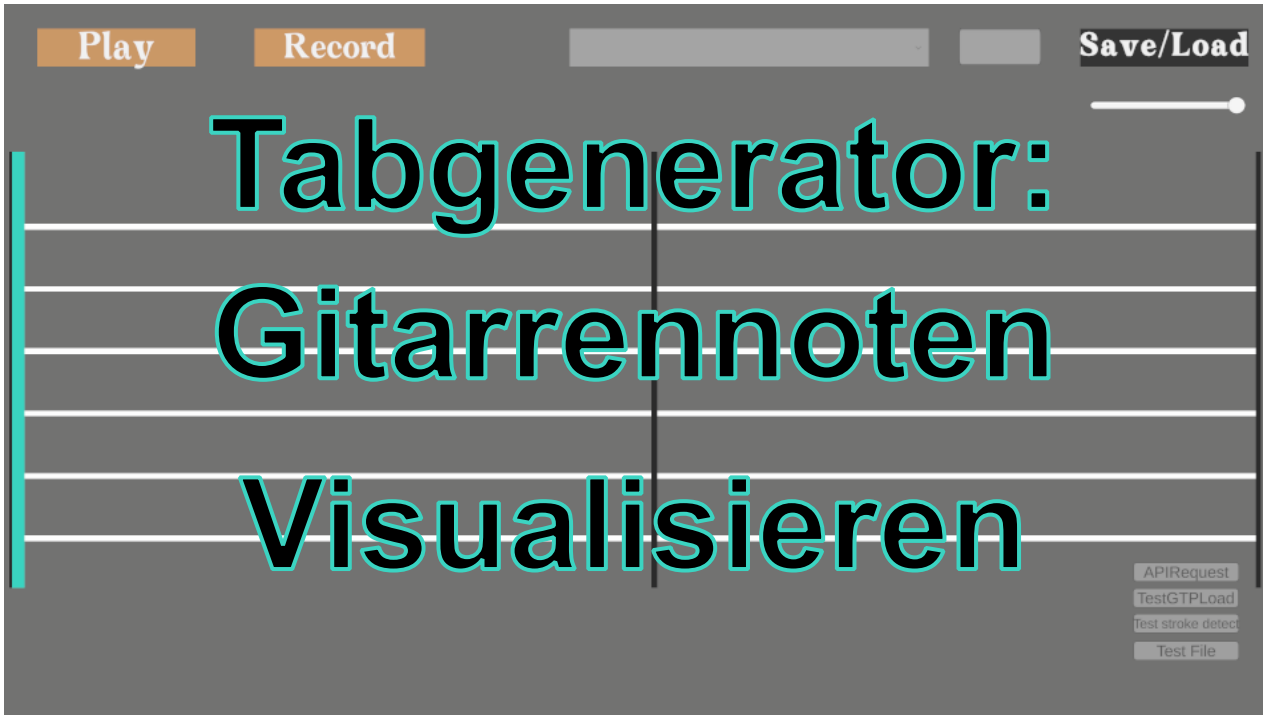


Tabgenerator: Gitarrennoten Visualisieren

Ben Kleinschnittger

Code: https://github.com/SuperPirate-ai/TabGenerator

Inhaltsverzeichnis:

Inhalt

Einleitung

Problemstellung

Bei einer Gitarre lernt man Lieder, wie bei vielen Instrumenten, mit Noten. Jedoch gibt es bei der Gitarre ein vereinfachtes System, welches speziell auf sie angepasst ist. Es nennt sich Gitarrentabs oder kurz einfach nur „**Tabs“**. Um einen Song in Tabs zu notieren, muss man, wie beim normalen Notensystem auch, jede Note **einzeln** eintragen. Dies ist ein zeitaufwendiger Prozess, der viele Nerven und Geduld kostet. Viel praktischer wäre es, diesen Prozess zu automatisieren. Und genau darum geht es auch in meinem Projekt. Ich möchte ein Programm entwickeln, das Gitarrenspielern und Musikproduzenten hilft Songs in Tabs zu notieren. Dies soll **automatisch** geschehen, um das Songschreiben **effizienter** und einfacher zu **benutzerfreundlicher**.

Generelle Idee der Umsetzung

Dieses Projekt beschäftigt sich mit der **automatischen** Visualisierung dieser Tabs mithilfe eines selbstgeschriebenen **Computerprogrammes**. Durch dieses Programm soll das Eintragen der Noten automatisiert werden, **ohne** dass **spezielles Equipment** benötigt wird. Dieses Projekt widmet sich auch der **Audioanalyse**, einem zentralen Thema mit wachsender Bedeutung in der heutigen Industrie. Für mich ist dies das erste Projekt, das seinen Fokus auf die Audioanalyse legt und ich hatte beim Start dieses Projektes keine ausschlaggebenden Kenntnisse von diesem Fachgebiet der Informatik.

Aufbau

Zur Umsetzung dieses Projektes nutze ich die **Spiele-Engine Unity** mit der Programmiersprache **C#**. Ich nutze hier eine Spiele-Engine aus Gewohnheitsgründen. Darüber hinaus bietet sie eine praktikable Grafikoberfläche.

Der analoge Aufbau besteht aus einer **Gitarre** und einem **Audiointerface**. Dieses dient als Schnittstelle zwischen Gitarre und Computer.

Open Source

Das Projekt wird nach der Entwicklung offen auf **GitHub**[1] zur Verfügung stehen. Dadurch ist eine **volle Transparenz** bei der Entwicklung und ein **freier Zugriff** für jeden gegeben. Somit sollen möglichst viele Menschen erreicht werden.

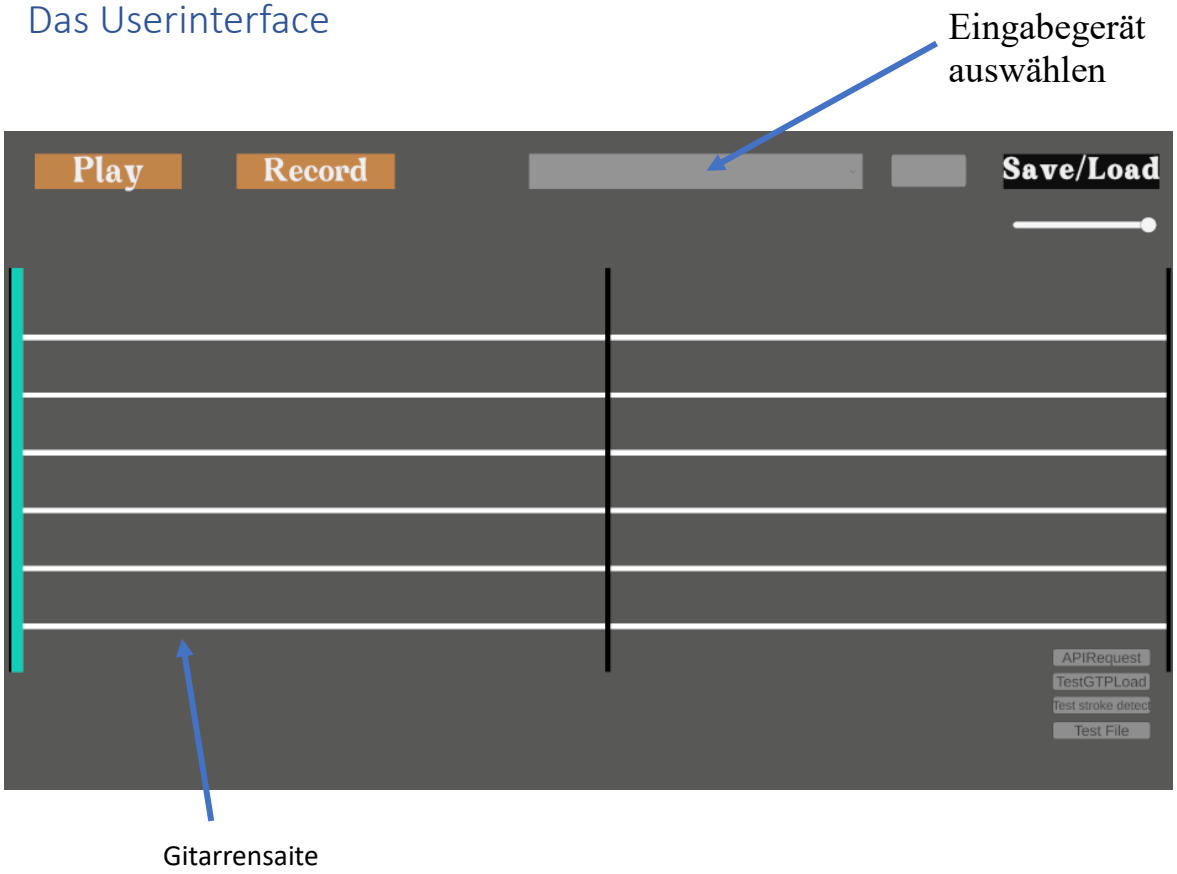
Grundidee

Die Grundidee besteht daraus mithilfe der **Fast Fourier Transformation** (FFT) das Ausgangssignal, in diesem Fall eine gespielte Note, zu analysieren, und somit die **stärkste Frequenz** in einem bestimmten Zeitintervall auszumachen. Dieser Frequenz, welche der gespielten Notenfrequenz entspricht, wird dann einer **Position im virtuellen Notensystem** zugeordnet und für den Nutzer dargestellt. Es besteht die Möglichkeit die gespielten Noten in einer Datei zu **speichern** und diese dann auch wieder zu **laden**.

Die Fast Fourier Transformation

Die **Fast Fourier Transformation**, kurz FFT, ist eine effiziente mathematische Operation zur Berechnung der **Diskreten Fourier-Transformation**. Die FFT ist wesentlich schneller als die herkömmliche Methode, da es mit dem **Teile-und-Hersche** Verfahren arbeitet. Dieses Verfahren teilt im Grunde ein Problem so lange rekursiv in kleine Teile, bis es im Teilproblem lösbar („beherrschbar“) ist. Anschließend wird die Teillösung genutzt, um die Lösung des Gesamtproblems zu rekonstruieren. Im großen Ganzen wandelt die FFT das Ausgangssignal vom **Zeitbereich** in den **Frequenzbereich** um. Wenn man das Signal in einem Graphen zeichnen würde, wäre vor der Kalkulation der FFT die X-Achse die **Zeit** und die Y-Achse die Amplitude, also die Lautstärke, des Signals. Nach der Kalkulation wäre die X-Achse die **Frequenz** und die Y-Achse die Amplitude. In diesem Projekt ist diese Funktion zur Frequenzbestimmung der einzelnen Noten von großem Nutzen. Für die Implementierung dieser Operation besteht die Möglichkeit sie entweder komplett selbst zu implementieren. Da dies aber mit sehr großem Aufwand verbunden ist und nicht dem Ziel dieses Projektes entspricht, wird in diesem Projekt eine Library genutzt. Genauer gesagt die Accord.Math Library. Die Implementierung der FFT findet man im AudioComponents Skript.

Das Userinterface



Die Umsetzung

Die Aufnahme einer Note

Zur Aufnahme einer Note wird das Signal, welches von der Gitarre „ausgesendet“ wird, zunächst mithilfe des Audiointerfaces auf den Computer übertragen. Da das ankommende Signal unendlich weitergeht, also weder einen klar definierten Anfang noch Ende hat, wird es in kleine Samples unterteilt. Diese haben eine **Buffer Size** von 8192 Sample. Bei einer zu kleinen Buffer Size besteht die höhere Wahrscheinlichkeit, dass die Note in zwei Samples aufgespalten wird, was die Identifikation erschwert. Die Aufnahme des Signals wird im **MicrophoneInput-Skript** geregelt.

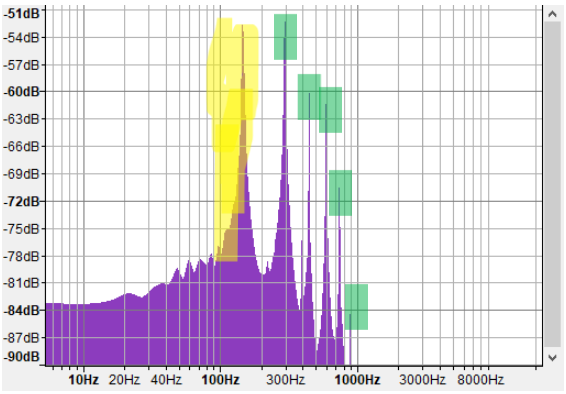
Die Audioanalyse

Die Audioanalyse findet hauptsächlich im **AudioAnalyzer-** und im **AudioComponent-Skript** statt.

Berechnung der Notenfrequenz

Berechnung des Frequenzspektrums

Nachdem das Signal in kleine Samples aufgeteilt wurde, muss nun jedes einzelne Sample analysieren werden, um die Frequenz, der in dem Abschnitt gespielten Note, zu erfassen. Dazu ist es am effizientesten die oben beschriebenen **Fast Fourier Transformation** zu nutzen.



Hier wurde ein D3 mit 145 Hz gespielt. Die **gelbe** Markierung entspricht der Notenfrequenz

Bestimmung der Hochpunkte

Wie man in der obigen Abbildung erkennt, kann man die gespielte Note anhand ihrer Amplitude (Lautstärke) bestimmen. Dazu schaut man sich alle **Hochpunkte** (grün) der FFTs an. Um Noise zu ignorieren, setzt man einen minimal **Amplituden-Threshold** für die Amplitude. Der tiefste Hochpunkt entspricht der Grundfrequenz, also der Frequenz der gespielten Note.

FFT-Ungenauigkeit

Nun ist es leider so, dass die FFT eine gewisse Ungenauigkeit bei der Frequenzbestimmung hat. Diese hat vor allem mit der Größe der **Buffer Size** zu tun. Je größer, desto genauer. Allerdings bleibt immer eine gewisse Fehlerquote. Diese ist vor allem bei tiefen Noten ein Problem, da hier die Frequenzen sehr nah beieinander liegen. Um diese Abweichungen von der eigentlichen Frequenz auszugleichen, sind die **Obertöne** von Relevanz. Diese sind die Hochpunkte, welche nach der Grundfrequenz kommen.

Obertöne und die genaue Frequenzbestimmung

Die Obertöne sind diejenigen Frequenzen, die mitschwingen, wenn eine Note, also eine bestimmte Frequenz gespielt wird. Sie sind immer ein **Vielfaches** der Grundfrequenz. Wenn man also ein tiefes A spielt, ist die Grundfrequenz 110 Hz, der erste Oberton 220 Hz, der zweite 330 Hz, der dritte 440 Hz und so weiter.

Nun ist es wichtig anzumerken, dass je höher die Frequenz einer Note ist, desto größer ist ihr Abstand zur nächstliegenden Note. Folglich ist es einfacher Noten mit hohen Frequenzen zu unterscheiden, als Noten mit niedriger.

Wenn wir nun den obigen Gedanken aufgreifen und ihn auf unser Problem mit der Ungenauigkeit der FFT anwenden, können wir uns die jeweiligen Obertöne anschauen, deren Frequenz bestimmen und diese dann auf die Grundfrequenz runterrechnen. Somit würden wir die **Genauigkeit** um einiges verbessern. Die Obertöne sind in der obigen Abbildung **grün** markiert und entsprechen den Hochpunkten.

Allerdings haben wir etwas wichtiges außerachtgelassen. Um die Hochpunkte zu bestimmen haben wir einen **Threshold** gesetzt. Was ist aber nun, wenn wir uns die Obertöne anschauen und einer oder mehrere unter dem Threshold liegen? Dann würden wir nicht alle erfassen und da wir auch nicht wissen, welche Obertöne wir haben und welche nicht, könnten wir auch nicht sagen den **wievielten** Oberton wir haben. Das heißt wir können die Grundfrequenz nicht berechnen.

Um dieses Problem zu lösen, erhöhen wir **iterativ** den Index und teilen dann die **Obertonfrequenz** durch den **Index** und anschließend durch die bis jetzt ungenaue **Grundfrequenz**. Der Index, bei dem das Ergebnis am kleinsten ist, wird als Index für den Oberton verwendet und die vom Oberton resultierende Grundfrequenz als neue Grundfrequenz. Dies machen wir für jeden Oberton bis wir das genauste mögliche Ergebnis haben.

Saitenanschlag erkennen

Bis jetzt ist es möglich die **Grundfrequenz** zu bestimmen. Allerdings besteht nun immer noch das Problem, dass man eben nicht immer exakt alle 8192 Samples, also jeden Aufnahme Buffer eine neue Note spielt. Sondern eben nur dann, wenn eine Saite angeschlagen wird. Dafür muss man sich überlegen, wann genau eine neue Note gespielt wird, die wir dann auch anzeigen wollen. Dies geschieht theoretisch immer, wenn wir einen **drastischen Amplitudenanstieg** haben, also eine Saite anschlagen, da das Spielen der Saite mit dem Finger oder einem Plektrum, die Amplitude erhöht. Dieser Ansatz funktioniert zwar im praktischen und wird auch in meinem Projekt zum Teil implementiert. Allerdings gibt es für einen Großteil der Fälle einen genaueren und simpleren Weg, um zu bestimmen, ob eine neue Note gespielt wurde. Da sich bei fast allen Noten, die man neu anspielt, nicht nur die Amplitude ändert, sondern auch die **Grundfrequenz**. Wenn diese stark von der vorherigen Grundfrequenz abweicht, kann man sich sicher sein, dass eine neue Note gespielt wurde. Dieser Ansatz funktioniert allerdings nicht, wenn dieselbe Note öfter hintereinander gespielt wird. Dafür wird die erstere Lösungsidee umgesetzt.

Erschließung der nächstliegenden Notenfrequenz

Nachdem die **Grundfrequenz** des jeweiligen Samples bestimmt und der **Saitenanschlag** erkannt wurde, muss jetzt die Grundfrequenz, einer Frequenz auf dem **Gitarrenspektrum** zugeordnet werden.

Guitar

OPEN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C
330	349	370	392	415	440	466	494	523	554	587	622	659	698	740	784	831	880	932	988	1047
B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G
247	262	277	294	311	330	349	370	392	415	440	466	494	523	554	587	622	659	698	740	784
G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#
196	208	220	233	247	262	277	294	311	330	349	370	392	415	440	466	494	523	554	587	622
D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#
147	156	165	175	185	196	208	220	233	247	262	277	294	311	330	349	370	392	415	440	466
A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F
110	117	123	131	139	147	156	165	175	185	196	208	220	233	247	262	277	294	311	330	349
E	F	F#	G	G#	A	A#	B	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C
82	87	92	98	104	110	117	123	131	139	147	156	165	175	185	196	208	220	233	247	262

Note and frequency in Hertz (approx.) for standard tuning.

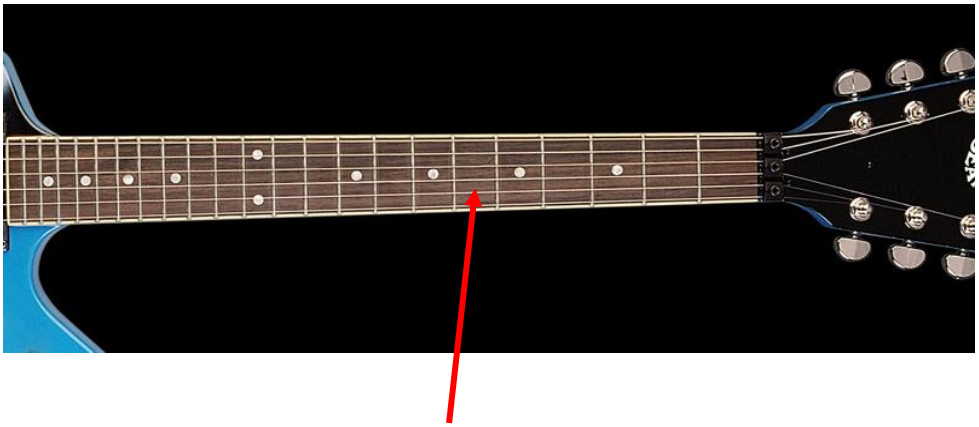
[Bild1]

Hierbei wird analysiert, welche Frequenz auf dem Gitarrenspektrum der Hauptfrequenz am nächsten liegt. Wichtig ist, dass es viele Frequenzen mehrfach auf der Gitarre gibt. Dies ist später von Relevanz.

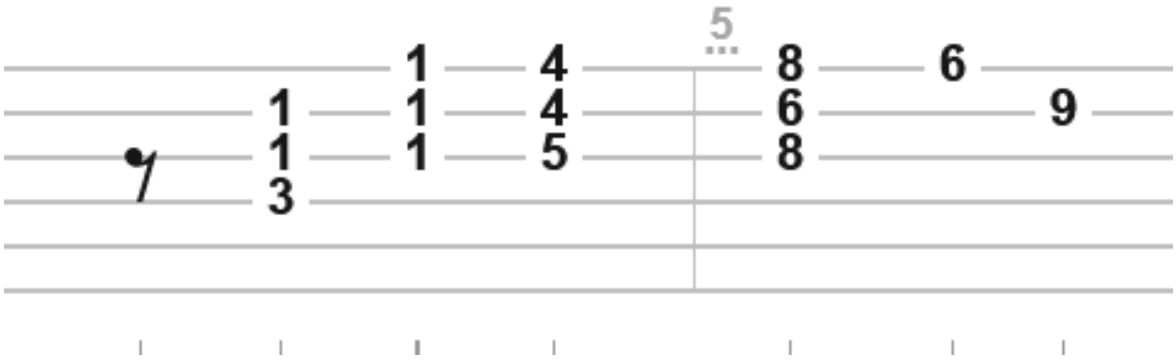
Die Visualisierung

Aufbau der Gitarre und der Tabs

[Bild2]



Der Gitarrenhals ist in sogenannte **Bünde** aufgeteilt. Diese sind jeweils mit **Bundstäbchen** getrennt. Außerdem hat er noch **sechs Saiten**. Jeder Bund hat einen **Halbton** Abstand zum nächstgelegenen Bund. Also auf der obersten Saite, welche leer gespielt einem E entspricht, haben wir dann im ersten Bund ein F und im zweiten ein F# und so weiter.



[Bild3]

Im **Tabs-System** gibt es ebenfalls diese sechs Saiten, als Striche dargestellt. Allerdings sind diese genau **andersherum** als auf der Gitarre. Also die höchste Saite ist oben und die tiefste unten. Die Bünde werden als Nummern auf der jeweiligen Saite dargestellt.

Bestimmung der Notenposition im visuellen Bereich

Mithilfe der vorher ermittelten Frequenz ist es nun möglich die dazu passenden Positionen auf der Gitarre zu bestimmen. Da die ermittelte Frequenz häufig mehr als einmal vorkommt, werden **mehrere mögliche Positionen** ermittelt, die auf der Gitarre gespielt worden sein könnten. Zum Beispiel kann man das G4 auf der G-Saite im 12. Bund oder auch auf der D-Saite im 17. Bund gespielt werden.

Berechnung der tatsächlich gespielten Notenposition

Nun herauszufinden, wo tatsächlich die Note gespielt worden ist, gestaltet sich etwas schwierig. Es ist kaum möglich, ohne eine zweite Quelle, wie zum Beispiel einer **Kamera**, herauszufinden, wo die Note gespielt worden ist, da jede Notenposition exakt derselben Note und somit derselben Frequenz entspricht. Wichtig anzumerken ist aber, dass jede **Notenposition** auf einer anderen **Saite** liegt. Schafft man es also die Saiten voneinander zu differenzieren, so schafft man es auch eine eindeutige Notenposition zu identifizieren. Hier für könnte man sich **saitenspezifische Eigenschaften** zunutze machen oder aber versuchen die bestmögliche Notenposition in Relation zu den bereits gespielten Noten mithilfe von **künstlicher Intelligenz** zu bestimmen.

Identifizierung der Saite mithilfe von Künstlicher Intelligenz

Da es sich zunächst schwierig gestaltete genaue Differenzen zwischen den Saiten anhand vom Audiosignal zu klassifizieren, bot sich eine **einfachere** und durchaus mögliche Option an, auf einem anderen Weg zu versuchen die gespielte Saite zu erkennen. Diese Identifizierung basiert nicht auf einer exakten physikalischen Identifizierung, sondern auf einem fundierten Ratespiel, Unterstützung durch **Künstlicher Intelligenz**.

Das Five-Note-Pattern

Die Idee basiert auf schon bestehenden Tabs, aus denen **Patterns** extrahiert werden. Diese Patterns bestehen aus **fünf** Noten. Die Noten bestehen aus einer **Frequenz** und der jeweiligen **Saite**, auf welcher sie gespielt werden. Damit sollen die häufigsten Notenkombinationen klassifiziert werden, um diese nachher in einem **Machine-Learning-Model** zu trainieren.

Diese sogenannten **Five-Note-Patterns** sind somit Kombinationen an Noten, die häufig gespielt werden und für den Gitarrenspieler leicht erreichbar und spielbar sind und deshalb auch mit größter Wahrscheinlichkeit von ihnen gespielt werden. Dafür ist eine repräsentative Menge an Tabs von Nöten.

Wenn wir nun genügend Patterns extrahiert haben, brauchen wir auch Negativbeispiele oder auch **Badpatterns**, also fünf Noten Patterns die als schlecht klassifiziert werden und unter normalen Umständen nicht gespielt werden. In diesen Patterns liegen die Noten weit auseinander, sind umständlich zu greifen und haben somit eine sehr geringe Wahrscheinlichkeit gespielt zu werden. Da diese aus offensichtlichen Gründen nicht vorhanden sind, werden sie **selbst generiert**.

Diese zwei Arten von Trainingsdaten werden nun dem Model zum Trainieren gegeben. Im Model nutze ich eine **Binäre Kreuzentropie** als Loss-Funktion und ein **Adam Optimizer**.

Das heißt also ich habe bei den Trainingsdaten nur eine **binäre Zuordnung**. Entweder ist es ein gutes oder ein schlechtes Pattern. Dadurch sind die erzielten Resultate oft fehlerhaft. Besser ist es sogenannte **Gewichtungen** einzuführen und Patterns zu bewerten. Also statt zu sagen dies ist ein gutes oder ein schlechtes Pattern, eher eine prozentuale Klassifizierung durchzuführen. Dies stellt sich allerdings als äußerst schwierig heraus, da man hier für ein bestimmtes **Bewertungsschema** bräuchte, um sie entweder selbst zu bewerten, bei knapp 2000 Patterns eine unschöne Idee, oder einen Algorithmus zu entwickeln, der diese Aufgabe übernimmt. Allerdings ist so ein Bewertungsschema sehr kompliziert und es entzieht sich meiner Kenntnis, wie man so etwas entwickeln könnte. Aus diesem Grund habe ich meinen Fokus auf die folgende Idee gestützt.

Identifizierung der Saite mithilfe von geeigneten physikalischen Eigenschaften

Nach längerer Recherche mit einem MIT Studenten aus den USA, bin ich auf eine Arbeit des Professors Daniel A. Russel von der Pennsylvania State University gestoßen, der einen anderen möglichen Ansatz skizziert, wie man dieses Problem angehen könnte. Der Professor geht in seiner Arbeit auf ein physikalisches Phänomen ein, welches vor allem bei Saiteninstrumenten auftritt und es ermöglicht Saiten voneinander zu unterscheiden: die sogenannte **Inharmonicity** (deutsch: Inharmonizität). Dieses Phänomen beschreibt, die

Abweichung der Obertöne von der eigentlichen Vielfachheit der Grundfrequenz. Dies geschieht durch die **Steifheit** einer Saite (engl. Stiffness).

Unterschiedliche Saiten sind unterschiedlich stark gespannt. Deshalb weichen die Obertöne auch unterschiedlich von den **Soll-Harmonien** ab, je nachdem welche Saite gespielt wurde.

Man kann sich diese Eigenschaft zunutze machen, indem für jede Saite eine Abweichung der Obertöne bestimmt und diese dann bei jeder neuen Note verglichen wird. Dazu kann folgende Formel verwendet werden:

$$B = \frac{\left(\frac{f_n}{f_0 * n}\right)^2 - 1}{n^2}$$

B ist eine Konstante, die von der Saitendicke abhängig ist. Sie kann mittels der Formel rechnerisch anhand der produzierten Frequenzen errechnet werden. $\mathbf{f_0}$ ist hierbei die Grundfrequenz, $\mathbf{f_n}$ die Obertonfrequenz und **n** der Index des Obertons.

So ist es möglich die Saiten voneinander zu differenzieren und dies auch noch exakt an physikalischen Eigenschaften fest zu machen.

[Artikel1]

Anzeigen der Noten

Mittels der bestimmten Frequenz und Saite, kann die passende Notenposition nun als Notenobjekt dargestellt und mit einer bestimmten Geschwindigkeit abgespielt werden. Diese ist abhängig von den Beats per Minute (kurz: BPM).

Das Speichern und Laden

Zusätzlich besteht die Option, die gespielten Noten in einer Datei zu speichern und diese dann zu einem anderen Zeitpunkt wieder abzurufen. Momentan ist dies nur in einem programmeigenen Format möglich. Dies soll noch ausgeweitet werden, sodass die Noten in der gebräuchlichen Form als .gp5 Datei abgespeichert werden können und somit mit den regulären Programmen, wie GuitarPro und Songsterr, nutzbar sind.

Quellen

[Bild1] https://i.pinimg.com/originals/6c/3e/7b/6c3e7bb6076c2c9eff8918a28bd6e67e.jpg

[Bild2] http://www.edroman.com/guitars/dean/dimebag/dean_from_hell_cfh_enlarged.jpg

[Bild3] https://www.songsterr.com/a/wsa/george-benson-song-for-my-father-tab-s417389

[Artikel1] https://www.acs.psu.edu/drussell/Demos/Stiffness-Inharmonicity/Stiffness-B.html