen Noten auf verschiedene Kombinationen getestet werden. Je geringer die Frequenzunterschiede zwischen aufeinanderfolgenden Noten sind, desto wahrscheinlicher stammen sie aus derselben Stimme. Dies liegt daran, dass melodische Linien in der Regel durch kleine, schrittweise Intervalle fortgeführt werden, während größere Sprünge häufiger auf einen Wechsel der Stimme hindeuten. Zudem werden Gruppierungen von Noten erstellt, welche am wahrscheinlichsten harmonisch nacheinander gespielt worden.

Zum Schluss ließ Moorer die gewonnenen Daten durch ein Programm laufen, um diese dann mithilfe eines Plotters in eine Notenschrift umzuwandeln.

2.2 Hidden Markov Modelle

Hidden Markov Modelle (HMM) sind statische Modelle, welche sich sehr gut zur Analyse von Musikstücken eignen. Sie wurden erstmals in den 1960er Jahren erfunden [Baum u. a. 1970] und sind ein zentraler Bestandteil früher AMT-Systeme. HMMs beschreiben eine Abfolge von „versteckten Zuständen“, welche im Kontext von AMT-Systemen Noten im Audiosignal darstellen. Durch indirekt beobachtbare Daten, wie zum Beispiel die Spektraldaten des Audiosignals, können diese Noten erschlossen werden.

HMMs bestehen aus:

- Zustände (States)

- Übergangswahrscheinlichkeiten (Transition Probabilities)
- Emissionswahrscheinlichkeiten (Emission Probabilities)
- Beobachtungen (Observations)

Nehmen wir das Beispiel eines Klaviers. Ein Klavier hat 88 Tasten, und somit mindestens 88 Zustände. Durch Akkorde können zudem mehr Zustände generiert werden. Die Übergangswahrscheinlichkeit stellt dar, wie wahrscheinlich es ist, von einem Zustand zu einem bestimmten anderen Zustand zu wechseln. Zum Beispiel könnte es wahrscheinlicher sein, dass auf die Note C4 der Ton G3 folgt statt D1, da diese Tonfolge harmonischer und musikalisch plausibler klingt. Dies ist jedoch nur eine Annahme anhand von gesammelten Testdaten, weshalb es nicht als Begründung ausreicht. Deshalb kommt als zweite Instanz die Emissionswahrscheinlichkeit hinzu. Diese gibt an, wie wahrscheinlich ein bestimmter Zustand in der momentanen Beobachtung ist. Die Beobachtung wird dabei aus Eigenschaften wie Frequenzverteilung, Spektrogramm oder anderen Merkmalen zusammengesetzt, die aus verschiedenen Modulen hergeleitet werden können. Aus den gesammelten Daten kann nun mithilfe von zum Beispiel dem Viterbi-Algorithmus [Takeda u. a. 2002] die wahrscheinlichste Abfolge von Zuständen berechnet werden.

In gewisser Weise lassen sich HMMs auf KI-Modelle übertragen. Beide arbeiten mit versteckten Zuständen und berechnen anhand von Testdaten die wahrscheinlichste Kombination von Zuständen. Natürlich sind moderne KI-Modelle weitaus komplexer, doch HMMs spielten im Kontext automatischer Musiktranskriptionssysteme eine zentrale Rolle dabei, wie KI-Methoden in diesem Bereich später eingesetzt wurden. Sie beeinflussten insbesondere den Umgang mit zeitlichen Abfolgen und Unsicherheiten in der Tonerkennung, was bis heute relevante Konzepte in KI-basierten AMT-Systemen sind.

2.3 Datensätze für AMT-Systeme

Während der Forschung an neuen AMT-Systemen wurden immer wieder neue Datensätze genutzt, um AMT-Systeme zu bespielen oder heutzutage auch zu trainieren. Es ist aufwändig nützliche Datensätze zu generieren, da viele verschiedene Musikstücke genutzt werden müssen und diese alle unter den gleichen Voraussetzungen aufgenommen werden müssen. In der Transkription von Musikstücken sind Datensätze sehr relevant, vor allem mit dem weiteren Einsatz von KI-Modellen werden große Datensätze gebraucht. Die ersten verwendeten Datensätze waren selbst erstellt und enthielten lediglich grundlegende Eigenschaften der Noten. In den AMT-Systemen von Moorer [Moorer 1977] oder auch Martin [Martin 1996] wurde solch ein Ansatz verfolgt. Durch die Einführung von MIDI-Dateien wurde ein Standard entwickelt, welcher jetzt im Großteil der AMT-Systeme eingebaut wird. Auf der Grundlage von MIDI-Dateien wurden auch spezialisierte Datensätze erstellte. Zwei der größten und wichtigsten sind MAPS (MIDI Aligned Piano Sounds) und MAESTRO (MIDI and Audio Edited for Synchronous Tracks and Organization). Dabei ist MAPS ein Datensatz aus künstlich generierten Audiospuren, während MAESTRO ein ungefähr 200 Stunden langer Datensatz von realen isolierten Studioaufnahmen ist. Vor allem bei dem Training neuer KI-Modellen sind solche großen Datensätze sehr hilfreich. Ein weiterer wichtiger Datensatz ist ADTOF (Annotated Drum Transcription Onset Features). Dieser besteht ausschließlich aus unharmonischen Instrumenten. In Moorers und anderen AMT-Systemen ist die Erkennung von unharmonischen Instrumenten ein großes Problem. KI-Modelle können sich mithilfe von dem ADTOF Datensatz auf diese Instrumente spezialisieren. Es gibt auch Methoden schnell seine eigenen Datensätze zu generieren. Eine der neuesten Methoden verwendet HMM- und Viterbi-basierte Alignment-Verfahren, um synthetische Audiosignale zu erzeugen und so zusätzliche Daten für das Training von KI-Modellen bereitzustellen [Joysingh, Vijayalakshmi und Nagarajan 2019].

2.4 Polyphone AMT-Systeme

Die meisten früheren AMT-Systeme befassten sich mit monophonen Musikstücken oder höchstens zweistimmigen Ansätzen. Der Großteil realer Musikwerke ist jedoch polyphon, das heißt, es erklin-

gen mehrere Stimmen gleichzeitig. Die Polyphonie stellte und stellt bis heute eine der größten Herausforderungen der automatischen Musiktranskription dar. Sie bringt zahlreiche Probleme mit sich, etwa komplexere Spektrogrammmuster, höheren Rechenaufwand und überlappende Stimmen. Eine Lösung für polyphone Musikstücke markierte daher einen wichtigen Meilenstein in der Entwicklung von AMT-Systemen. Aus diesem Grund widmeten sich mehrere Forscher dieser Fragestellung und entwickelten innovative Ansätze, wie etwa das Blackboard-System oder RASTA-basierte AMT-Systeme, die in diesem Kapitel vorgestellt werden.

2.4.1 Blackboard-System

Zunächst wird das Martin AMT-System [Martin 1996] vorgestellt. Sein Ansatz, zur Lösung von polyphonen Musikstücken, war ein Blackboard-System. Dieses integrierte er in sein AMT-System mit innovativen Modulen und Ansätzen.

Zur Verarbeitung des Input Audiosignals nutzt Martin ein Correlogram. Im Gegensatz zum Spektrogramm, das die Energieverteilung über Frequenzen zeigt, hebt das Correlogram durch Autokorrelation vor allem periodische Strukturen und Tonhöhen hervor. Nachdem das Input Audiosignal in einem Correlogram verarbeitet wurde, spaltet sich Martins AMT-System, in einen Analysepfad und einen Rhythmuspfad, auf. Dabei konzentriert sich der Analysepfad auf die Zusammenhänge der verschiedenen Noten, während der Rhythmuspfad sich mit der Lautstärke und den Notenanschlägen beschäftigt. Am Ende werden diese Kenntnisse zusammengeführt. Der Output besteht aus dem Onset, der Dauer und der Frequenz jeder gespielten Note. Martins AMT-System gibt nicht explizit zurück, welcher Stimme jede Note zugeordnet ist. Durch das Blackboard-System und insbesondere durch die sequentielle Analyse der Intervalle lassen sich die erkannten Noten nachträglich den einzelnen Stimmen korrekt zuordnen.

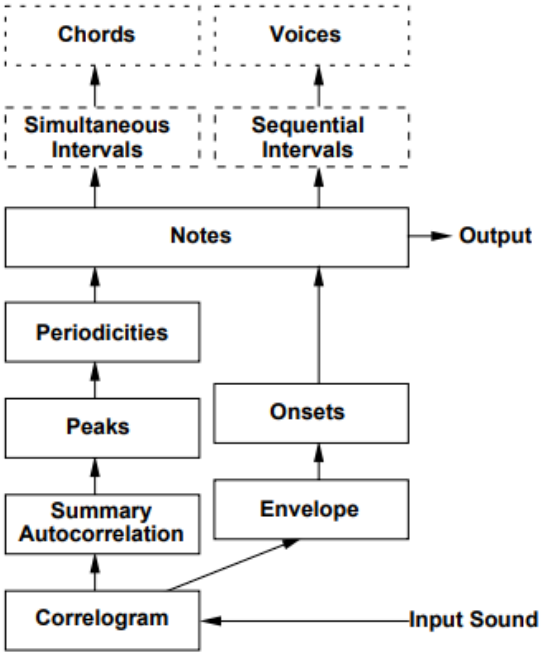

Abbildung 4: Systemstruktur des AMT-Ansatzes von Keith D. Martin, aus [Martin 1996].

Martin nutzt ein Blackboard-System. Dieses besteht aus zwei Hauptbestandteilen. Der erste Teil ist die Sinuskurvenanalyse, mit der die für die Hypothesenbildung notwendigen Daten gewonnen werden. Zu ihr gehören das Correlogram, das Envelope, die Summary Autocorrelation und die Peaks. Durch diese Verarbeitungsschritte können zentrale Eigenschaften der Noten bestimmt werden, die als Grundlage für weitere Analysen dienen.

Zum zweiten Teil gehören die Module Onsets, Periodicities sowie Simultaneous und Sequential Intervals. Diese Module arbeiten kollaborativ auf einem gemeinsamen Datenraum, indem sie Hypothesen über die ermittelten Noten aufstellen und sich gegenseitig mit zusätzlichen Informationen versorgen. Auf diese Weise lassen sich die erkannten Noten später einzelnen Stimmen zuordnen.

Der Analysepfad beginnt bei dem Correlogram. Hier wird der Input, ein reales Audiosignal, zunächst verarbeitet und dann in einem Correlogram dargestellt. Dieses visualisiert die Korrelation der gegebenen Noten in Abhängigkeit der Zeit, Frequenz und Verzögerung. Die Verzögerung soll in diesem Fall darstellen, wann sich ein bestimmtes Signal wiederholt. Dabei ist das wiederholte Signal dasjenige, das dem Ursprungssignal maximal ähnelt, was bedeutet, dass der Korrelationswert an dieser Stelle am höchsten ist. Heißt, wenn ein Ton mit der Frequenz x gespielt wird und dieser sich nach beispielsweise 10ms wiederholt, wird das rechnerisch durch die Verzögerung dargestellt. Als Nächstes wird mithilfe der Summary Autocorrelation das Correlogram komprimiert und die Datenstruktur normalisiert. Dadurch entsteht eine stabile Grundfrequenz, sodass die weiteren Schritte das Correlogram effizient untersuchen können. Nun werden, basierend auf der normierten Struktur, die Peaks gesucht. Diese Peaks deuten darauf hin, wann periodische Komponenten auftreten. Dabei geben sie nicht den Onset der Noten zurück, sondern nur die generelle Frequenz zu einer bestimmten Zeit. Darauf werden mehrere Peaks, die regelmäßig wiederkehren, gruppiert. So können Muster in der gespielten Musik erkannt werden und kurzzeitige Störfaktoren ausgeschlossen werden.

Im Rhythmuspfad wird wiederum die zeitliche Analyse der Noten durchgeführt. Zunächst wird mit dem Envelope die Lautstärke des Signals zu jedem Zeitpunkt festgelegt. Dies dient als Grundlage, um anschließend die Onsets der Noten zu bestimmen. Jeder Onset wird präzise im Envelope, durch einen plötzlichen Anstieg der Lautstärke, dargestellt.

Nachdem die beiden Pfade durchlaufen wurden, liegen bereits Noten vor, die grundsätzlich in Notenschrift transkribiert werden können. Musikalisch sind diese Noten jedoch noch nicht interpretiert. In der Abbildung-(4) gibt es noch die Bestandteile Simultaneous und Sequential Intervals. In dem Modul Simultaneous Intervals wird ermittelt, welche Töne gleichzeitig erklingen. Diese können, wenn sie harmonisch zueinander sind, zu einem Chord gebunden werden. Sequential Intervals analysieren dahingegen, welche Töne nacheinander gespielt werden und möglicherweise eine Melodie bilden könnten. Dies erfolgt durch Tonhöhenverläufe, Pausen und Frequenzunterschiede. Dadurch können Melodien, Basslinien oder Begleitungen voneinander getrennt werden. Zum Schluss werden dadurch die verschiedenen Stimmen aufgetrennt.

2.4.2 RASTA-basiertes System

Ende 1997 publizierte Klapuri seine Masterarbeit „Automatic transcription of Music“ [Klapuri und Eronen 1998]. In seiner Arbeit benutzt er ein neues RASTA-basiertes Verfahren, zur Unterdrückung nicht harmonischer Signalkomponenten. Somit konnten auch Musikstücke mit nicht harmonischen Instrumenten, wie Trommeln, transkribiert werden. Zudem führte er ein Modul ein, zur Schätzung der Anzahl an gleichzeitigen Stimmen, die in dem Musikstück vorkommen. Klapuris AMT-System konzentriert sich, im Gegensatz zu Martins AMT-System, mehr auf die Robustheit gegenüber echten polyphonen Audioaufnahmen mit Rauschen, Störgeräuschen und nicht harmonischen Instrumenten. Die Struktur des Systems ist linear aufgebaut und präzise in dem Aufgabenfeld, für die es entwickelt wurde.

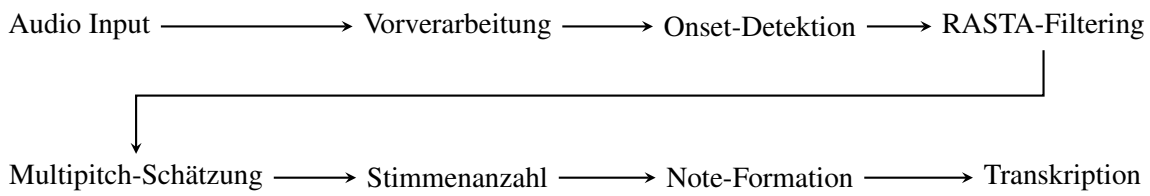

Abbildung 5: Eigene Darstellung: Struktur von Klapuris RASTA-basiertem AMT-System

Klapuris System ist sequentiell aufgebaut und besteht aus sechs Modulen. Als Input wird eine reale Audioaufnahme genutzt, welche in ein Monosignal umgewandelt wird. Dieses Monosignal wird mithilfe von STFT transformiert, sodass durch das entstandene Spektrogramm Zeit, Frequenz und Amplitude der gegebenen Noten ausgelesen werden kann. Anhand dieser Daten werden als Nächstes die Onsets der Noten bestimmt. Klapuri nutzt zur Onset-Erkennung ein Schema von Eric D. Scheirer [Scheirer 1998]. Dabei wird das Audiosignal in Frequenzbänder aufgeteilt und in jedem Band die zeitliche Änderung der Energie, die sogenannten „Energie-Deviate“, berechnet. Anschließend werden die resultierenden Energieänderungen der verschiedenen Bänder summiert. Die nun entstehenden Peaks werden als Onsets der Noten interpretiert. Als Nächstes werden unharmonische Instrumente und transiente Geräusche, wie zum Beispiel Rauschen, entfernt. Dies erfolgt mithilfe der RASTA-Filterung, welches auf das spektrale Signal, vom gegeben Spektrogramm, wirkt. Mehr zu der RASTA-Filterung wird im folgenden Abschnitt im Detail erläutert. Um mehrere Töne, die gleichzeitig im Signal erklingen, zu erkennen wird „harmonic matching“ genutzt. Die Multipitch-Schätzung funktioniert so, dass zu einer bestimmten Zeit immer die dominanteste Tonhöhe, basierend auf der Energieverteilung im Signal, gesucht wird. Sobald diese ermittelt wurde, wird sie vom Signal subtrahiert, und das Verfahren wird erneut angewendet, bis jeder signifikante Pitch untersucht ist. Jetzt wird die Stimmenanzahl geschätzt. Dieses Modul wird im übernächsten Abschnitt detailliert erklärt. In dem Modul Note-Formation werden jetzt die gegebenen Daten zusammengeführt und in eine MIDI-Datei zusammengefasst. Diese MIDI-Datei wäre jetzt bereit in Notenschrift transkribiert zu werden.

RASTA-Filterung

Die RASTA-Filterung ist eine der neuen Module von Klapuri. Sie bewirkt, das nicht-harmonische Störsignale, wie Rauschen, unharmonische Instrumente und vom Musikstück unabhängige Geräusche ausgefiltert werden. Dadurch werden die harmonischen Komponenten des Stückes hervorgehoben, wodurch folgende Module einfacher weitere Eigenschaften den gegebenen Noten zuordnen können. Zudem stärkt dies die Robustheit der Multipitch-Schätzung, in der mehrere Töne zur gleichen Zeit im Audiosignal erkannt werden müssen. Dieses Verhalten wird erzielt, indem ein Filter gesetzt wird, der analysiert, wie laut jede Frequenz zu jedem Moment ist. Nach der Berechnung gibt der Filter eine Lautstärke-Kurve zurück. Alle Frequenzanteile, die entweder zu kurzzeitig sind, wie etwa Claps oder Hi-Hats, oder zu langanhaltend, wie dauerhaftes Rauschen, werden durch einen Bandpassfilter aus dem Audiosignal herausgefiltert. Dadurch bleiben vor allem die zeitlich stabilen, musikalisch relevanten Komponenten erhalten.

Multipitch-Schätzung

Auch die Stimmanzahlschätzung ist ein neu eingefügtes Modul von Klapuri. Vor allem bei der Multipitch-Schätzung ist dieses sehr wichtig, da das System dadurch ein Bild davon bekommt, wie viele Töne gleichzeitig erklingen können. Die Multipitch-Schätzung zieht den spektralen Abdruck eines erkannten Tons vom Audiosignal ab und sucht im verbleibenden Signal nach weiteren Tönen. Ohne zusätzliche Information ließe sich jedoch nicht zuverlässig bestimmen, wann dieser Vorgang beendet werden sollte. Hier greift die Stimmanzahlschätzung ein. Diese analysiert das neue Audiosignal nach jeder Multipitch-Schätzung und gibt aus, ob in der spektralen Energie noch Noten zu erkennen sind. Dies ordnet die Stimmanzahlschätzung durch drei Fragen ein:

- Gibt es in der spektralen Energie noch typische Muster von harmonischen Klängen?
- Wie viele Stimmen wurden bereits erkannt?
- Tragen die weiteren extrahierten Stimmen noch wesentlich zur Erklärung des gesamten Musikstückes bei?

Dadurch extrahiert die Multipitch-Schätzung nur die nötigen Noten und unnötig Störfaktoren werden ausgelassen. Zudem lässt sich eine ungefähre Einschätzung darüber gewinnen, wie viele Stimmen in dem jeweiligen Musikstück erklingen.

2.5 Einbindung von Künstlicher Intelligenz

Für eine lange Zeit basierten AMT-Systeme größtenteils auf Signalverarbeitenden Algorithmen, wie zum Beispiel HMMs (2.2). Doch auch diese Systeme blieben vom zunehmenden Aufkommen KI-basierter Lösungsansätze nicht unberührt. 2010 bis 2012 wurden die ersten Ansätze von AMT-Systemen erstellt, welche maschinelles Lernen nutzten [Eyben u. a. 2010]. Oft wurden KI-Modelle nur für bestimmte Teilprobleme genutzt wie Onset Detection, Pitch Estimation oder Instrument Classification. Diese Entwicklung setzte sich fort, sodass heutzutage nahezu jedes AMT-System eine Form von KI-Modell nutzt.

Einige AMT-Systeme nutzen auch mehrere KI-Modelle gleichzeitig. Solch ein AMT-System wird auch in einer 2016 veröffentlichen Arbeit beschrieben [Sigtia, Benetos und Dixon 2016]. Es ist eins der ersten KI basierenden Ent-to-End AMT-Systeme, welches polyphone Musikstücke transkribieren kann. Der KI-Ablauf des AMT-Systems ist folgendermaßen aufgebaut. Ein CNN bekommt als Input ein Log-Mel-Spektrogramm des Audiosignals bereitgestellt und entzieht diesem bestimmte Merkmale der Noten. Diese Merkmale werden einem RNN weitergegeben, welches dadurch die zeitlichen Abhängigkeiten herausfiltert. Als Nächstes wird durch Frame-wise Multi-Label Classification vorhergesagt, welche Noten zu welcher Zeit aktiv sind. Als Output entsteht eine Binärmatrix, die den Pitch in Abhängigkeit von der Zeit darstellt. Das System wird mit dem MAPS Datensatz durch Supervised Learning trainiert. Diese Arbeit stellt den drastischen Übergang von heuristischen, regelbasierten AMT-Systemen zu selbstlernenden AMT-Systemen dar. Heutzutage sind CNNs und RNNs in Kombination, oder ähnliche KI alternativen, kaum aus der Struktur von AMT-Systemen wegzudenken.

Viele Wissenschaftler haben sich über die Jahre mit AMT-Systemen beschäftigt. Dadurch konnten nachfolgende Arbeiten neue Module und Ansätze für sich nutzen, um deren Entwicklung weiter voranzutreiben. Auch heutzutage ist dies immer noch der Fall, wodurch es stets neue AMT-Systeme mit neuen Ansätzen gibt. Diese neuen, auf KI basierenden Systeme, haben alle jeweils ihre eigenen Stärken und Schwächen. Auch wenn die Forschung von AMT-Systemen in den letzten Jahren zahlreiche Fortschritte erfuhr, bestehen dennoch große Herausforderungen und ungelöste Aufgaben. Im folgenden Kapitel wird genauer der momentane Stand von AMT-Systemen behandelt. So werden anhand einiger unterschiedlichen Beispiele, verschiedene AMT Strukturen dargestellt. Zudem werden die relevantesten Hindernisse in der heutigen automatischen Musiktranskription aufgegriffen.

3 Hindernisse Moderner AMT-Systeme

Trotz der Einführung von KI-Modellen bestehen weiterhin zahlreiche Probleme, die bislang ungelöst oder nur unzureichend gelöst sind. Zudem agieren KIs völlig anders als normale Algorithmen. Dadurch entstehen einige zusätzliche Herausforderungen die behandelt werden müssen. In den folgenden Abschnitten werden die größten und wichtigsten Probleme, mit der sich die AMT Forschung momentan auseinandersetzt, aufgezählt.

3.1 Onset Detection

Die On-/Offset Erkennung von Noten wurde schon ausgiebig in zahlreichen Arbeiten behandelt. Trotzdem ist diese noch nicht komplett akkurat. Das liegt daran, dass On-/Offsets an Lautstärkesprüngen und spektralen Änderungen erkannt werden. Bei polyphonen Musikstücken mit vielen verschiedenen Noten ist es schwieriger diese Unterschiede zu erkennen. Lautstärkesprünge werden ungenauer, da häufig mehrere Noten während des Onsets einer bestimmten Note spielen. Spektrale Änderungen stehen für Veränderungen der Energieverteilung. Einer der ausschlaggebendsten Anteile ist die Spektrale Fluktuation. Diese stellt den plötzlichen Anstieg von Energie in bestimmten Frequenzbändern dar. Wenn mehrere einzelne Töne hintereinander, auf einem Klavier, gespielt werden, kann der Onset der gespielten Noten leicht ermittelt werden. Bei Instrumenten, wie einer Geige, können dabei Probleme entstehen. Auf Geigen können Noten gebunden gespielt werden, was zu einem Unterschied in der Frequenz, jedoch nicht in dem Energie level führt, da die Stärke des Bogenstrichs gleichmäßig bleibt. Zudem kann eine Note, durch Crescendo, zunächst leise gespielt werden und mit der Zeit an

Lautstärke zunehmen. Dieser Onset hat keinen Energie-Peak und somit erkennt das System hier auch nur schwierig eine scharfe Kante. So welche Töne, welche keinen starken Einschlag haben, werden als nicht-perkussiv bezeichnet. Ein weiteres Problem der Onset Erkennung sind nachwirkende Geräusche oder Störgeräusche. Bleiben wir bei dem Beispiel einer Geige. Wenn ein Ton gespielt wird und der Bogen von der Geige abgehoben wird entsteht eine Resonanz des Tons. Das führt zu einem nachklingen des Tons, was sehr natürlich passieren kann, jedoch meist kein beabsichtigtes Merkmal eines Tons ist. Das KI-Modell sieht dieses nachklingen aber noch als der gespielte Ton und erkennt nicht perfekt das Offset des Tons. Zudem könnten Töne von anderen Instrumenten verdeckt werden, wodurch das System den Onset nicht erkennt. Ähnlich kann dies auch ein Problem darstellen, wenn im Audiosignal ein starkes Rauschen besteht.

3.2 Polyphonie

Durch die Nutzung von polyphonen Musikstücken sind, wie schon bei der Onset Erkennung angemerkt, einige Probleme schwerwiegender und deutlicher geworden. Ein weiteres Problem, das gezielt von diesen Anwendungsfällen abhängt, ist die Verwechselung von Noten im Audiosignal. Einige Noten haben sehr ähnliche Frequenzen. Wenn diese Noten gleichzeitig gespielt werden, ist es schwieriger für das System, die korrekten Noten herauszuhören. Neuronale Netzwerke helfen hierbei deutlich, da diese nicht nur das Spektrum zur Analyse einbeziehen, sondern auch auf einer zeitlichen und harmonischen Ebene den Kontext zuordnen können und somit die wahrscheinlichsten nächsten Noten zurückgeben können. Trotz dessen scheitert auch KI an der Einordnung der richtigen Noten, wenn die gespielten Noten eine sehr ähnliche Frequenz besitzen oder die Akkorde zu unterschiedlich zu den Trainingsdaten sind. In der Referenzierten Arbeit [Marták, Kelz und Widmer 2022] werden diese Probleme nochmals deutlicher aufgegriffen.

3.3 Spezielle Instrumente

Oft ist die Qualität der Transkription auch abhängig von den vertretenen Instrumenten in dem Audiosignal. Ethnische Instrumente zum Beispiel sind Instrumente die einer bestimmten Kultur angehören und in der westlichen Kultur weniger vertreten sind. Dadurch gibt es auch weniger Datensätze, welche diese Instrumente beinhalten. KIs brauchen Trainingsdaten und ohne diese können sie bestimmte Instrumente nicht richtig zuordnen. Die meisten Datensätze zur Musiktranskription bestehen hauptsächlich aus Klavier und Geigen Noten. Instrumente wie Flöten oder Orgeln können von diesen Noten gut abgeleitet werden, da sich deren Struktur deutlich ähnelt. Eine weitere Gruppe von Instrumenten die Probleme bei der Transkription bereitet sind elektronische Instrumente. Wenn zum Beispiel eine E-Gitarre gespielt wird, kann diese Effekte nutzen, welche nicht üblich in klassischen Klavier Datensätzen vorkommen. Somit kann die KI diese Töne nicht korrekt erkennen. Ein weiteres Instrument welches Schwierigkeiten bringt, ist der Gesang. Jede Stimme ist einzigartig und vor allem nicht konstant. Wenn ein Ton auf einem Klavier gespielt wird, besitzt dieser, wenn das Klavier richtig gestimmt ist, immer die gleiche Frequenz. Ein Mensch kann jedoch nicht jeden Ton immer komplett perfekt singen, wodurch eine große Varianz an Tönen entsteht. Zudem verläuft der Klang einer Stimme von einer Note zur nächsten. Es gibt nicht immer starke Peaks zur Onset Erkennung. Gesang ist auch, wie die vorher genannten Instrumentgruppen, nicht sonderlich vertreten in größeren Datensätzen. Die folgenden Paper konzentrieren ihren Fokus speziell auf das Thema des Gesangs in der Musiktranskription [Gu, Zeng u. a. 2023; Gu, Ou u. a. 2024]. Somit wurde eine Onset Erkennungsgenauigkeit von ungefähr 80% festgestellt, für Gesang.

Die folgende Tabelle stellt einige traditionelle Instrumente aus verschiedenen ethnischen Gruppen dar. Diese Tabelle ist nicht vollständig und dient ausschließlich dazu, die Vielfalt der Instrumente in unserer Welt zu veranschaulichen.

Region	Instrumente
Europa	Klavier, Violine, Gitarre, Flöte
Arabische Welt	Oud, Qanun, Nay, Darbuka
Türkei	Bağlama (Saz), Kemençe, Zurna, Davul
Persien / Iran	Tar, Setar, Santur, Tombak
Indien	Sitar, Tabla, Harmonium, Bansuri
China	Guzheng, Erhu, Pipa, Dizi
Japan	Koto, Shamisen, Shakuhachi, Taiko
Indonesien	Gamelan, Kendang, Suling, Rebab
Afrika	Kora, Djembe, Balafon, Mbira

Tabelle 2: Übersicht populärer Instrumente verschiedener Weltregionen

3.4 Notendauer

Neben dem Onset einer Note muss auch erkannt werden, wie lange eine bestimmte Note gespielt wird. Eine Schwierigkeit dabei ist der, im Abschnitt-(3.1) angesprochenen, Nachhall einer Note. Dieser muss auseinandergehalten werden von der wirklichen Notendauer. Es gibt jedoch bei einigen Instrumenten, wie zum Beispiel dem Klavier, ein spezielles Problem. Wenn während des Spielens einer Note das Dämpferpedal eines Klaviers gedrückt wird, gibt es keinen klaren Punkt, an dem der Übergang zwischen der Note und dem Nachhallen eindeutig eingeordnet werden kann. Die Lautstärke sinkt dabei nicht abrupt, sondern nur langsam ab. Natürlich ist dieses Problem in polyphonen Musikstücken deutlich stärker, da sich dort die verschiedenen Nachhallphasen überlappen. Dies ist eins der grundlegendsten Probleme, zusammen mit der Onset Erkennung [Jamshidi u. a. 2024].

3.5 Reale Audioaufnahmen

Ein perfektes AMT-System sollte sogar auf realen Audioaufnahmen 100% Genauigkeit besitzen. In der Realität funktioniert dies jedoch noch nicht, da Live-Audioaufnahmen Hintergrundrauschen enthält. Dies war schon vor der Einführung von KI ein Problem und wurde durch die Einbindung von KI-Modellen nicht sonderlich verbessert. Das liegt daran, dass KIs mit isolierten Studioaufnahmen, wie es im MAESTRO Datensatz der Fall ist, trainiert werden. Natürlich gibt es auch Datensätze mit realen Audioaufnahmen, jedoch bräuchte es dafür viel mehr Trainingsdaten und vor allem auch eine große Anzahl von unterschiedlichen Hintergrundgeräuschen, damit die KI auf alle Möglichkeiten trainiert wird. Um dem entgegenzuwirken, können Studio-Datensätze wie MAESTRO durch Data-Augmentation variiert werden, wodurch realistischere Audioaufnahmen erstellt werden können. So wurden auch die Trainingsdaten in folgender Arbeit erzeugt [Kusaka und Maezawa 2024]. Es wird Timber, Reverb, Noise variierte und die Qualität des Aufnahmegerätes angepasst. So werden zahlreiche neue Datensätze kreiert, die passend auf die gegebenen Anwendungsfälle ausgelegt sind.

3.6 Frame-basierte vs Event-basierte AMT-Systeme

Die meisten AMT-Systeme sind Frame-basiert und nicht Event-basiert. Frame-basiert bedeutet, dass die Analyse des Audiosignals in Zeitfenster, von ungefähr 20ms, aufgeteilt wird. Ein KI-Modul, wie zum Beispiel ein RNN, analysiert diese Zeitfenster einzeln, in Abhängigkeit von den vorherigen Zeitfenstern. Falls nun zum Beispiel ein Onset einer Note genau zwischen zwei Zeitfenstern liegt, wird dieser verschoben oder verzerrt, sodass das Onset klar in einem der gegebenen Zeitfenstern liegt. Event-basierte AMT-Systeme hingegen fragen immer ab, was das nächste musikalische Ereignis im Audiosignal ist und reagieren dementsprechend. Somit müssen keine musikalischen Merkmale verschoben werden, da sie ihre eigene Position besitzen und nicht abhängig von Zeitfenstern (Rastern) sind. Dadurch entspricht der Output viel mehr dem Input, in relation zu der zeitlichen Abfolge. Im Ergebnis sind Event-basierte AMT-Systeme somit Frame-basierten AMT-Systemen überlegen. In der Realität sind die meisten AMT-Systeme trotzdem Frame-basiert. Das liegt daran, dass Event-basierte AMT-Systeme eine sehr neue Entwickelung sind und zudem auch schwerer zu implementieren sind.

Frame-basierte Methoden wurden ungefähr im Jahre 2000 entwickelt und haben sich seitdem in zahlreichen Arbeiten durchgesetzt [Martin 1996; Klapuri und Eronen 1998]. Dahingegen sind Event-basierte Methoden erst ungefähr 15 Jahre später entwickelt worden [Magenta Team 2017]. Dadurch waren Frame-basierte Methoden lange der Standard für AMT-Systeme. Zudem ist die Umsetzung von Event-basierten AMT-Systemen komplizierter als Frame-basierte. Das System muss jedes Event korrekt einschätzen und zuordnen. Zum Beispiel könnte hintergrund Rauschen als Note erkannt werden und somit im Gedächtnis eines RNNs, als Note, gespeichert werden. In Event-basierenden Systemen wird eine Note nur einmal für die nachfolgenden Frames, in denen sie klingt, vorhergesagt. Heißt es existiert eine falsche Note in der Transkription, die auch nicht mehr von KI-Modellen oder anderen Algorithmen entfernt wird. Bei der Frame-basierten Transkription hingegen wird eine Note in jedem Zeitfenster neu analysiert, wodurch fehlerhafte Noten öfter erkannt werden. Eine falsche Note wirkt sich ausschlaggebend auf das weitere Audiosignal aus, da zum Beispiel RNNs diese Noten teilweise im Gedächtnis speichern und daraus Rückschlüsse auf zukünftige Noten führen. Auch das Training von Event-basierten AMT-Systemen kann nicht parallel verlaufen, sondern muss tokenweise abgearbeitet werden. Deswegen sind Event-basierte AMT-Systeme eine größere Herausforderung. Sie führen deutliche mehr Risiken mit sich, geben dafür aber auch ein besseres Ergebnis zurück, wenn alles perfekt funktioniert. Da irgendwann ein perfektes AMT-System entstehen sollte ist der Übergang von Frame zu Event mit der Zeit unausweichlich, da somit der Inhalt von Musikstücken präziser wiedergeben werden kann und ein Ergebnis erzielt wird, welches durch Frame-basierte transkription sonst nicht möglich wäre. Die Forschung im Bereich der Event-basierten Transkription hat sich in den letzten Jahren deutlich intensiviert.

3.7 Blackbox von KI-Modellen

Das letzte Problem betrifft KI im generellen. Ein neuronales Netz arbeitet mit sehr hochdimensionalen Räumen, die für uns Menschen nicht begreifbar sind. Selbst wenn sich jemand alle Gewichtungen in einem neuronalen Netz anschauen würde, würde dieser keinen Zusammenhang feststellen können. Dies stellt auch ein Problem in der Musiktranskription dar, da somit nicht erfasst werden kann, welche Einstellungen eine AMT-KI genau braucht, damit sie alle Musikstücke perfekt transkribieren kann. Um dieses Problem zu lösen, gibt es Methoden, welche die Blackbox ein wenig umgehen. Das Forschungsgebiet der Explainable AI befasst sich damit, KI-Modelle zu entwickeln, welche für den Menschen nachvollziehbar und verständlich sind. Dies geschieht durch verschiedene Methoden und Darstellungen, welche den Denkfluss einer KI veranschaulichen sollen. Einige von diesen Methoden finden ihre Anwendung auch in KI-basierten AMT-Systemen wieder. Methoden die zur Nutzung in AMT-Systemen diskutiert werden, sind „Concept Activation Vectors“, „Layer-wise Relevance Propagation“ und „Surrogat-Modelle“. Diese wurden jedoch noch in keinem AMT-System richtig implementiert. Musikalische Konzepte wie Akkorde oder Onsets sind nicht direkt lable bar, wie zum Beispiel Katzen oder Hunde, da diese stark abhängig von spektralen und zeitlichen Mustern sind. AMT-Systeme sind zeitlich-sequenziell und besitzen pro Zeitfenster mehrere Outputs. Viele lokale Änderungen auf ein bestimmtes Zeitfenster, wie den dB-Wert ändern, haben einen globalen Einfluss. Methoden, welche in AMT-Systemen angewendet werden, haben dahingegen Eigenschaften, die durch die Struktur von AMT-Systemen profitieren. Methoden die schon in AMT-Systemen eingesetz werden sind Saliency Maps, Feature-Visualisierung und Attention-Mapping. Saliency Maps stellen dar, wie stark jedes Zeitfenster beeinflusst, das ein bestimmter Ton erkannt wurde. Dies basiert auf dem Gradienten und ist besonders hilfreich bei CNN-basierten Modellen. Feature-Visualisierung beobachtet konkret die einzelnen Neuronen jeder Ebene im neuronalen Netzwerk. Es wird geschaut, welche Neuronen sehr stark bei bestimmten Inputs reagieren. So kann erkannt werden, welche Teile des Netzes für welche Aufgaben verantwortlich sind und ob vielleicht bestimmte Stellen des Netzes gar nicht genutzt werden. Attention-Mapping nimmt den gegebenen Input und gewichtet diesen, je nachdem welche Teile davon wichtig für eine bestimmte Vorhersage sind. So wird erkannt, welche Zeitfenster den meisten Einfluss auf einen bestimmte Onset hatten. Vor allem bei polyphonen Musikstücken können relevante zeitliche Zusammenhänge erkannt werden. Ein gutes Beispiel von Attention-Mapping kann in folgender Arbeit gefunden werden [K. W. Cheuk, Luo u. a. 2021].

4 KI-Modelle in AMT-Systemen

KI-Systeme haben in den letzten Jahren stark an Beliebtheit gewonnen. AMT-Systeme bilden da keine Ausnahmen. Vor allem durch die integration von CNNs und RNNs konnten AMT-Systeme zahlreiche Prozesse deutlich verbessern und neue Errungenschaften in dem Forschungsgebiet erzielen. Es gibt jedoch auch weitere wichtige KI-Modelle die in AMT-Systemen häufig genutzt werden oder zur integration in Planung stehen. In diesem Kapitel werden genau diese KI-Modelle stärker behandelt und deren Aufgaben in der automatischen Musik transkription weiter erläutert.

4.1 Convolutional Neural Network

CNNs sind neuronale Netze, welche besonders gut räumlich strukturierte Daten analysieren können. Deshalb werden diese vor allem in der Analyse von Bildern genutzt. Sie sind in der Lage, den Inhalt eines Bildes präzise zu analysieren und darin enthaltene Objekte zuverlässig zu identifizieren. Auch multimodulare KI-Systeme, wie das System hinter ChatGPT, nutzen weiterentwickelte Formen von CNNs zur Bildanalyse. Im Fall der Musiktranskription wird als Input Bild ein Spektrogramm verwendet. Spektrogramme können ähnlich wie zweidimensionale Bilder gehandhabt werden, da auf diesen auch alle wichtigen Daten des Inputs Audiosignals zu finden sind. CNNs bestehen aus mehreren verschiedenen Layern. Einfache CNN Modelle bestehen nur aus zwei bis fünf Layern, wobei komplexere CNNs aus über Tausenden Layern bestehen können. In AMT-Systemen haben die meisten CNNs etwa drei bis zehn Layer. Diese Layer sind Convolutional Layer, Activation Layer (ReLU), Pooling Layer, Batch Normalization, Dropout Layer und Upsampling. Mit jedem Layer kann ein CNN immer abstraktere Merkmale erkennen. Außer dem Convolutional Layer und Activation Layer sind die anderen Layer jedoch nicht unbedingt notwendig. Eine Arbeit, die die Stärke von CNN-Modellen in AMT-Systemen sehr gut darstellt, heißt „Onsets and Frames“ [Hawthorne u. a. 2017]. In dieser Arbeit werden direkt zwei verschieden spezialisierte Teilnetzwerke für Onsets und Sustain der Noten behandelt. Mit diesem System wird die Entwicklung des Forschungsgebietes illustriert. Insbesondere für polyphone Klaviertranskription ist dieses AMT-System ausgezeichnet. Im Folgenden werden die verschiedenen Layer eines CNNs, in einem AMT-System, erläutert.

4.1.1 Layer eines CNNs

In dem Convolution Layer werden Filter verwendet. Filter sind 2D-Matrizen, die aus trainierbaren Gewichten bestehen. Ein Filter deckt jeweils einen Eingabebereich des Inputbildes ab. Je nach Aufgabenfeld und Rechenaufwand unterscheidet sich die größe eines Filters. Ein 3x3 Pixel Filter ist der am weitesten verbreitete Standard. Je größer ein Filter ist, desto mehr Rechenaufwand wird benötigt. In AMT-Systemen werden häufig 5x3 oder 7x3 Pixel Filter genutzt, da somit mehr zeitlicher Kontext als Frequenzkontext abgedeckt wird. Jeder auf das Bild angewendete Filter wird über die gesamte Eingabe verschoben und analysiert dabei nacheinander alle Bildbereiche. Ein Skalarprodukt aus Filter und Eingabebereich beschreibt dann einen Aktivierungswert. Aus allen Aktivierungswerten eines Filters entsteht eine Feature Map. Beim Vergleich von Aktivierungswerten können Patterns und Eigenschaften erkannt werden. In der Musiktranskription werden dadurch Onsets, Sustains oder harmonische Verläufe herausgefiltert. Onsets werden beispielsweise erkannt, wenn es eine plötzliche Energieänderung gibt. Am Ende entsteht ein 3D-Tensor, welcher alle Feature Maps beinhaltet.

Die Batch Normalization sorgt dafür, das die Aktivierungswerte normalisiert werden. Jede Feature Map wird dabei einzeln normalisiert. Dadurch wird Rechenleistung eingespart. Zudem lassen sich im Training durch Mini-Batches mehrere Spektrogramme gleichzeitig durch die CNN-Struktur leiten. Dadurch wird das Training schneller und Ergebnisse werden früher erreicht.

Es kann passieren das die Summe eines Convolution Layers negativ ist. Dies kann passieren, wenn stärker gewichtete Filter einen negativen Aktivierungswert herausgeben. Negative Werte können zu Informationsverlust, von Eigenschaften des Musikstückes, führen. Um das zu vermeiden werden im Activation Layer, meistens mit der ReLU Funktion, alle negativen Aktivierungswerte auf 0 gesetzt. So kann das Netz nichtlineare Beziehungen modellieren.

Als Nächstes wird mit dem Pooling Layer der Rechenaufwand verringert. Dieser nimmt jede Feature Map einzeln und reduziert ihre Matrix zu einer kleineren, standardmäßig 2x2, Matrix. Das erfolgt, indem sich der Pooling Layer zunächst eine gesamte Feature Map nimmt und diese dann in kleinere Blöcke aufteilt. Es gibt entweder Max-Pooling oder Average-Pooling. Je nachdem welche Methode gewählt wird, wird immer der höchste Wert oder der durchschnittliche Wert extrahiert. Der extrahierte Wert von jedem Block wird nun in die reduzierte Feature Map zurückgeführt. Dadurch wird nicht nur Rechenaufwand reduziert, sondern auch Überanpassung vermieden. Wenn das System jeden kleinsten Wert berücksichtigt, passt es sich zu sehr an die Trainingsdaten an und kann womöglich andere Daten nicht mehr richtig analysieren.

Der Dropout Layer ist, im Gegensatz zu den anderen Layern, nur im Training relevant. Er schaltet zufällig Neuronen aus, sodass sich Neuronen nicht ausschließlich auf bestimmte andere Neuronen verlassen können. Somit wird das gesamte neuronale Netzwerk robuster und vielseitiger.

Upsampling ist das Gegenteil von dem Pooling Layer. Anstatt die Feature Maps zu reduzieren, werden diese wieder hochskaliert. Dadurch lassen sich bestimmte Features wieder zeitlich präziser bestimmen, da die Zeitfenster wieder genauer zum originellen Audiosignal passen. Meist wird dieser Layer jedoch weggelassen, da er häufig nicht sehr relevant für AMT-Systeme ist.

Der Output eines CNNs ist ein 3D-Tensor:

$$\text{CNN-Ausgabe} \in \mathbb{R}^{T \times F \times C}$$

- T: Anzahl der Zeitfenster (Frames) im Spektrogramm.
- F: Frequenzachsenlänge.
- C: Anzahl der Filter des CNNs.

In der folgenden Darstellung wird ein CNN dargestellt, mit deren gegebenen Layern. Als Input wird beispielhaft ein CQT-Spektrogramm verwendet. Ein Kernel extrahiert einen lokalen Eingabebereich. Durch die Convolution-Layer und ReLU-Layer werden die Merkmale extrahiert und somit entstehen Feature Maps. Diese Feature Maps werden durch Batch Normalization normalisiert und anschließend durch Pooling-Layer sukzessive reduziert. Darauf folgend werden diese Merkmale vektorisiert und durch ein Fully Connected Layer klassifiziert. Am Ende entsteht eine Wahrscheinlichkeitsverteilung über mögliche Klassen. In diesem Beispiel werden als Output Tonhöhen vorhergesagt. Das gleiche CNN könnte aber auch andere musikalische Merkmale extrahieren, je nachdem wie dieses trainiert wurde. Die Darstellung veranschaulicht somit die Merkmalsextraktion mithilfe eines CNNs.

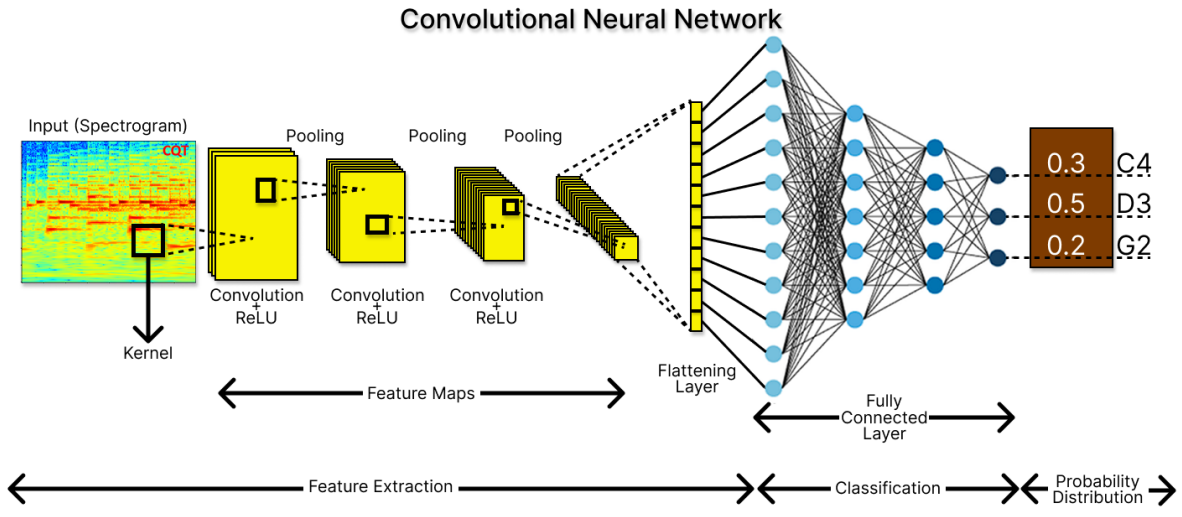

Abbildung 6: Visualisierung einer CNN Struktur zur automatischen Musiktranskription. Eigene Darstellung in Anlehnung an [Shahriar 2020].

Üblicherweise schließt sich, in einem AMT-System, nach einem CNN ein RNN an. Dieses kann jedoch nur 1D-Vektoren verarbeiten und nicht einen 3D-Tensor. Dieser 3D-Tensor wird deshalb vor der Übergabe durch eine Flattening-Operation zeitschrittweise in 1D-Vektoren umgewandelt. Dabei wird der Vektor mit einer Dimension von F × C erhalten. Diese Vektoren werden dann an das folgende RNN weitergeleitet.

4.1.2 Geschichtliche Einordnung von CNNs

CNNs finden ihren Ursprung im Jahre 1979. Die erste Architektur für CNNs wurde, unter dem Namen „Neocognitron“, veröffentlicht [Fukushima 1980]. Das Neocognitron kann zwar als Vorläufer moderner Convolutional Neural Networks betrachtet werden, unterschied sich jedoch wesentlich von späteren CNN-Architekturen. Es dauerte weitere 10 Jahre bis das erste vollwertige Convolutional Neural Network veröffentlicht wurde [LeCun, Boser u. a. 1989]. Dieses CNN Modell unterscheidet sich speziell in drei bestimmten Punkten zu dem Vorgänger Neocognitron. LeCun integriert in seinem CNN Gradientenlernen durch Backpropagation, wodurch das gesamte Netzwerk erstmals auf ein gemeinsames Ziel zu trainieren konnte. Neocognitron besaß zudem, im Gegensatz zu LeCuns CNN, keine Gewichtsverteilung mithilfe von Filtern, wie es in heutigen CNN Modellen Standard ist. Das Neocognitron hatte zudem keine praktische Anwendung. Es stellte zwar ein theoretisch wegweisendes Architekturkonzept dar, doch die damaligen technischen Rahmenbedingungen erschwerten eine praktische Umsetzung. Für AMT-Systeme wurden CNNs erst etwa ab dem Jahre 2015 relevant [Sigtia, Benetos und Dixon 2016]. In dieser Arbeit wurden erstmals polyphone Musikstücke mithilfe von CNNs und weiteren KI-Modelle, transkribiert. Seitdem sind CNNs ein wichtiger Bestandteil der modernen automatischen Musiktranskription.

4.2 Recurrent Neural Network

RNNs sind neuronale Netze, welche entworfen wurden um Daten mit zeitlicher Struktur zu verarbeiten. Dabei ist die Besonderheit von RNNs das diese ein Gedächtnis haben. Wenn in einem AMT-System eine bestimmte Tonabfolge gespielt wurde, kann sich das RNN diese merken und dementsprechend die fortlaufenden Ausgaben anpassen. Dies passiert dank den Hidden States. Diese stellen das Gedächtnis des RNNs dar und sind einer der wichtigsten Faktoren in einem RNN. Die Hidden States werde im nächsten Abschnitt ausführlicher beschrieben. Dieses Prinzip ist in der Musiktranskription sehr hilfreich, da jede Note stark von den vorherigen gespielten Noten abhängt. Takt, Rhythmus, Harmonie und die Melodie eines Musikstückes sind alles gute Beispiele, warum diese sequenzielle Abfolge so passend in AMT-Systemen ist.

RNNs haben in der automatischen Musiktranskription mehrere Aufgabenfelder. Je nachdem wie das RNN trainiert wird, bewältigt dieses alle Aufgaben oder nur einen Teil. Diese Aufgaben bestehen aus Frame-Glättung (Temporal smoothing), Kontext-Modellierung (Temporal context modeling), Feature-Zusammenführung (Sequential integration of acoustic features) und Ausgabevorbereitung (time-distributed output classification). Diese Aufgaben werden auf jedes Zeitfenster, auf jeden 1D-Vektor des CNNs, angewendet. Doch bevor der Fokus auf diese einzelnen Aufgaben gelegt werden kann, muss zunächst geklärt werden, was Hidden States sind.

Hidden States

Hidden States sind das grundlegende Prinzip, warum RNNs funktioniere. Sie stellen das Gedächtnis des RNNs dar und helfen somit anderen Modulen Vorhersagen über bestimmte musikalische Eigenschaften zu treffen. Für jeden 1D-Vektor, den das CNN liefert, wird ein Hidden State erstellt. Diese werden sequenziell, mit Abhängigkeit zum vorherigen Hidden States, definiert.

Die folgende Gleichung beschreibt, wie der aktuelle Hidden State h_t aus dem Eingabevektor x_t und dem vorherigen Hidden State h_{t-1} berechnet wird. Der vorherige Hidden State h_{t-1} repräsentiert nicht nur den unmittelbar vorausgehenden Zustand, sondern auch alle davorliegenden, da diese in die Berechnung jedes neuen Hidden States einfließen. Die Funktion f repräsentiert dabei eine nichtlineare Transformation, mithilfe der Gewichte des RNNs. Somit

wird sichergestellt, dass der neue Hidden State sowohl den aktuellen Eingabevektor x_t, als auch die vorherigen Hidden States h_{t-1} berücksichtigt.

$$\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1})$$

- $\mathbf{x}_t$: Eingabevektor zum Zeitpunkt t.
- $\mathbf{h}_{t-1}$: Vorheriger Hidden State.
- f: Nichtlineare Transformationsfunktion.
- $\mathbf{h}_t$: Neuer Hidden State.

Hidden States enthalten das Wissen der vorherigen Zeitfenster über zeitliche Muster, wie Sustain und Akkordstruktur. Alleine können Hidden States keine eigene Entscheidung über bestimmte Noten fallen. Dafür müssen andere Module die Informationen der Hidden States später richtig verwerten. Jeder Hidden State hat eine Anzahl von Dimensionen, welche gleich der Anzahl der Neuronen im RNN ist.

4.2.1 Grundstruktur eines RNNs

Bei der Frame-Glättung bekommt das RNN als Input die Aktivierungswerte des Zeitfensters. Es betrachtet diese mit Kontext zu den vorherigen Zeitfenstern, um falsche Vorhersagen auszuschließen. Das CNN hat dem RNN schon Vorhersagen für bestimmte Noten gegeben. Diese Vorhersagen könnten jedoch Fehler enthalten. Zum Beispiel kann die Note C4 für den Frame 6 und 8 aktiv sein, aber bei dem Frame 7 hat das CNN diese Note nicht als aktiv angesehen. Es ist sehr unwahrscheinlich, das eine Note für nur einen Frame ausfällt. Solche Arten von Fehlern verarbeitet das RNN über mehrere Zeitfenster hinweg, mithilfe seiner wiederkehrenden Struktur. Dieser Vorgang wird auch als zeitliche Entfaltung bezeichnet. Deshalb aktiviert das RNN den Ton C4 auch im Frame 7.

Die folgende Abbildung zeigt die Entfaltung eines einzigen RNN-Blocks über mehrere Zeitpunkte hinweg. In der Abbildung steht x für den 1D-Inputvektor eines CNNs, h für den Hidden State und dem somit verbundenen Gedächtnis des RNNs und o für den Outputvektor, mit dem weiter gerechnet wird. Die Buchstaben W, U und V stellen dabei die verschiedenen Gewichte „Input-Weight“, „Recurrent-Weight“ und „Output-Weight“ dar.

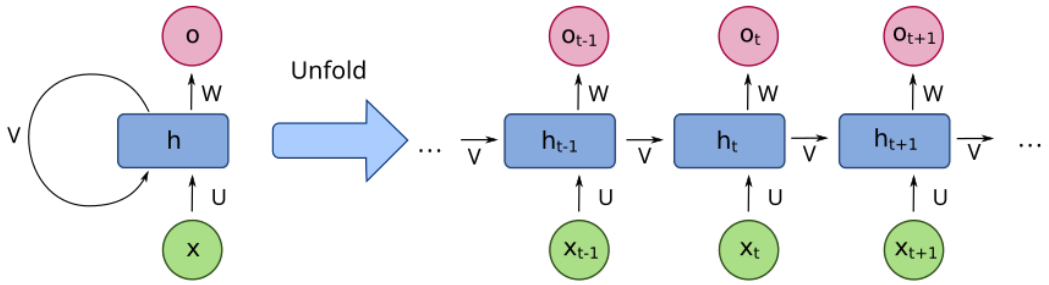

Abbildung 7: Entfaltung eines RNN-Blocks [Wikimedia Commons contributors 2016].

Diese Abbildung illustriert folgende Formel. Dabei steht b für den Bias-Vektor des Hidden Layers und c für den Bias-Vektor des Output-Layers.

$$h_t = \tanh\left(W \cdot x_t + U \cdot h_{t-1} + b\right)$$
$$o_t = \sigma\left(V \cdot h_t + c\right)$$

- x_t, o_t: Eingabevektor und Outputvektor zum Zeitpunkt t.

- h_t, h_{t-1}: Neuer und vorheriger Hidden State.
- W, U, V: Input-Weight, Recurrent-Weight und Output-Weight.
- b, c: Bias-Vektor des Hidden Layers und des Output-Layers.
- $\tanh$: Aktivierungsfunktion, beschränkt den Wertebereich des Hidden States auf $[-1, 1]$.
- σ: Aktivierungsfunktion des Output-Layers.

Neben der Frame-Glättung wird die Kontext-Modellierung auf den gegebenen Input angewendet. Als Input bekommt diese auch die 1D-Vektoren des CNNs. In der Kontext-Modellierung werden größere zeitliche Zusammenhänge betrachtet. So kann die Kontext-Modellierung, mithilfe des Hidden States, den Notenverlauf oder auch die Länge einer Note vorhersehen. Je nach den Trainingsdaten ist es zum Beispiel üblicher das auf C4 ein D3 folgt, was durch die Kontext-Modellierung angepasst wird. Dieser musikalische Kontext ist aber immer stark von der Musikrichtung und von den Musikern abhängig, welche in den Trainingsdaten vorhanden sind. Bei Jazz und Pop oder Johann Sebastian Bach und Taylor Swift unterscheidet sich der Stil der Tonabfolge extremst. Zudem wird Onset, Sustain und Offset stabilisiert. Durch vorherige Beispiele weiß die Kontext-Modellierung, wie lange eine bestimmte Note andauern wird und kann so das Offset der Note einschätzen. Als Output entstehen kontextabhängige Vektoren. Sie haben die gleiche Struktur wie die 1D-Vektoren vom CNN, sind aber entsprechend dem Kontext angepasst.

Die Feature-Zusammenführung ist der letzte wichtige interne Schritt eines RNNs. Bei diesem werden aus lokalen alleinstehenden Informationen eines Zeitfensters, konkrete musikalische Ereignisse. Dadurch schreibt das RNN Ereignisse wie Akkorde, Noten, Onsets und weitere heraus. Dafür muss die Feature-Zusammenführung sich die einzelnen 1D-Vektoren als eine Folge von Events anschauen. Dies passiert über den Hidden State. Das wird vor allem wichtig, wenn später eine MIDI-Datei ausgegeben werden soll, da in dieser auch die einzelnen musikalischen Ereignisse aufgeschrieben sind. Als Output entstehen kontextreiche Vektoren. Diese Vektoren sind, durch die vorherigen Module angepasste und verbesserte, Hidden States.

Die Ausgabevorbereitung ist der letzte Schritt des RNNs, wodurch nun die gesammelten Daten zu echten musikalischen Ereignissen zusammengefügt werden. Als Input werden die, durch die Feature-Zusammenführung verbesserten, Hidden States genutzt. Zunächst werden diese durch ein Fully Connected Layer geschickt.

Ein Fully Connected Layer ist ein Klassifikator, welcher den Hidden States auf eine gewünschte Dimension bringt. Wenn zum Beispiel Klaviertasten vorhergesagt werden sollen, werden alle Hidden States mit 88 Dimensionen ausgestattet. Bei MIDI-Dateien wären es 128 Dimensionen. Die Werte, welche aus dem Fully Connected Layer stammen, sind jetzt noch nicht richtig interpretierbar. Um diese als konkrete und normalisierte Wahrscheinlichkeiten darstellen zu können, wird eine Aktivierungsfunktion eingesetzt. Mit beispielsweise der Sigmoid-Funktion als Aktivierungsfunktion können alle Werte normiert in einem Bereich zwischen 0 und 1 gebracht werden. Sagen wir, wir wollen jetzt die gespielten Klaviertasten vorhersagen. Dann hat jeder Hidden State für alle Klaviertasten einen eigenen Wert mit einer Wahrscheinlichkeit, dass diese Taste zu dem gewählten Moment gespielt wurde. Zum Schluss muss jetzt ein Threshold bestimmt werden. In polyphonen Musikstücken können immer mehrere Noten gleichzeitig erklingen, weshalb nicht einfach die Note mit der höchsten Wahrscheinlichkeit ausgewählt werden kann. Deshalb wird ein Threshold genutzt, zum Beispiel bei 50%, welcher bestimmt, wie viel Prozent eine Note braucht, um als aktiv zu gelten. Durch Postprocessing können einige Eigenschaften wie Rauschen noch herausgefiltert werden. Postprocessing ist jedoch nicht relevant für den KI-Ablauf. Wenn das Ergebnis zufriedenstellend ist, kann es in das gewünschte Output-Format, standardmäßig MIDI-Dateien, eingefügt werden.

Heutzutage sind RNNs nur die grundlegende Struktur. Basierend auf dieser gibt es einige verbesserte Systeme, welche Aktiv in AMT-Systemen und anderen KI-Systemen genutzt werden. Zwei dieser Systeme sind Long Short-Term Memory's (LSTM) und Bidirektionale RNNs (BiRNN). Diese werden im folgendem Abschnitt ausführlicher erklärt.

4.2.2 Long Short-Term Memory

LSTMs sind verbesserte RNNs. Diese kontrollieren durch Gates besser, welche Daten sie wirklich in den Hidden State speichern möchten. Dadurch lässt sich das neuronale Netz noch weiter an die gewünschten Ansprüche anpassen.

In einem einfachen RNN werden alle Daten, egal ob sinnvoll oder nicht, miteinander in dem Hidden State kombiniert. Somit kann sich das RNN langfristig schwieriger Information merken. Wenn zum Beispiel im 2. Hidden State ein Onset erkannt wurde, kann der 20. Hidden State sich das schlechter merken, da viele andere Informationen mitgeschrieben wurden. LSTMs lösen dieses Problem mit Forget, Input und Output Gates und dem Cell State. Der Cell State stellt das Langzeitgedächtnis des LSTMs dar. Er berechnet sich aus den drei Gates. Das Forget Gate bestimmt, welche Informationen aus dem vorherigen Cell State gelöscht werden sollen. Das Input Gate bestimmt, welche neuen Inhalte aus dem neuen Zeitfenster aufgenommen werden. Dabei ist der Cell-candidate die Datenmenge, welche zum Speichern, durch das Input Gate, vorgeschlagen wird. Das Output Gate bestimmt, welcher Teil des Cell States zu dem neuen Hidden State hinzugefügt wird.

$$
\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) && \text{(Forget Gate)} \\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) && \text{(Input Gate)} \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) && \text{(Output Gate)} \\
\tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c) && \text{(Cell-candidate)} \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t && \text{(Cell State)} \\
h_t &= o_t \odot \tanh(c_t) && \text{(Aktueller Hidden State)}
\end{aligned}
$$

Dadurch kann sich ein LSTM die verschiedenen musikalischen Ereignisse, über das gesamte Audiosignal, besser im Zusammenhang merken.

Folgende Abbildung zeigt einen Zeitschritt in einem LSTM. Dabei sind die Komponenten gleich benannt wie in der Abbildung-(7). Das c steht für den Cell-State. Intern in der „LSTM Unit“ stehen die Buchstaben F, I und O für die Gates: „Forget Gate, Input Gate und Output Gate“.

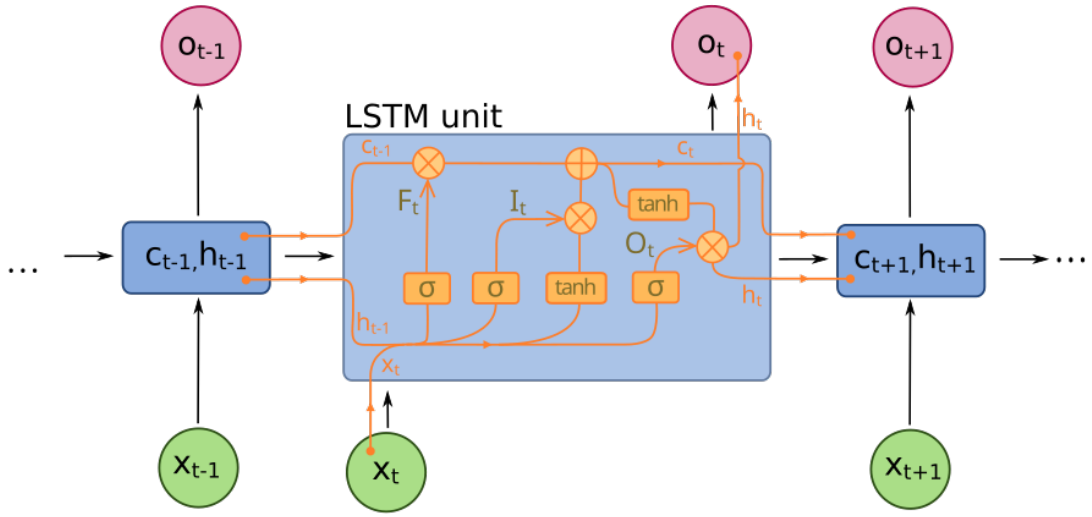

Abbildung 8: Zeitschritt eines LSTMs [Wikimedia Commons contributors 2016].

4.2.3 Gated Recurrent Units

GRUs sind, neben LSTMs, eine weitere spezielle Art von RNNs [Chung u. a. 2014]. Sie wurden erfunden, um bestimmte Probleme von RNNs zu lösen. Insbesondere das Problem des „Vanishing Gradients“ sollten GRUs lösen. GRUs steuern mithilfe von Gates, welche Informationen gemerkt und welche vergessen werden sollen. Sie sind, im Gegensatz zu LSTMs, eher für kleinere Aufgaben geeignet. Dafür sind sie weniger fehleranfällig und haben ein schnelleres Training.

Ein GRU besitzt zwei Gates. Sie haben, genau wie bei LSTMs, die Aufgabe das Gedächtnis des KI-Models zu verwalten. Das Reset Gate schaut sich den alten Hidden State an und entscheidet, wie viel von diesem Wissen in den neuen Hidden State mit einfließt. Das Update Gate hingegen sieht den aktuellen Hidden State und überlegt, welche Informationen davon in das Langzeitgedächtnis mit einbezogen werden. Danach führt das Update Gate die beiden überarbeiteten Hidden States zusammen zum neuen Hidden State.

Das Problem des Vanishing Gradients betrifft das Gedächtnis neuronaler Netze und tritt insbesondere während der Trainingsphase auf. Bei der Backpropagation werden die Gradienten mit jedem durchlaufenen Layer kleiner, sodass betroffene Layer im Netzwerk kaum noch dazulernen. Dies liegt daran, dass Aktivierungsfunktionen Werte im Bereich zwischen 0 und 1 zurückgeben und ihre Ableitungen ebenfalls kleiner als 1 sind. Da die Gradienten bei jedem Schritt mit diesen Ableitungen multipliziert werden, schrumpfen sie exponentiell mit zunehmender Netzwerktiefe. Infolgedessen verliert das Modell die Fähigkeit, Informationen über längere Zeiträume hinweg zu speichern. Bei GRUs wird das Problem des Vanishing Gradients durch die Gates gelöst. Sie stellen eine gewichtete Mischung aus altem und neuem Hidden State dar. Dadurch wird der Gradient nicht ständig verkleinert und wichtige Informationen können über einen langen Zeitraum gespeichert werden.

Folgende Abbildung zeigt einen Zeitschritt in einem GRU. Die Komponenten sind gleich benannt wie in der Abbildung-(7). Intern in der „GRU Unit“ stehen die Buchstaben „R und Z“ für die Gates: „Reset Gate und Update Gate“.

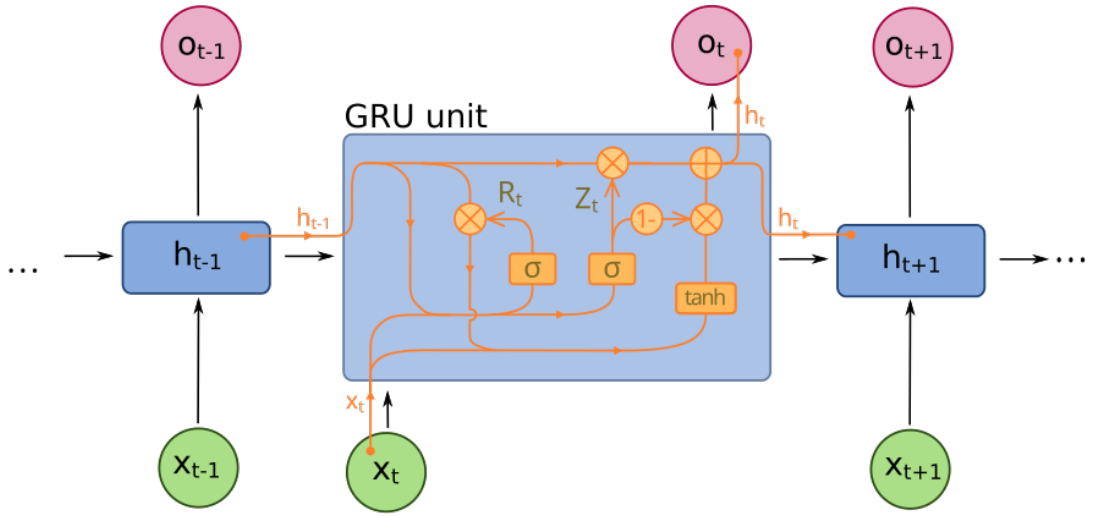

Abbildung 9: Zeitschritt eines GRUs [Wikimedia Commons contributors 2016].

4.2.4 Bidirektionale RNNs

Um ein noch besseres Ergebnis zu erhalten, kann auch ein bidirektionales RNN genutzt werden. Dieses besteht aus zwei RNNs. Eines liest die Zeitfenster von vorne und das andere von hinten ab. Dadurch entsteht die doppelte Menge an Hidden States und die gesamte Vorhersage wird robuster.

Die folgende Formel beschreibt die Konstruktion des Hidden States $\mathbf{h}_t$ in einem bidirektionalen RNN. Dabei setzt der neue Hidden State sich aus zwei Teilen zusammen. Dem Vorwärts-Hidden-State $\overrightarrow{\mathbf{h}}_t$, der die Informationen aus der Vergangenheit bis zum Zeitpunkt t enthält, und dem Rückwärts-Hidden-State $\overleftarrow{\mathbf{h}}_t$, der die Informationen aus der Zukunft bis t einbezieht. Diese beiden Hidden States werden zusammengefügt, sodass der gewonnene Hidden State Informationen über das gesamte Musikstück enthält.

$$\mathbf{h}_t = \left[\overrightarrow{\mathbf{h}}_t \,;\, \overleftarrow{\mathbf{h}}_t\right]$$

- $\overrightarrow{\mathbf{h}}_t$: Vorwärts-Hidden-State.
- $\overleftarrow{\mathbf{h}}_t$: Rückwärts-Hidden-State.

- $[\cdot\ ;\ \cdot]$: Zusammenfügung der beiden Hidden States.
- $\mathbf{h}_t$: Vollständiger Hidden State zum Zeitpunkt t.

Auch in AMT-Systemen sind bidirektionale RNNs sehr hilfreich, da musikalische Ereignisse auch eine klar verständliche Abhängigkeit von der Zukunft in die Vergangenheit haben. Das gleiche Prinzip lässt sich auch auf LSTMs oder GRUs anwenden, sodass beispielsweise ein bidirektionales LSTM entsteht. Bidirektionale RNNs sind noch robuster, aber dafür ist der Rechenaufwand bei weitem höher. Ein bidirektionales RNN findet sich auch in folgender Arbeit [Hawthorne u. a. 2017].

4.2.5 Geschichte und Weiterentwicklung von RNNs und ihren Varianten

Die Geschichte von Recurrent neural networks reicht bis in das Jahr 1990 zurück [Elman 1990]. In dieser Arbeit wurde erstmals die Idee der Rückkopplung eingebaut, wodurch der Output eines Neurons als zusätzliche Eingabe im nächsten Zeitfenster genutzt wird. Das resultierte später in den Hidden States. Diese Arbeit baute den Grundstein für alle weiterführenden RNNs. Ein weiterer früher Ansatz wurde bereits 1986 als technischer Bericht vorgestellt und 1997 publiziert [Jordan 1997]. Er führte sogenannte Kontexteinheiten ein, bei denen der Output eines Neurons in das folgende Zeitfenster rückgekoppelt wird. Damit wurde ein alternativer Mechanismus zur Sequenzmodellierung vorgeschlagen, der sich von Elmans Rückkopplung aus dem Hidden Layer unterschied und wichtige Grundlagen für spätere Varianten von RNNs schuf. Zudem wurde, im Jahre 1997, das erste Bidirektionale RNN [Schuster und Paliwal 1997] und das erste LSTM erfunden [Hochreiter und Schmidhuber 1997]. Erst über ein Jahrzehnt später wurde das erste GRU entwickelt [Chung u. a. 2014]. RNNs fanden ihren Weg in die automatische Musiktranskription, fast zeitgleich zu CNNs, im Jahre 2015 [Sigtia, Benetos, Boulanger-Lewandowski u. a. 2015]. Für LSTMs dauerte die Einbindung in AMT-Systeme etwas länger. Erst im Jahre 2016 wurde ein LSTM in einem AMT-System eingefügt [Sigtia, Benetos und Dixon 2016]. Dahingegen wurde das erste Bidirektionale RNN erst Ende 2017 in ein AMT-System integriert [Hawthorne u. a. 2017]. Noch ein Jahr später wurde das erste Mal auch ein GRU in einem AMT-System genutzt [Jung, H. Lee und Tani 2018].

4.3 Transformer

Transformer sind eine, von Google entwickelte Deep-Learning Architektur [Vaswani u. a. 2017], die mit dem Prinzip der Self-Attention, zur natürliche Sprachverarbeitung genutzt wird. Auch Transformer verarbeiten die Daten, wie ein LSTM, sequentiell. Jedoch wurde zu dieser Verarbeitung noch der Attentionmechanismus hinzugefügt, welcher der grundlegende Baustein eines Transformer-Modells ist. Die Architektur eines Transformers besteht grundlegend aus drei Bausteinen. Diese sind Input Embedding mit Position Encoding, Self-Attention Layer und Feedforward Layer. Input Embedding und Position Encoding werden nur einmalig am Anfang des Transformers durchgeführt. Dahingegen werden Self-Attention und Feedforward mehrmals hintereinander aufgerufen. Dies passiert grundsätzlich 6-, 12- oder 24-mal, wodurch das Transformer-Modell an Layern gewinnt. Dieser Schritt wird so oft wiederholt, damit der Transformer immer komplexere musikalische Strukturen erkennen kann. Als Nächstes werden die einzelnen Schritte eines Transformers näher erläutert.

4.3.1 Input Embedding und Position Encoding

Zunächst müssen die Daten, welche der Transformer verarbeiten soll in das richtige Format für diesen umgewandelt werden. Dafür ist das Input Embedding zuständig. Tokens sind einzelne Datenpunkte wie beispielsweise kurze Wörter wie „hallo“. Wenn zum Beispiel bei ChatGPT die Anfrage „Wie alt sind die Planeten unseres Sonnensystems“ gestellt wird, dann wird zwar das Wort „alt“ als ein Token gespeichert, aber längere Wörter wie „Sonnensystem“ werden aufgeteilt in „Sonn“ und „ensystem“. Je nach Tokenizer-Version, die der jeweilige Transformer nutzt, kann dies jedoch abweichen. Bei diesem Beispielsatz werden, alleine für die Wörter, neun Tokens genutzt. Tokens können auch einzelne Satzzeichen oder Symbole sein. Für verschiedene Transformer Modelle gibt es immer eine maximale Tokenanzahl, bei GPT-3.5 sind es zum Beispiel ungefähr 4000 Tokens. Diese Tokens sind für den In-

put und Output. Heißt, wenn der Input zu viele Tokens nutzt, gibt es weniger Tokens für den Output. Dies ist jedoch häufig, wie an der Menge der Tokens für unseren Beispielsatz zu sehen ist, kein größeres Problem. In AMT-Systemen stellen Tokens Zeitfenster (Frames) oder Musik-Events (Onsets etc.) dar. Jedoch können diese von einem Transformer nicht verarbeitet werden, weshalb sie durch Input Embedding erstmals in Vektoren umgewandelt werden. Dies passiert über eine Embedding-Matrix, welche die einzelnen Tokens in den Vektorraum einbettet.

$$\text{Embedding-Matrix} \in \mathbb{R}^{(\text{Maximale Tokenanzahl} \times \text{Embedding-Dimension})}$$

Tokens, welche ähnliche musikalische Eigenschaften speichern, werden im Vektorraum näher beieinander gespeichert. Das Modell kann die Tokens, durch die gesammelten Daten im Training, richtig einordnen.

Nun sieht der Transformer die Tokens trotzdem nur als zahlreiche Vektoren, ohne Rheinfolge, an. Um den Tokens einen Sinn zu geben, müssen diese eine explizite Position erhalten. Das wird durch Position Encoding gemacht. Durch Sinus- und Cosinunsfunktionen wird für jeden Token ein eindeutiger Positionsvektor berechnet. Dies geschieht, indem für jede Dimension, die unsere Tokens besitzen, eine Funktion mit unterschiedlicher Frequenz gebildet wird. Bei geraden Dimensionen wird eine Sinusfunktion genutzt und bei ungeraden eine Cosinusfunktion. So erhält jede Position einen eindeutigen Positionsvektor, während nahe Positionen ähnliche Encodings und somit geringeren Abstand zueinander besitzen. Dadurch kann der Transformer die relative Position, verschiedener Tokens, gut miteinander vergleichen und zugleich wiederkehrende Muster erkennen. Danach wird der Positionsvektor dem Tokenvektor aufaddiert. Moderne Transformer besitzen manchmal auch gelernte Position Embeddings, wodurch die Positionen schon durch das Training bekannt sind. Diese Vektoren werden, als Input, weiter an den Transformer geleitet.

4.3.2 Self-Attention Layer

Jeder Input-Vektor wird einzeln behandelt. Durch Self-Attention kann sich jeder Vektor merken, welche anderen Vektoren für ihn relevant sind. Zunächst wird jeder Input-Vektor in drei verschiedene Vektoren umgewandelt. Diese Vektoren heißen „Query“, „Key“ und „Value“.

- **Query** (Q): Bestimmt, auf welche Informationen aus anderen Tokens das Modell aktuell achten möchte.
- **Key** (K): Zeigt anderen Tokens, was dieser Input-Vektor anderen Tokens, für Informationen, anbieten kann.
- **Value** (V): Enthält die tatsächlichen Informationen des Input-Vektors.

Als Nächstes wird der „Attention Score“ berechnet. Durch diesen wird die Kompatibilität zu anderen Tokens berechnet. Mit Kompatibilität ist dabei die Ähnlichkeit zwischen einem Query- und einem Key-Vektor gemeint. Diese gibt an, wie stark ein Token im Verhältnis zu einem anderen berücksichtigt werden soll. Als Ergebnis entsteht für jeden Token eine Matrix. Jeder Wert in dieser Matrix steht für die Ähnlichkeit zwischen zwei verschiedenen Tokens. Diese Werte sind die Attention Weights. Durch Softmax werden diese Werte normalisiert. Als Letztes werden diese Attention Weights mit den Value-Vektoren verbunden.

$$\text{Output} = \text{Attention Weights} \times \text{Value Vektoren}$$

Nun können die einzelnen Tokens sich besser auf die Tokens fokussieren, welche wirklich wichtig für deren Ergebnis sind.

4.3.3 Feedforward Layer

Der Feedforward Layer stammt von der Idee eines „Feedforward Neural Networks“ (FNN). Dieses neuronale Netz verläuft immer nur in eine Richtung und hat keine Rückkopplung. Das heißt jeder Token wird für sich alleine, nacheinander, umgeschrieben. Das funktioniert, da die jeweilige Gewichtung schon im Self-Attention Layer stattgefunden hat. Im Feedforward Layer passiert dann folgendes.

Zunächst werden die Vektoren, welche aus dem Self-Attention Layer stammen, mit einer Gewichtsmatrix auf eine höhere Dimension transformiert. Durch die Erhöhung der Dimension bekommt das Netzwerk mehr Freiraum Informationen voneinander zu trennen und komplexere Zusammenhänge zu modellieren. Durch eine Aktivierungsfunktion, zum Beispiel ReLU, erlernt das Modell jetzt nichtlineare Abhängigkeiten. Diese Aktivierungsfunktion gehört zu dem Hidden Layer des FNN. Ein FNN kann mehrere Hidden Layer besitzen, welche alle die Input-Werte in irgendeiner Form verändern. Als Nächstes wird der Vektor wieder auf seine ursprüngliche Dimension zurückprojiziert, wodurch nur die wichtigsten Informationen erhalten bleiben. Mathematisch sieht der Feedforward Layer folgendermaßen aus:

$$y_t = \text{FFN}(x_t) = (x_t W_1 + b_1) \xrightarrow{\text{ReLU}} W_2 + b_2$$

- x_t: Eingabevektor des Tokens.
- W_1, W_2: Gewichtsmatrizen, zuständig für Dimensionserweiterung und -reduktion.
- b_1, b_2: Bias-Vektoren der jeweiligen linearen Projektionen.
- $(x_t W_1 + b_1)$: Ergebnis der ersten linearen Projektion.
- ReLU: Aktivierungsfunktion zur Einführung von Nichtlinearität.
- y_t: Ausgabewert des Feedforward Layers für das Token.

4.3.4 Geschichtliche Einordnung von Transformern

Das erste Transformer-Modell wurde im Jahre 2017 erfunden [Vaswani u. a. 2017]. Durch den Self-Attention Layer wurde die sequenzielle Modellierung von Daten revolutioniert. Es dauerte noch lange bis der erste Transformer auch in AMT-Systemen genutzt wurde. Das liegt an mehreren Gründen. Einerseits gibt es im Gegensatz zu natürlichen Sprachverarbeitung weniger Datensätze, von denen die Modelle lernen könnten. Transformer trainieren auf einem globalen Level und brauchen daher eine große Menge an Datensätzen. Die Rechenkosten von Transformer Modellen sind auch weitaus höher als bei anderen Systemen, wie CNNs und RNNs. Zudem lag der Fokus bei AMT-Systemen für eine sehr lange Zeit überwiegend bei CNN und RNN basierenden Systemen, da diese Anwendungsfälle bekannter waren in AMT-Systemen. Das größte Problem war jedoch wahrscheinlich die Anpassung. Transformer Modelle waren einfach nicht dafür ausgelegt musikspezifische Daten zu verarbeiten. In der Musiktranskription wird immer mit langen Sequenzen, ein gesamtes Musikstück, gearbeitet. Spezielle Transformer Modelle für diesen Anwendungsfall mussten noch programmiert werden. Die ersten Ansätze für Transformer in AMT-Systemen wurden zwischen den Jahren 2021 und 2023 erstellt. Eines der ausschlaggebendsten Modelle war der Music Transcription Transformer MT3, welcher vom Magenta-Team bei Google Brain erstmals im Jahre 2022 veröffentlicht wurde [Gardner u. a. 2021]. Dieses spielt eine große Rolle in der Transformer-basierten automatischen Musiktranskription, da es das erste weit verbreitete Multi-Task-Transkriptionsmodell ist. Eine der Letzten bedeutenden Errungenschaften zu Transformern und Musiktranskription ist das YourMT3+ Modell, indem die MT3-Architektur noch weiter verbessert wurde [Chang u. a. 2024].

4.4 Potentielle KI-Modelle

Bei der KI integration in AMT-Systemen werden überwiegend CNNs und RNNs oder Transformer Modelle wie MT3 verwendet. Jedoch gibt es noch andere KI-Modelle, die in AMT-Systemen integriert werden können. Diese gehen, bei der Musiktranskription, ganz anders vor als die vorgestellten Module. Je nach KI übernehmen sie auch eine ganz andere Aufgabe. Im folgenden Abschnitt werden einige von diesen, eher experimentellen, KIs vorgestellt. Wir fangen dabei bei der am meisten erprobten KI an, sodass die letzten KIs nur theoretisch, für die Musiktranskription, besprochen worden.

Variational Autoencoder

Variational Autoencoder (VAE) ist ein KI-Modell, welches komplexe Daten in eine verdichtete Form überführt und durch Wahrscheinlichkeiten bestimmte Muster vorhersagen kann [Kingma, Welling u. a. 2019]. Dabei ist wichtig zu erwähnen das VAE kein alleinstehendes KI-Modell ist, sondern eher als Zusatz gilt um bestimmte Bereiche zu verbessern. In der Musiktranskription könnte ein VAE zum Beispiel bei der Mehrdeutigkeit von Musik helfen. Wenn mehrere Instrumente spielen ist es schwer die genauen Noten herauszuschreiben. Da VAEs, anders als CNNs oder RNNs, als Ausgabe keine eindeutigen Noten herausgeben, sondern eine Wahrscheinlichkeit, könnte dies helfen die Entscheidung, welche Note gerade spielt, robuster zu gestalten. Dadurch, dass VAEs nur eine wahrscheinliche Version der Musik aus den Daten extrahieren, könnten damit auch kreative Variationen des Musikstückes gebildet werden. Die Methode, welche in VAEs genutzt wird, fand Ihren Ursprung in folgendem Paper [Kingma, Welling u. a. 2013].

Graph Neural Network

Graph Neural Network (GNN) ist ein KI-Modell, welches dazu dient Daten, mit Abhängigkeit voneinander, zu verarbeiten. Diese Daten werden in einem Graphen aus Knoten und Kanten gespeichert. Dabei hat jeder Knoten seine eigenen Daten, die er in jedem Rechenschritt mit seinen benachbarten Knoten austauscht. Knoten sind benachbart, wenn sie durch Kanten verbunden sind. Dadurch wird jeder Knoten im Netzwerk mit mehr Kontext ausgestattet. Natürlich machen RNNs und Transformer etwas Ähnliches wie ein GNN, mit zum Beispiel Backpropagation. Der Unterschied ist hier, das GNNs keine bestimmte Reihenfolge, wie Wissen weitergeleitet wird, haben. Es ist ein anderer Ansatz harmonische Abhängigkeiten miteinander zu kombinieren [Z. Wu u. a. 2020].

Diffusion Model

Diffusion Modelle arbeiten mit Rauschen. Aus komplettem Rauschen bauen sie Schritt für Schritt ein Musikstück zusammen. Dabei entscheiden sie pro Rechenschritt nur einen kleinen Teil des Musikstückes, wodurch mehrdeutige Passagen in Musikstücken auch noch im späteren Transkriptionsprozess behandelt werden können. In dem Projekt DiffRoll, aus dem Jahre 2022, wurde ein Diffusion Model auf ein AMT-System angewendet [K. W. Cheuk, Sawata u. a. 2023].

Reinforcement Learning

Reinforcement Learning benutzen ein Belohnungssystem, damit der Agent sich beim Training richtig anpasst. Sobald der Agent ein bestimmtes Ziel erreicht oder eine Handlung ausführt, wird dieser dafür belohnt. Die Agenten, welche am meisten Belohnungen erzielen, werden kopiert und in der nächsten Iteration eingesetzt, bis ein Agent die gewünschte Aufgabe erfüllt. Dieses Prinzip wird vor allem in Videospielen eingesetzt, wo es grundlegend eindeutige Ziele gibt. In der Musiktranskription könnte dieses Prinzip folgendermaßen umgesetzt werden: Der Agent erhält als Input eine Audiodatei. Danach erzeugt er Noten nacheinander. Durch eine andere KI oder vorgefertigte MIDI-Dateien steht ein Vergleichsdatensatz zur Verfügung. Falls die gewählten Noten vom Agenten gleich dem Vergleichsdatensatz sind, bekommt der Agent Belohnungen. So kann er sich selber musikalisches Wissen aneignen und Transkriptionen von anderen KIs nochmal korrekturlesen [Li u. a. 2018].

Energy-Based Model

Der letzte Ansatz ist ein Energy-Based Model (EBM). Ein EBM arbeitet mit Energie anstatt von Wahrscheinlichkeiten [LeCun, Chopra u. a. 2006]. Die verschiedenen Arten, wie das EBM das Musikstück transkribieren kann, haben alle jeweils ihre eigene Energie. Dabei haben die besten Möglichkeiten immer die niedrigste Energie, wie bei dem Gradientenabstiegsverfahren. Dies ist für AMT-Systeme interessant, da das EBM somit zahlreiche verschiedene Transkriptionsraten wählen könnte. In mehr improvisierten Musikrichtungen, wie Jazz, bildet das eine größere Vielfalt. Auch Musiktheorie lässt sich in das Modell einbauen, wodurch aus einem Musikstück Unmengen an Remixen kreiert

werden können. Der Ansatz von EBMs in AMT-Systemen ist, von den zusätzlich vorgestellten KI-Modellen, wahrscheinlich einer der Interessantesten. Jedoch sind EBMs sowohl schwer zu trainieren, als auch sehr rechenintensiv. AMT-Systeme sind ohnehin schon recht anspruchsvoll, weshalb noch kein EBM-Modell in der automatischen Musiktranskription zum Einsatz kam.

5 KI-basierende AMT-Systeme im Vergleich

Im Laufe der Forschung zu AMT-Systemen wurden schon einige verschiedenen Architekturen eingesetzt und verbessert. Jedes neue System hat, unabhängig des KI-Modells, eine komplett eigene Struktur und Vorgehensweise. Im Verlauf dieser wissenschaftlichen Arbeit wurde die Geschichte von AMT-Systemen behandelt und die wichtigsten Konzepte der automatischen Musiktranskription dargestellt. Um dieses Forschungsgebiet zurück in die jetzige Gegenwart zu bringen, handelt das letzte Kapitel von den heutigen „State of the Art“ AMT-Systemen. Dafür werden zwei verschiedene AMT-Systeme jetzt vorgestellt. Diese sind das CNN + GRU basierte Omnizart und das Transformer-basierte MT3-Modell. Diese nutzen unterschiedliche KI-Modelle. Deren Architektur, sowie deren Stärken und schwächen, werden in dem folgenden Kapiteln ausgiebig erläutert.

5.1 Omnizart

Das erste System ist Omnizart, welches CNNs und GRUs als KI-Modelle nutzt [Y.-T. Wu u. a. 2021]. Der Name Omnizart setzt sich aus den Wörtern „Omni“ (alles) und „Mozart“ zusammen, da ihr Ziel darin liegt, so viele Arten von Musik wie möglich zu transkribieren. Je nach Anwendungsfall werden verschiedene KI-Systeme genutzt. Dabei folgt der Aufbau dieser KI-Systeme meistens dem gleichen Schema. Alle KI-Modelle bestehen aus einem CNN und einem bidirektionalen GRU. Der Anwendungsfall, zum Beispiel Drums oder Melody, hat dabei nur Einfluss auf die Trainingsdaten und dem Output. Omnizart ist ein Open-Source-Toolkit für AMT. Dadurch lässt sich je nach Bedarf ein KI-Modell auswählen, das perfekt auf eine bestimmte Aufgabe ausgelegt wurde. Omnizart findet seinen Ursprung im Jahre 2020 am Music and Culture Technology Lab, National Taiwan University. Omnizart hat seit seiner Gründung keine ausschlaggebenden weiteren Technologien, in der Richtung KI, hinzugefügt. Dafür gibt es für jeden vertretenen Anwendungsfall verschiedene CNNs und GRUs und einen open source code, welcher gut zur eigenen Forschung an AMT-Systemen genutzt werden kann. Omnizart ist ein modulares AMT-System, welches auf einer Deep-Learning-Architektur basiert. Der Trainingssatz besteht aus CQT-Spektrogrammen. Die KI-Modelle lernen dabei durch Supervised Learning.

Die Pipeline von Omnizart lässt sich in drei Schritte aufteilen:

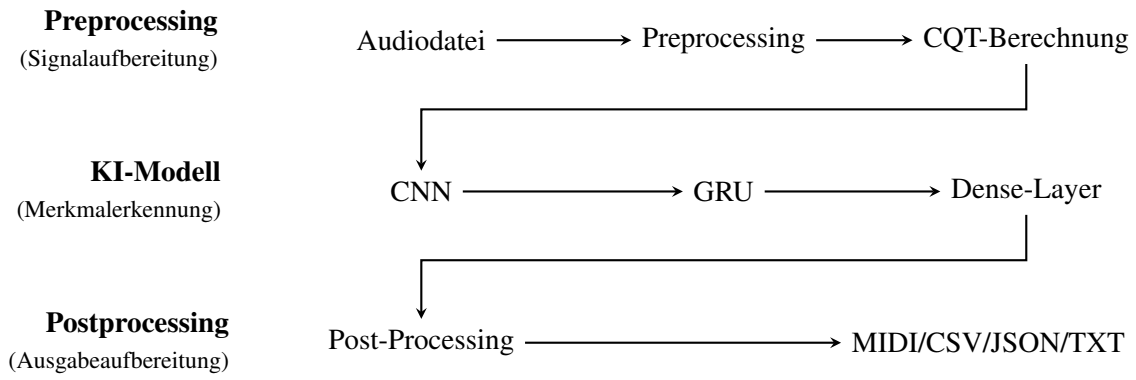

Abbildung 10: Eigene Darstellung der Omnizart Pipeline

Der erste Schritt ist die Vorverarbeitung und Merkmalsextraktion (Preprocessing). Dadurch wird die Inputaudiodatei in ein CQT-Spektrogramm umgewandelt. Zunächst wird das gegebene Audiosignal durch folgende Methoden standardisiert:

1. **Mono-Konvertierung:** Die KI benötigt keine räumlichen Informationen, weshalb der linke und rechte Kanal von Stereosignalen in ein Monosignal addiert werden.
2. **Normalisierung:** Der Wechsel von zu großen und kleinen Amplituden kann die KI überfordern und ungenaue Ergebnisse liefern, weshalb das Audiosignal durch Normalisierung auf einen einheitlichen Lautstärkebereich gebracht wird.
3. **Resampling:** Unterschiedliche Abtastraten führen zu Verzerrung und Frequenzverschiebung, deshalb wird diese auf eine einheitliche Rate, passend zu dem genutzten Modul, gebracht.
4. **Trimming:** Falls am Anfang oder Ende des Audiosignals Stille ist, wird diese durch Trimming entfernt, sodass das KI-Modell nicht unnötig verwirrt wird.

Danach wird das standardisierte Audiosignal umgeformt zu einem CQT-Spektrogramm.

Im zweiten Schritt werden die KI-Modelle genutzt, um die Merkmale des Audiosignals vorherzusagen und zu extrahieren. Die meisten AMT-Systeme, welche im Laufe dieser Arbeit vermerkt wurden, besitzen eine ähnliche Architektur wie Omnizart [Hawthorne u. a. 2017]. Der Unterschied zu diesen AMT-Systemen ist das Omnizart, je nach Anwendungsfall, verschiedene Module nutzt. Omnizarts Module sind:

- **Chord:** Akkorderkennung
- **Drum:** Drum-Transkription
- **Melody:** Melodietranskription
- **Vocal:** Gesangsmelodietranskription
- **Piano:** Polyphone Klaviertranskription
- **Multi-Pitch:** Mehrstimmige Tonhöhenschätzung
- **Beat/Downbeat/Chord-Labelling:** Rhythmus, Takt & Akkorde

Jedes Modul bekommt als Input ein CQT-Spektrogramm. Dieses wird durch ein CNN verarbeitet, welches die Eigenschaften und Merkmale der Musik extrahiert. Mit diesen Daten modelliert dann ein GRU die zeitliche Abhängigkeit. Durch den Dense-Layer werden die Ergebnisse des GRUs in zum Beispiel Onsets, Beats und zahlreiche weitere musikalischen Attribute, umgewandelt.

Im dritten Schritt wird der Output der KI-Modelle nochmals aufbereitet. Fehler werden zunächst durch folgende Methoden verbessert:

1. **Noten-Segmentierung:** Wenn derselbe Ton in zwei nacheinander folgenden Frame ein Onset hat, wird der zweite Onset gelöscht und der Note hinzugefügt, sodass keine Notendopplungen entstehen.
2. **Onset-Korrektur:** Falls ein Onset zeitlich nicht zur richtigen Zeit erfasst wurde, wird der Einschwingzeitpunkt des Onset an seinen Frame genau angepasst.
3. **Thresholding:** Noten, die nicht einen bestimmten Wahrscheinlichkeits-Threshold überschreiten, werden aussortiert.
4. **Quantisierung:** Je nach Taktstruktur können Noten zeitlich angepasst werden, sodass diese besser in beispielsweise einen 3/4-Takt passen und das Stück somit rhythmischer ist.

Je nach Modul, welches gerade genutzt wird, werden nur ein paar oder alle dieser Methoden eingesetzt. Auch deren Parameter unterscheiden sich je nach Modul. So hat das Drums-Modul ein viel kleineren Schwellwert als das Melodien-Modul bei Thresholding. Danach werden die Vorhersagen

in vollständige Noten (Onset, Sustain, etc.) zusammengefasst und in das gewünschte Format übertragen. MIDI ist dabei das wichtigste Format, da dieses die relevanten Daten der Transkription für verschiedene Musiksoftware besitzt. Es gibt aber auch noch drei andere Formate die, je nach Modul, ausgegeben werden. Fast immer wird auch eine CSV-Datei ausgegeben. Darin befinden sich die Transkriptionsdaten, welche zur Analyse oder Forschung genutzt werden können. JSON-Dateien werden in den Chord- und Beat-Modulen ausgegeben. Dies liegt daran, dass JSON-Dateien verschachteltet Daten, wie Akkordfolgen über mehrere Takte, besser speichern können. In ihnen werden vor allem Takt-Informationen und Daten für Akkorde gespeichert. TXT-Dateien werden dahingegen nur wahlweise in Modulen genutzt. Sie sind ausschließlich für Debugging dar, weshalb sie für die meisten Nutzer nicht relevant sind. Die Daten werden, in einer TXT-Datei, in einer unstrukturierten Liste ausgegeben.

5.2 MT3

Das Transformer-Modell, welches jetzt näher erläutert wird, heißt MT3. MT3 steht für „Multi-Task Multitrack Music Transcription“. MT3 ist gleichzeitig der Name für das Transformer-basierte KI-Modell, als auch für das AMT-System, indem dieses KI-Modell veröffentlicht wurde [Gardner u. a. 2021]. Diese beiden sind untrennbar voneinander, da sie zu einer End-to-End-Architektur gehören. Das MT3-Modell bildet einen wichtigen Meilenstein für die Transformer-basierte automatische Musiktranskription. MT3 ist das erste weit verbreitete Multi-Task-Transkriptionsmodell. Das heißt verschiedene Transkriptionsaufgaben werden von einem einzigen Modell gelöst. Frühere Modelle brauchten zum Beispiel für Drums und die Melody, wie es bei Omnizart der Fall ist, verschiedene KI-Modelle. Durch die Communityversion „YourMT3+“ wird dieses KI-Modell immer weiter gepflegt und verbessert.

MT3 ist ein Transformer-Modell, welches spezifisch für Musiktranskription entwickelt wurde. Es besteht grundlegend aus folgenden Bereichen:

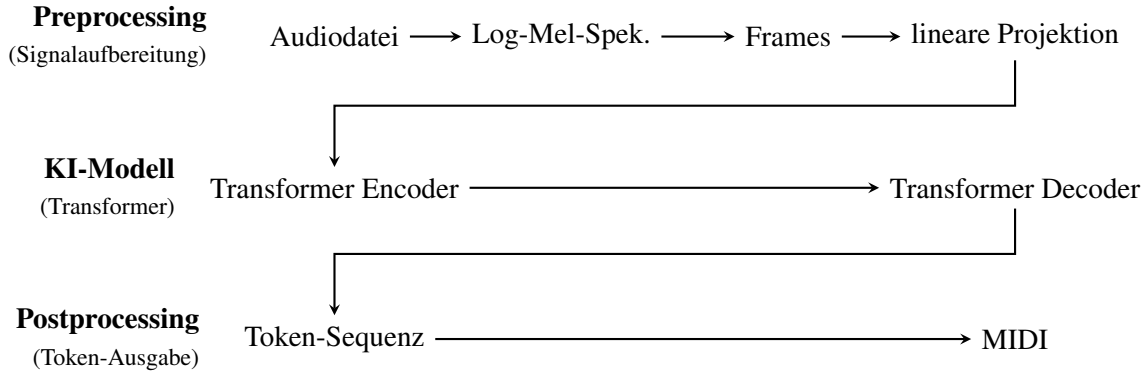

Abbildung 11: Eigene Darstellung der MT3 Pipeline

Als Input wird eine Audiodatei bereitgestellt. Meistens wird der Datentyp WAV (Waveform Audio File Format) genutzt, da dieser verlustfrei und standardisiert ist. Zudem wurde MT3 mit WAV-Dateien trainiert, weshalb es wenig Sinn macht einen anderen Datentyp zu nutzen. Das Audiosignal wird dann mit STFT analysiert und als Spektrogramm ausgegeben. Mithilfe von Librosa wird aus diesem ein Log-Mel-Spektrogramm erzeugt. Librosa ist ein Python Paket, welches wichtige Methoden für Musik und Audio Analyse bereitstellt. Dieses Log-Mel-Spektrogramm wird nun in Frames aufgeteilt. In dem MT3-Modell sind diese Frames meist 32ms lang. Jeder Frame besitzt NxMel-Frequenzbänder, welche die Dimensionsgröße dieses Frames bestimmen. Durch die Anzahl der Mel-Frequenzbänder wird die Frequenzauflösung des Inputs bestimmt. In dem MT3-Modell werden standardmäßig 128 Mel-Frequenzbänder pro Fram genutzt. Um jetzt für jeden Frame eine einheitliche Dimension zu bekommen, mit der die KI arbeiten kann, wird lineare Projektion genutzt. Lineare Projektion ist ein

Matrix-Multiplikator, welche ein Frame auf eine, für das KI-Modell, normalisierte Dimension bringt, zum Beispiel 512 oder 1024. Daraus entstehen Input Embeddings, mit denen die KI jetzt arbeiten kann.

Jetzt folgt der Encoder, welcher das normale Prinzip eines Transformers, mit Self-Attention Layer und Feedforward Layer, ausführt. In dem MT3-Modell werden, vor der Ausführung der Layer, die Positionen der Frames durch Positional Embeddings festgelegt. In MT3 besteht der Encoder aus 6 vollständigen Transformer-Layern. Der Inhalt jedes Frames wird jetzt mit dem aller anderen kontextabhängigen Frames zusammengeführt. Daraus resultieren Vektoren, welche kontextreiche Informationen des Musikstückes besitzen. Diese Vektoren werden weiter an den Decoder geleitet.

Aus den gegebenen Vektoren extrahiert der Decoder die wichtigen Daten. Neben den Encoder Vektoren als Input bekommt dieser zudem autoregressive Tokens. Autoregressive Tokens sind bereits generierte Tokens, die im Training meist aus den echten Daten stammen (Teacher Forcing) und beim Einsatz vom Modell, mithilfe der bereits generierten Tokens, selbst generiert werden. Dadurch bekommt der Decoder mehr Kontext für die zu generierenden Tokens. Neben den normalen Tokens bekommen auch die autoregressiven Tokens Positional Embeddings. Bevor der Decoder nun durchläuft, lässt sich mithilfe von Task-Conditioning ein Task-Token hinzufügen. Das MT3-Modell transkribiert immer polyphon, jedoch kann durch den Task-Token trotzdem der Output auf einen bestimmten Task, wie zum Beispiel Klavierstücke oder Trommelnoten, ausgelegt werden. Durch vorheriges Lernen ist das MT3-Modell darauf ausgelegt, nach einem Task-Token die folgenden Tokens in einer bestimmten Art und Weise zu transkribieren. Der Ablauf des Decoders sieht folgendermaßen aus:

1. **Self-Attention:** verarbeitet den Kontext der autoregressiven Tokens
2. **Cross-Attention:** extrahiert relevante Informationen aus dem Encoder-Output
3. **Feedforward Layer:** verfeinert die Repräsentationen der Tokens lokal
4. **Lineare Layer:** wandelt den Vektor in ein konkretes Token um

Dabei besteht der Decoder, im MT3-Modell, auch aus 6 vollständigen Transformer-Layern. Jeder Token stellt einen Teil eines musikalischen Events dar. Diese heißen zum Beispiel „Note-On C4“ oder „Shift +10 ms“. Ein „Shift“ von beispielsweise 10ms bedeutet, dass das nächste Ereignis 10 Millisekunden nach vorne in der Zeitleiste verschoben wird. Ein musikalisches Event ist eine volle Note, mit Notennamen, Onset und Offset, Sustain und weiteren musikalischen Eigenschaften, die auf einer Note angewendet werden können. Als Output gibt der Decoder jeden einzelnen Token zurück.

Jetzt werden im Post-Processing die Tokens zu Sequenzen (musikalischen Events) zusammengesetzt. Durch einige weitere Tools werden die Noten zudem noch bereinigt und verbessert.

- **Zeitberechnung:** Die Shift-Tokens werden aufsummiert, sodass alle Noten zur richtigen Zeit abgespielt werden.
- **Noten validierung:** Jede Note wird darauf geprüft, ob sie ein On- und Offset besitzt.
- **Velocity Korrektur:** Manche Noten haben keine oder eine fehlerhafte Anschlagstärke, welche bei diesem Schritt im Nachhinein hinzugefügt oder überschrieben wird.
- **Fehlerbereinigung:** Unlogische Notenfolgen, wie zwei Shifts hintereinander, werden entfernt.

Am Ende werden die bereinigten Noten in einem standardisierten Musikformat, als MIDI-Datei, ausgegeben.

In der folgenden Darstellung ist der Transkriptionsablauf des MT3-Modells dargestellt. Für den Input wird ein Audiosignal angegeben, welches in ein Spektrogramm umgewandelt wird. Als nächstes gewichtet der Encoder, mithilfe der Self-Attention Layer, die Tokens und transformiert dann diese, mithilfe des Feedforward Layer, um nicht-lineare Merkmale pro Token zu extrahieren. Daraufhin wandelt der Decoder, die vom Encoder gegebenen Vektoren, in einzelne Tokens mit musikalischen

Merkmalen um. Im Output ist oben ein Ausschnitt einer MIDI-Datei mit zwei Noten zu sehen und darunter diese dargestellt als Piano-Roll.

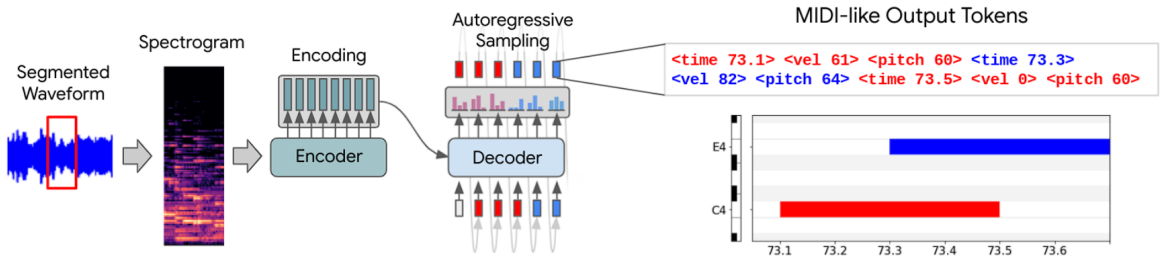

Abbildung 12: Verarbeitungspipeline des Transformer-basierenden MT3-Modells [Gardner u. a. 2022].

5.3 Bewertung der AMT-Systeme im Vergleich

Die beiden vorgestellten AMT-Systeme sind grundlegend verschieden aufgebaut. Jetzt ist die Frage, welches dieser beiden Systeme besser geeignet ist. Bei welchem Anwendungsfall und unter welchen Voraussetzungen sollte man welches System wählen?

Der größte Unterschied in der Architektur liegt darin, das Omnizart verschiedenen KI-Modelle für unterschiedliche Aufgaben besitzt, während MT3 ein einziges leistungsstarkes KI-Modell besitzt. Somit lässt sich bei Omnizart leichter ein bestimmter Task verbessern oder analysieren. Das ist vor allem gut, wenn der Fokus auf monophonen Musikstücken liegt und ein einfacherer Einstieg in die automatische Musiktranskription erreicht werden soll. Zudem gibt Omnizart eine größere Menge an Outputdaten zurück als MT3, bei dem nur eine MIDI-Datei ausgegeben wird. MT3 hingegen besitzt ein einziges KI-Modell, wodurch kein spezieller Task gezielt umstrukturiert werden kann. Dafür eignet sich das KI-Modell direkt das Wissen der verschiedenen Tasks an und kombiniert diese. Somit können polyphone Musikstücke deutlich besser transkribiert werden. Durch YourMT3+ gibt es zudem weiteres Ausbaupotential. Dahingegen entwickelt sich Omnizart nicht sonderlich weiter.

Bei den KI-Modellen hat MT3 einen klaren Vorsprung. CNNs und GRUs sind schon seit längerem in AMT-Systemen vertreten. Sie wurden ausgiebig angepasst und verbessert für diesen Anwendungsfall. Hingegen zu diesen KI-Modellen ist das MT3 erst seit paar Jahren im Rennen. Durch YourMT3+ wird es immer weiter entwickelt, wodurch es in der nahen Zukunft zu einem Standard der automatischen Musiktranskription werden könnte. Zudem ist das MT3-Modell deutlich besser auf polyphone Musikstücke ausgelegt. In der Mehrheit von Audiosignalen gibt es mehrere Stimmen. Die meisten Menschen möchten auch lieber eine einfache Lösung, die wenig Eigenaufwand benötigt. Deshalb wäre ein zentrales KI-Modell, zur Transkribierung von allen verschiedenen Audiosignalen, die populärste Lösung. Ein Schwachpunkt des MT3-Modells ist der Rechenaufwand. Alle KI Prozesse werden durch ein KI-Modell gelöst. Deshalb muss dieses umso mehr Rechenschritte durchführen und braucht exponentiell mehr GPU Auslastung und Speicherkapazität.

Omnizart ist empfehlenswert, falls gerade ein neuer Einstieg in das Forschungsgebiet erfolgt. Fortschritte werden deutlich schneller sichtbar, und der Fokus kann zunächst auf kleinere KI-Modelle gelegt werden. In den meisten anderen Fällen wäre jedoch das MT3-Modell oder das Nachfolgermodell YourMT3+ empfehlenswerter. Dieses ist der „State of the Art“ und wird auch noch in einigen Jahren Support erhalten. Zudem bestehen bei diesem Modell keinerlei Einschränkungen, und jegliche Musik kann transkribiert werden. Dieses Modell geht jedoch auch mit viel Rechenaufwand und einem guten Verständnis des Forschungsgebiets einher. Deshalb sollte vor der Nutzung des MT3-Modells eine gründliche Auseinandersetzung damit erfolgen.

6 Fazit

Automatische Musiktranskription entwickelt sich stetig weiter. Besonders in den letzten Jahren hat dieses Forschungsgebiet erhebliche Fortschritte gemacht. Verantwortlich dafür ist vor allem die rasan-

te Entwicklung Künstlicher Intelligenz. In nur wenigen Jahren wurden unzählige KI-Modelle entwickelt, die im Monatsrhythmus beachtliche Fortschritte zeigten. Fast jedes Problem, was vorher durch Algorithmen gelöst wurde, konnte durch ein schnelleres und besseres KI-Modell ersetzt werden. Dies gilt auch für Algorithmen und Architekturen in AMT-Systemen. CNNs und RNNs entwickelten sich in kürzester Zeit zum neuen Standard innerhalb der AMT-Forschung.

Dieses Phänomen ist jedoch erst der Anfang von Künstlicher Intelligenz und deren Anwendung in AMT-Systemen. So schnell wie CNNs und RNNs die automatische Musiktranskription beeinflusst haben, ebenso schnell etablieren sich bereits neue, verbesserte KI-Modelle. Mit dem Transformer-basierten KI-Modell MT3 wurde erneut ein neuer Stand der Technik erreicht, der die alten KI-Modelle ersetzen sollte. Es ist zu erwarten, dass sich dieser Zyklus noch vielfach wiederholen wird, da wir uns noch am Anfang der KI-Forschung befinden.

Trotz dieser großen Meilensteine gibt es in der automatischen Musiktranskription noch einige offene Probleme, welche gelöst werden müssen. Datensätze sind unzureichend, transkribierte Noten sind fehlerhaft oder unvollständig und rauschende Audioaufnahmen verwirren die KI-Modelle zu stark. KI-Modelle haben zahlreiche Probleme und Fehler beseitigt. Gleichzeitig öffneten sie neue Fehlerquellen, die zuvor nicht existierten. Ein Beispiel dafür ist das Blackbox-Verhalten moderner KI-Modelle. Dadurch wird unklarer, was an dem Datensatz oder direkt in dem KI-Modell verändert werden muss, um ein besseres Ergebnis zu erzielen.

Eine der größten Herausforderungen bleibt jedoch die Überführung in lesbare Notenblätter. Tatsächlich gibt es einige Tools und Firmen die diese Aufgabe mithilfe eines AMT-System schon versuchen zu lösen. Häufig sind die daraus resultierenden Notenblätter jedoch kaum brauchbar. Die Noten werden nicht den richtigen Stimmen zugeordnet und musikalische Regeln werden nicht beachtet. Das führt dazu, dass die Musiknoten nicht intuitiv spielbar sind. Eine Lösung dafür wäre zum Beispiel ein weiteres KI-Modell, das sich mithilfe einer MIDI-Datei systematisch damit auseinandersetzt, wie Musiknoten platziert werden müssen und wann Vorzeichen, Dynamikangaben sowie Artikulationszeichen gesetzt werden.

Im Ganzen ist die Einbindung von Künstlicher Intelligenz ein großer Fortschritt für die automatische Musiktranskription. In naher Zukunft werden AMT-Systeme kontinuierlich weiterentwickelt und in nicht allzu langer Zeit könnten diese auch im Alltag eingesetzt werden.

Literaturverzeichnis

Baum, Leonard E u. a. (1970). „A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains“. In: *The annals of mathematical statistics* 41.1, S. 164–171.

Böck, Sebastian und Markus Schedl (2012). „Polyphonic Piano Note Transcription with Recurrent Neural Networks“. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Zugriff am 19.05.2025, S. 121–124. URL: https://www.cp.jku.at/people/schedl/Research/Publications/pdf/boeck_schedl_icassp_2012.pdf.

Brown, Benjamin, Sheila G. Crewther und David P. Crewther (2016). „Pattern-onset and pattern-offset visual evoked potentials: normative data and clinical applications“. In: *Documenta Ophthalmologica* 133, S. 141–156. DOI: 10.1007/s10633-016-9554-y. URL: https://www.researchgate.net/figure/A-typical-pattern-onset-offset-VEP-Note-that-with-a-300-ms-sweep-only-the-pattern-onset_fig2_305480881.

Chang, Sungkyun u. a. (2024). „YourMT3+: Multi-Instrument Music Transcription with Enhanced Transformer Architectures and Cross-Dataset STEM Augmentation“. In: *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, S. 1–6.

Cheuk, Kin Wai, Yin-Jyun Luo u. a. (2021). „Revisiting the onsets and frames model with additive attention“. In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, S. 1–8.

Cheuk, Kin Wai, Ryosuke Sawata u. a. (Apr. 2023). „DiffRoll: Diffusion-based Generative Music Transcription with Unsupervised Pretraining Capability“. In: *ICASSP*. arXiv preprint arXiv:2210.05148, Oct 2022.

Cheuk, Tom, Kai Shan und Chris Raphael (2020). „The Impact of Audio Input Representations on Neural Network Based Music Transcription“. In: *arXiv preprint arXiv:2001.09989*. URL: https://arxiv.org/abs/2001.09989.

Chung, Junyoung u. a. (2014). „Empirical evaluation of gated recurrent neural networks on sequence modeling“. In: *arXiv preprint arXiv:1412.3555*.

Dave, Smith und Wood Chet (Okt. 1981). „the 'usi', or universal synthesizer interface“. In: *journal of the audio engineering society* 1845.

Elman, Jeffrey L (1990). „Finding structure in time“. In: *Cognitive science* 14.2, S. 179–211.

Eyben, Florian u. a. (2010). „Universal onset detection with bidirectional long-short term memory neural networks“. In.

Fukushima, Kunihiko (1980). „Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position“. In: *Biological cybernetics* 36.4, S. 193–202.

Gardner, Josh u. a. (2021). „MT3: Multi-task multitrack music transcription“. In: *arXiv preprint arXiv:2111.03017*.

– (2022). *Music Transcription with Transformers - MT3 Colab Notebook*. Zugriff am 27.07.2025. URL: https://colab.research.google.com/github/magenta/mt3/blob/main/mt3/colab/music_transcription_with_transformers.ipynb.

Goswami, AP und Makarand Velankar (2013). „Study paper for Timbre identification in sound“. In: *International Journal of Engineering Research & Technology (IJERT)* 2.10.

Graves, Alex, Santiago Fernández und Jürgen Schmidhuber (2007). „Multi-dimensional recurrent neural networks“. In: *International conference on artificial neural networks*. Springer, S. 549–558.

Gu, Xiangming, Longshen Ou u. a. (2024). „Automatic lyric transcription and automatic music transcription from multimodal singing“. In: *ACM Transactions on Multimedia Computing, Communications and Applications* 20.7, S. 1–29.

Gu, Xiangming, Wei Zeng u. a. (2023). „Deep audio-visual singing voice transcription based on self-supervised learning models“. In: *arXiv preprint arXiv:2304.12082*.

Han, Yoonchang, Jaehun Kim und Kyogu Lee (2016). „Deep convolutional neural networks for predominant instrument recognition in polyphonic music“. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.1, S. 208–221.

Hawthorne, Curtis u. a. (2017). „Onsets and frames: Dual-objective piano transcription“. In: *arXiv preprint arXiv:1710.11153*.

Heinemann, Ernst-Günter, Hrsg. (1992). *Das Wohltemperierte Klavier I, BWV 846–869. Nr. 5 in D-Dur*. BWV 850. München.

Hochreiter, Sepp und Jürgen Schmidhuber (1997). „Long short-term memory“. In: *Neural computation* 9.8, S. 1735–1780.

iZotope (n.d.). *What is the Noise Floor?* Zugriff am 19.05.2025. URL: https://www.izotope.com/en/learn/what-is-the-noise-floor.html.

Jamshidi, Fatemeh u. a. (2024). „Machine learning techniques in automatic music transcription: A systematic survey“. In: *arXiv preprint arXiv:2406.15249*.

Jordan, Michael I (1997). „Serial order: A parallel distributed processing approach“. In: *Advances in psychology*. Bd. 121. Elsevier, S. 471–495.

Joysingh, S Johanan, P Vijayalakshmi und T Nagarajan (2019). „Development of large annotated music datasets using HMM based forced Viterbi alignment“. In: *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*. IEEE, S. 1298–1302.

Jung, Minju, Haanvid Lee und Jun Tani (2018). „Adaptive detrending to accelerate convolutional gated recurrent unit training for contextual video recognition“. In: *Neural Networks* 105, S. 356–370.

Kingma, Diederik P, Max Welling u. a. (2013). *Auto-encoding variational bayes*.

– (2019). „An introduction to variational autoencoders“. In: *Foundations and Trends® in Machine Learning* 12.4, S. 307–392.

Klapuri, Anssi und Antti Eronen (1998). „Automatic transcription of music“. In: *Proceedings of the Stockholm Music Acoustics Conference*, S. 6–9.

Kusaka, Yuta und Akira Maezawa (2024). „Mobile-AMT: Real-Time Polyphonic Piano Transcription for In-the-Wild Recordings“. In: *2024 32nd European Signal Processing Conference (EUSIPCO)*. IEEE, S. 36–40.

LeCun, Yann, Bernhard Boser u. a. (1989). „Backpropagation applied to handwritten zip code recognition“. In: *Neural computation* 1.4, S. 541–551.

LeCun, Yann, Sumit Chopra u. a. (2006). „A tutorial on energy-based learning“. In: *Predicting structured data* 1.0.

Li, Juncheng u. a. (2018). „Music theory inspired policy gradient method for piano music transcription“. In: *Advances in Neural Information Processing Systems* 31.

Magenta Team (2017). *Performance RNN: Generating Expressive Piano Performances with Neural Networks*. https://magenta.tensorflow.org/performance-rnn. Accessed: 2025-06-22.

Mankowitz, Daniel J u. a. (2023). „Faster sorting algorithms discovered using deep reinforcement learning“. In: *Nature* 618.7964, S. 257–263.

Marták, Lukáš Samuel, Rainer Kelz und Gerhard Widmer (2022). „Balancing bias and performance in polyphonic piano transcription systems“. In: *Frontiers in Signal Processing* 2, S. 975932.

Martin, Keith D. (1996). *Automatic Transcription of Simple Polyphonic Music: Robust Front End Processing*. Techn. Ber. Technical Report 399. Zugriff am 19.05.2025. MIT Media Lab. URL: https://www.media.mit.edu/publications/automatic-transcription-of-simple-polyphonic-music-robust-front-end-processing-2/.

Moorer, James A. (1977). „On the Transcription of Musical Sound by Computer“. In: *Computer Music Journal* 1.4. Zugriff am 19.05.2025, S. 32–38. URL: https://www.jamminpower.org/PDF/JAM/Transcription%20of%20Musical%20Sound%20-%20CMJ%201977.pdf.

Scheirer, Eric D (1998). „Tempo and beat analysis of acoustic musical signals“. In: *The Journal of the Acoustical Society of America* 103.1, S. 588–601.

Schuster, Mike und Kuldip K Paliwal (1997). „Bidirectional recurrent neural networks“. In: *IEEE transactions on Signal Processing* 45.11, S. 2673–2681.

Shahriar, Nafiz (2020). *What is Convolutional Neural Network (CNN) — Deep Learning*. Zugriff am 27.07.2025. URL: https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5.

Sigtia, Siddharth, Emmanouil Benetos, Nicolas Boulanger-Lewandowski u. a. (2015). „A hybrid recurrent neural network for music transcription“. In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, S. 2061–2065.

Sigtia, Siddharth, Emmanouil Benetos und Simon Dixon (2016). „An end-to-end neural network for polyphonic piano music transcription“. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.5, S. 927–939.

Takeda, Haruto u. a. (2002). „Hidden Markov model for automatic transcription of MIDI signals“. In: *2002 IEEE Workshop on Multimedia Signal Processing.* IEEE, S. 428–431.

Telila, Yohannis, Tommaso Cucinotta und Davide Bacciu (2025). „Automatic music transcription using convolutional neural networks and constant-q transform“. In: *arXiv preprint arXiv:2505.04451*.

Vaswani, Ashish u. a. (2017). „Attention is all you need“. In: *Advances in neural information processing systems* 30.

Wikimedia Commons contributors (2016). *Recurrent neural network unfold.svg*. Zugriff am 27.07.2025. URL: `https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg`.

Wu, Yu-Te u. a. (2021). „Omnizart: A general toolbox for automatic music transcription“. In: *arXiv preprint arXiv:2106.00497*.

Wu, Zonghan u. a. (2020). „A comprehensive survey on graph neural networks“. In: *IEEE transactions on neural networks and learning systems* 32.1, S. 4–24.