ChatGPT

音源分離からMIDI・ギターTAB譜生成までの技術調査

音源分離（ボーカル・ギター・ドラム等の分離）

楽曲音源から各パート（ボーカル、楽器ごと）を分離するには、近年発展した**音源分離（source separation）**のAIモデルやライブラリが利用できます。代表的なオープンソース・ライブラリを以下にまとめます：

- **Spleeter**（Deezer社） – Python（TensorFlow）製の音源分離ライブラリ [1]。事前学習済みモデルにより手軽に高品質な分離が可能で、2 stem（ボーカル+伴奏）、4 stem（ボーカル・ドラム・ベース・その他）、5 stem（ボーカル・ドラム・ベース・ピアノ・その他）のモデルが提供されています [2]。特に4 stemモデルは標準的なデータセットMUSDB18で高い性能を示し、GPU使用時には実時間の100倍の速度で処理できると報告されています [3]。コマンドラインツールまたはPython APIで利用でき、導入も `pip install spleeter` で簡単です。処理精度は後述のDemucsに若干劣るものの、**動作の軽快さと手軽さ**で優れています [4]。

- **Demucs**（Facebook AI Research） – Python（PyTorch）製の最先端モデル [5]。**時間領域のU-Netアーキテクチャ**にLSTMやTransformerを組み合わせ、高精度な分離を実現しています [6] [5]。標準では4 stem（ボーカル・ドラム・ベース・その他）分離モデルが提供され、MUSDB HQデータセットでSDR 9.2dBと**最新モデル中トップクラスの性能**を達成しています [5]。v4では実験的に6 stem（ボーカル・ドラム・ベース・**ギター**・ピアノ・その他）のモデルも公開されており、ギター専用の出力にも挑戦しています（ギターは「まずまず」だがピアノにはアーティファクトが多いと報告） [7]。Demucsは高音質な代わりに計算量が非常に大きく、**処理速度はSpleeterより遅い**です [4]。GPUなしでの長時間音源処理は困難なため、Webアプリで扱う場合はサーバー側でGPU利用するか、必要に応じてバッチ処理とする構成が望ましいです。また、DemucsもオープンソースでありPyPIからインストール可能です。

- **Open-Unmix (UMX)** – Python（PyTorch）実装のオープンソース分離モデル [8]。周波数領域ベースのモデルで、4 stem（ボーカル・ドラム・ベース・その他）の分離に対応しています [8]。精度はSpleeterやDemucsに比べると中程度ですが、**シンプルで扱いやすい**リファレンス実装です。派生版の**Cross-Net UMX (X-UMX)** や **D3Net** なども研究公開されており、Spleeter Web（後述）ではこれらを切り替えて使用する例があります [9]。Open-UnmixはONNXモデルやTensorFlow.jsへの変換例もあり、実際に**ブラウザ上でモデルを動作**させたデモも公開されています [10]（UMXモデルをONNX経由でTensorFlow.js化した例）。ただしブラウザ実行の場合は**処理が非常に遅く**、30秒以上の音源には非推奨とされています [11]。

- **その他のツール**:

- *Asteroid*（PyTorch製の音源分離フレームワーク）や *nussl*（オーディオ研究向けライブラリ）も、Conv-TasNetやDemucsなど複数手法を統一的に利用できるオープンソースプロジェクトです [12]。これらは柔軟な実験やカスタムモデル統合に適しています。
- *LALAL.AI* や *Moises* など商用APIも存在しますが、オープンソースではないため、本調査では主に自前実装可能なOSSを中心に取り上げます。

精度や制約: 音源分離モデルの性能はトレーニングデータに大きく依存します。多くのオープンモデル（SpleeterやDemucsなど）はMUSDB18/HQ [13] [14] というポップス中心のデータセットで訓練されており、**一般的なポップ/ロック系楽曲では良好な分離**が期待できます。一方、ジャズの生録音やクラシック、民族音楽など**訓練分布から外れたジャンル**では性能が落ちたり、対象外の楽器音（例: バイオリンや笛など）が「その他」トラックに混ざり精度低下する可能性があります。また、入力音源にリバーブやエコーが強い場合、分離後に**残響やノイズ成分**が各ステムに混入することがあります（例：Demucs初期版ではサイレント部分に微かなヒスノイズが加わるとの報告がありました [15] ）。**ギターの分離**については、既存モデルのデフォルトでは「その他」に含まれることが多く、完全なギター単独抽出は難しいケースがあります。Demucs v4の6stemモデルのようにギターを明示的に分離する試みもありますが、ピアノなど他の楽器と比べ精度は限定的です [7] 。必要に応じて、まず5stemモデルでピアノを分け、残る「その他」からギター音を抽出する工夫や、ギターに特化したモデル（訓練データを増やす、エフェクトの少ない生演奏データで微調整する等）を検討するとよいでしょう。

音高・音長推定とMIDI変換（自動採譜）

分離した各パート音源から**音高（ピッチ）や音の開始/終了（音長）**を推定しMIDIノート列に変換する工程です。これは自動採譜（Automatic Music Transcription, AMT）と呼ばれ、多くの研究・ライブラリがあります。要求精度や対象とする楽器の種類に応じて、以下のツール・手法が利用可能です。

- **Basic Pitch**（Spotify開発） – **オープンソースの音声→MIDI変換モデル**です [16] 。Pythonライブラリおよびブラウザ実行デモが公開されており、汎用性と軽量さが大きな特徴です。Basic Pitchは**楽器非依存**かつ**ポリフォニック対応**で、ピアノ・ギター・歌声など様々な音源から複数音の同時検出が可能です [17] 。さらに**ピッチベンド（音程の連続的な変化）**も捉えられるため、ギター特有のチョーキングやスライド、ボーカルのビブラート等もMIDIのベンド情報として出力できます [18] 。モデルサイズが小さく**リアルタイムより高速**に動作するよう工夫されており [19] 、実際ブラウザ上でも重くならずに動作します（Web版デモではモデルをTensorFlow.jsでロードし、音声をアップロードせずクライアント側で処理 [20] ）。Basic Pitchはニューラルネットワークによるオンセット検出＋フレームごとのマルチピッチ推定を組み合わせたモデルで、**精度と軽量性のバランス**に優れています [21] [22] 。Spotifyの発表によれば、専門のピアノ自動採譜モデルに匹敵する精度を保ちつつ、多様な楽器・歌声に対して良好な結果を示したとのことです [23] 。GitHub上でコードと学習済みモデルが公開されており [16] 、Pythonから簡単に利用できます。ボーカルなど単旋律から、ギターの和音を含む演奏まで幅広く適用できるため、本用途には最有力と言えます。

- **Onsets and Frames**（Magenta, Google Brain） – ピアノ音源向けに開発された深層学習ベースの自動採譜モデル [24] 。音の開始（Onset）と持続（Frame）を別々の出力として学習することで、ピアノ独奏の高精度な採譜を実現しています [25] 。Magentaの実装や学習済みモデル（MAESTROデータセットで訓練）が公開されており、**ソロピアノ曲**であれば非常に高い再現性でMIDI化可能です。ただし**対象は主にピアノ**で、他楽器やバンド音源には最適化されていません。またモデルが大型で推論にGPUを要するため、Webサービスに組み込むには非現実的です。ピアノ以外の楽器にはBasic Pitchの方が適していますが、Onsets and Framesの研究は「オンセット＋持続」の考え方としてBasic Pitchなど他のモデルにも応用されています [26] 。

- **aubio** – Cで実装された軽量音楽分析ライブラリで、Pythonバインディングもあります。ディープラーニングを使わず**信号処理アルゴリズム（FFTベース）**で基本周波数やオンセット検出を行います。**モノフォニック（単旋律）音源**のピッチ追跡に使え、リアルタイム処理も可能なほど高速・軽量です。ただし和音の分離や多声音の同時検出はできません。例えばボーカルのメロディライン抽出や、ギター独奏フレーズ

のラフなMIDI変換には使えますが、**コード演奏には不向き**です。精度面も機械学習ベースには及ばないため、Basic Pitch等の利用が推奨されますが、シンプルな用途では選択肢となります。

- **CREPE** – Googleが公開した**ディープラーニングによる高精度単音ピッチ検出**モデル（TensorFlow/PyTorch）。楽器や声問わず単一の基本周波数を追跡でき、微小なピッチ変動も高解像度で取得します。モノラル音の音高抽出専用ですが、その精度から研究用途で広く使われています。例えばギター独奏を1弦ずつ録音したような場合にはCREPEで各弦のピッチ列を取得する、といった応用も可能です。ただし**多声音には対応しない**ため、やはり一般音源の自動採譜にはBasic Pitchなどマルチピッチ対応モデルが必要です。CREPEはPython用ライブラリ・学習済みモデルが公開されています。

- **その他**: *Essentia*（音響特徴抽出ライブラリ）は、MELODIAアルゴリズムによるメロディ抽出などを収録していますが、扱いが難しく大規模です。またヤマハの研究でVOCALOID用に開発された*HTT*音高変換など商用技術もありますが、入手性の観点でここでは割愛します。Spotify Basic Pitchが登場する以前は、Justin Salamon氏の**Melodia**（メロディ抽出アルゴリズム）や、Ibanezらの**pYIN**（モノフォニックF0検出）などがよく使われていました [27]。現在はBasic Pitchのような軽量NNモデルにより、ある程度包括的に対応できるようになっています。

精度や制約: 自動採譜の精度も音源の特性に左右されます。**音源分離を事前に行うことの利点**はここで大きく現れます。たとえばバンド音源からギターの音高を取る場合、ミックス全体から直接検出するのは困難ですが、ギター成分を抽出した上でBasic Pitchなどにかければ、他の楽器に邪魔されず精度が向上します。ただし分離の不完全さ（他楽器の混入や分離アーティファクト）は採譜結果にも影響し、ノイズが**誤検出（ファルスノート）**として出力される場合もあります。またギターの和音演奏では倍音が複雑に重なり合うため、必ずしも正確にすべての音を拾えるとは限りません。Basic Pitchはポリフォニック対応ですが、それでも高密度な和音や速弾きフレーズでは見逃しやリズム位置のずれが発生し、**MIDIノートのタイミング修正**や**不要ノートの削除**といった**手動の微調整**は避けられないでしょう。特にドラム音やノイズが残っていると誤検出しやすいため、可能であれば**対象パート専用に録音された音源**（例：ギターのライン録音）を用意するか、分離モデルの品質向上を図ることが重要です。

関連データセットやモデル: 自動採譜の研究には、ピアノならMAESTROデータセット、歌声ならMir1k等、ギターでは次項のGuitarSetなどが利用されています。Spotify Basic Pitch自体は様々な楽器の音源データセットで学習・検証されており [23]、歌声データ（Molinaデータセット）に対しても良好な結果を示しました [23]。ギター音源向けには、NYUらが公開した**GuitarSet**という貴重なデータセットがあります [28]。GuitarSetには6本のギター弦それぞれを独立録音できる特殊なピックアップで収録した音源が含まれ、各音の**ピッチカーブや発音タイミング、対応する弦・フレット位置（タブ譜）**までラベリングされています [28] [29]。このような高精度データにより、ギターに特化した自動採譜モデルの研究も進んでいます [30]。たとえば近年の研究では、Transformerを用いて**MIDIからギタータブへの変換**（後述）や、**オーディオから直接タブ譜を推定**する試みもあります [31]。これらのモデルはGuitarSetや、インターネットから収集した大規模タブ譜データセット**DadaGP**（GuitarPro形式26,000曲以上） [32] で訓練されています [33]。現状では実験段階ですが、将来的には音声から直接TAB譜を起こす高度なAIアシスタントも期待できます。

MIDIからギターTAB譜への変換

自動採譜したMIDIデータ（音符の高さとタイミング情報）を**ギターのタブ譜形式**に落とし込むステップです。MIDI自体は「どの音をいつ鳴らすか」は記述できますが、「どの弦の何フレットで押さえるか」といった運指情

報は含みません。そのため**ギター特有のタブ譜への変換**には追加の処理が必要です。ポイントは以下の通りです：

- **音符の弦・フレット割り当て**: ギターは同じ音高の音を複数の場所で弾けるため（例：音高A4は1弦5フレットでも2弦10フレットでも出せる）、各MIDIノートをどの弦の何フレットに対応させるか決める必要があります。単音フレーズであれば簡単で、例えば**基本的な割当戦略**として「なるべく開放弦や低いポジションを優先する」「物理的に不可能な運指（1本の弦に同時に2音など）は避ける」といったルールで決定できます。和音（複数ノートが同時）については、一度に6音まで・1弦1音という制約で**組み合わせ探索**を行い、全音をカバーする弦割当を求めます [34] [35]。全探索すると組み合わせ爆発しますが、ギターでは現実的に近接したポジションでしか押さえられないため、例えば**「連続5フレット以内に収まるポジションのみ許容」**など制限することで解を絞り込みます [35]。割当候補が複数ある場合、**直前直後の運指とのつながり**（ポジション移動の少なさ）や**開放弦の利用**などのスコアを付け、最適経路を選ぶ方法（動的計画法や遺伝的アルゴリズムの応用）も研究されています [36] [37]。実装を簡略化するなら、**まず低い弦から優先して音を当てていく**貪欲法でもそれなりの結果は得られます。ただしこの場合、ギターのポジション移動が頻繁になり不自然な運指になることも多いため、出力後に人間が修正できる前提で割り切ることも必要です。実際、2023年の研究ではTransformerモデルでMIDI→TAB変換を試み、**DadaGPで学習したAIが人間らしい運指を提案**できることを示しています [33]（それでも約17〜18%の音で誤った弦選択が起きるとの報告 [38]）。このような高度なアプローチもありますが、まずは**シンプルなアルゴリズム**で実装し、ユーザが後から運指を調整できるワークフローを整えるのが現実的でしょう。

- **TAB譜フォーマットとライブラリ**: 一旦弦とフレットの対応が決まれば、出力フォーマットとしてはテキストの**タブ譜表示**だけでなく、専用のファイル形式で提供すると便利です。一般的なギター譜フォーマットとしては**MusicXML**と**Guitar Pro形式**があります。MusicXMLは楽譜交換用のオープン標準XMLで、タブ譜情報（弦番号とフレット番号、調弦情報など）を記述可能です [39] [40]。実際、`<technical>`タグ内に`<string>`（弦）と`<fret>`（フレット）要素を入れることで、その音符がどの弦・フレットかを指定できます [40]。MusicXMLに対応した楽譜ソフト（**MuseScore**や**Finale**、**Sibelius**等）やWebライブラリ（後述のAlphaTabなど）で読み込めば、TAB譜として表示・編集できます。一方、**Guitar Pro**形式（*.gp* や *.gp5*, *.gpx* など）はギター譜作成ソフトのデファクト標準ですがバイナリ形式です。ただ幸い、オープンソースで互換ライブラリがいくつか存在します。例えばPython製の**PyGuitarPro**は、GP3/4/5形式の**読み書きと操作**を可能にするライブラリです [41]。もともとAlphaTab（後述）やTuxGuitarのコードをPythonに移植したもので、ギター譜データ（曲情報や小節、音符、弦・フレットなどのモデル）を操作し、GP5以前の形式でファイル出力できます [42]。GPX（GP6）や最新GP（GP7/8）はXMLやZipベースのフォーマットですが、現状PyGuitarProは対応していません。ただしGP5形式までででも音符・TAB情報は保持でき、Guitar ProソフトやTuxGuitarで読み込んで最新形式に変換することも可能です。またAlphaTabは**GP3〜7および MusicXML**のインポートに対応しており [43]、これらのファイルをそのままWeb上で表示・再生できます。以上より、**最終出力**としては**MusicXML**または**Guitar Pro形式（GP5推奨）**でユーザに提供すると良いでしょう。ユーザは好みの楽譜エディタで開いて運指や表記を編集できますし、MusicXMLであればMuseScoreでの編集やPDF楽譜化も容易です。

- **タブ譜表示・編集の連携**: Webアプリ上で結果を確認・微修正できるようにしたい場合、**AlphaTab**ライブラリが有力です。AlphaTabはJavaScript/TypeScript製の楽譜・TAB譜レンダリングライブラリで、ブラウザ上でギター譜を表示し再生する機能を提供します [44]。インポート可能な形式は前述の通り多岐にわたり、特にGuitar Pro 3〜7とMusicXMLに対応しています [43]。内部的には譜面データモデルを持ち、読み込んだ譜面をHTML5 CanvasやSVG上に**楽譜（五線譜やタブ譜）として描画**します [45]。また組み込みのMIDIシンセサイザ（TinySoundFontベース）で譜面を再生することも可能です [46]。AlphaTabを使えば、バックエンドで生成したMusicXMLやGPファイルをフロントエンドに渡して表示・試聴させる、と

いったことが比較的容易に実装できます。さらにAlphaTabのデータモデルを直接操作すれば、ウェブ上での譜面編集機能を作ることも可能です（運指の変更や音符の追加削除など）。ただし完全なエディタを実装するのは難易度が高いため、基本は**生成結果のプレビュー**と**ファイルダウンロード**を提供し、詳細な編集はユーザが手元の専用ソフト（Guitar ProやMuseScore等）で行う、という形が現実的でしょう。

Webアプリ実装のポイント（フロントエンド・バックエンド構成）

以上の機能を統合してWebアプリケーションとして提供する際の構成例を示します。

- **バックエンド**: 音源分離や自動採譜には計算が重くGPUが必要な処理が多いため、通常はサーバー側でこれらを行います。Pythonで実装する場合、FlaskやDjangoでAPIサーバーを構築し、ユーザがアップロードした音源ファイルを受け取って処理する形になります。実例として、前述のSpleeterを組み込んだ「Spleeter Web」というOSSプロジェクトでは、**Django（Python）をバックエンドAPI**に、Reactをフロントエンドに採用し、非同期処理にCelery+Redisのジョブキューを用いる構成を取っています [47]（Dockerによるデプロイもサポート [48]）。このようにバックエンドで時間のかかる処理をキューイングし、完了後に結果ファイルを用意してフロントに通知する仕組みは、長い処理でもタイムアウトせず安定して提供する上で有効です。バックエンドでは上記で挙げたライブラリ群を組み合わせ、例えば**音源分離にはSpleeter or Demucs**、**音高検出にはBasic Pitch**、**TAB割当とファイル生成にPyGuitarPro**…といったパイプラインを実装します。各ステップで得られた中間生成物（分離後の音声、推定MIDI、生成したMusicXML/GPファイル等）は必要に応じてユーザにダウンロード提供したり、プレビュー用にフロントへ送ったりできます。

- **フロントエンド**: ユーザインタフェースでは、ファイルアップロードと処理結果の表示・ダウンロードが主になります。アップロードには一般的なフォーム送信や、ファイルを読み込んでバックエンドAPIへ送信するXHR/Fetchなどを使います。結果の**視覚的フィードバック**として、例えば音源分離の結果を各トラックごとに再生・ミキシングできるプレイヤーや、採譜結果の楽譜表示などが考えられます。音声再生にはブラウザの**Web Audio API**で複数音源を同期再生したり、単純に各分離音源ファイルの<audio>要素を用意するだけでも実現できます。採譜結果の楽譜表示には前述の**AlphaTab**を組み込めば、MusicXMLやGuitarProファイルを直接レンダリングしてTAB譜プレビューと再生が可能です。もし簡易的にテキストのタブ譜（いわゆるASCII Tab）で良いなら、サーバー側でMusicXMLをパースしてテキスト生成するか、あるいはPyGuitarProで一度GP5を書き出しTuxGuitar等でASCII変換する方法もあります。ただ、せっかくMusicXML等標準フォーマットが使えるので、ブラウザでのリッチな表示・編集体験を目指すならAlphaTabの活用が有力です。AlphaTabはnpmパッケージが提供されており、ReactやVueからも利用できます。また**楽譜編集を可能にするか**も検討事項です。完全な編集機能は難しくとも、例えば「検出結果のMIDIノートを削除する」「運指番号（ポジション）を変更する」といった限定的な編集であれば、AlphaTabのデータモデルを操作するスクリプトを用意することで実現可能です。とはいえ時間がかかる部分でもあるため、まずは**自動生成→ダウンロード**までを実装し、その先の細かな修正はユーザ環境に委ねる形でも充分実用的でしょう。

- **ブラウザ内処理の活用**: 補足として、可能な限りフロント（ユーザ側PC）で処理させてサーバー負荷を下げる、というアプローチも考えられます。Basic Pitchは**ブラウザ内での実行デモ**が示すように、モデルをTensorFlow.jsに移植すればクライアントサイド推論が可能です [49]。これによりMIDI変換はユーザ端末上で完結し、サーバーはMIDI→TAB変換やファイル提供だけ担う、といった構成もあり得ます。ただし音源分離に関しては、現状クライアントサイドで実用的に動かせるモデルは限られます。Open-UnmixのTF.js版デモはありましたが、非常に遅く短い音源に限られる [11] ため、実務ではサーバー実行が現実的で

す。ゆえに**ハイブリッドなアプローチ**として、音源分離はサーバーで行い、得られた各パート音源をBasic Pitchのような軽量モデルでブラウザ側MIDI化する、といった負荷分散も検討できます。ただモデルのロード時間やブラウザのCPU負荷も考慮する必要があり、ユーザ体験とのトレードオフになります。まずはオーソドックスに全処理をサーバーで行い、その結果を受け渡す構成から始め、ボトルネックやサーバーコストに応じて最適化を図るとよいでしょう。

以上、音源分離→MIDI変換→TAB譜生成の各工程について、利用可能なライブラリ/ツールと実装上のポイントを概説しました。**まとめると**:

- 音源分離にはSpleeterやDemucsなど高性能なOSSが揃っており、Pythonで容易に導入できます。それぞれ速度や分離精度に特徴があり、用途に応じて使い分けます [4]。
- 自動採譜にはSpotify Basic Pitchが現状もっともバランス良く有力で、複数楽器・多声音に対応した軽量モデルとして適しています [17] [19]。補助的にオンセット検出や単音ピッチ抽出のツールも利用可能です。
- ギターTAB譜への変換では、弦・フレットの割り当てというギター特有の問題があります。自動化は可能ですが完全な正解を得るのは難しく、**自動提案 + 人手修正**の流れが現実的です。出力フォーマットはMusicXMLやGuitarPro形式を採用し、既存ツール（PyGuitarProなど）でエクスポートすることで、後工程の編集互換性を確保します [42] [43]。
- Webアプリとしては、バックエンドで重い処理を担当させ、フロントエンドは結果表示とユーザ入力を扱う構成を取ります。Django+CeleryやFlaskなどでAPIを構築し、React/Vue等でUIを実装するのが典型例です [47]。音楽特有のUI（波形表示や譜面表示）にはAlphaTab等の専用ライブラリを活用し、ユーザ体験を向上させます。

以上の技術を組み合わせることで、ユーザが音源ファイルをアップロードすると、**各パートの音が分離され、それぞれの音高がMIDIノート列に変換され、さらにギターの運指情報付きのタブ譜ファイルが出力される**、という一連の処理を実現できます。現在のオープンソース技術を活用すればかなりの部分が自動化可能ですが、完全な誤差ゼロの譜面生成は難しいため、**「80%自動化して20%手修正」**をサポートするツールとして設計・実装するのがおすすめです。各工程で紹介したライブラリ（**[Spleeter]** [2], **[Demucs]** [5], **[Basic Pitch]** [17], **[PyGuitarPro]** [41] 等）を組み合わせ、必要に応じてカスタマイズすることで、本要件を満たすWebアプリケーションを構築できるでしょう。

---

1 2 3 13 GitHub - deezer/spleeter: Deezer source separation library including pretrained models.
https://github.com/deezer/spleeter

4 6 15 Demucs vs Spleeter - The Ultimate Guide | Beats To Rap On
https://beatstorapon.com/blog/demucs-vs-spleeter-the-ultimate-guide/

5 7 14 GitHub - facebookresearch/demucs: Code for the paper Hybrid Spectrogram and Waveform Source Separation
https://github.com/facebookresearch/demucs

8 Open-Unmix - Music Source Separation for PyTorch - GitHub
https://github.com/sigsep/open-unmix-pytorch

9 47 48 GitHub - JeffreyCA/spleeter-web: Self-hostable web app for isolating the vocal, accompaniment, bass, and drums of any song. Supports Spleeter, D3Net, Demucs, Tasnet, X-UMX. Built with React and Django.
https://github.com/JeffreyCA/spleeter-web

10 11 ► umx.js - web demo | SigSep
https://sigsep.github.io/open-unmix/js.html

12 Map of Open-Source Source Separation Projects — Open-Source Tools & Data for Music Source Separation
https://source-separation.github.io/tutorial/intro/open_src_projects.html

16 17 18 19 21 22 23 26 27 Meet Basic Pitch: Spotify’s Open Source Audio-to-MIDI Converter | Spotify Engineering
https://engineering.atspotify.com/2022/6/meet-basic-pitch

20 49 Basic Pitch: An open source MIDI converter from Spotify - Demo
https://basicpitch.spotify.com/

24 Onsets and Frames: Dual-Objective Piano Transcription
https://magenta.withgoogle.com/onsets-frames

25 [PDF] ONSETS AND FRAMES: DUAL-OBJECTIVE PIANO TRANSCRIPTION
https://archives.ismir.net/ismir2018/paper/000019.pdf

28 29 GuitarSet - Home
https://guitarset.weebly.com/

30 High Resolution Guitar Transcription via Domain Adaptation - arXiv
https://arxiv.org/html/2402.15258v1

31 Fretting-Transformer: Encoder-Decoder Model for MIDI to Tablature ...
https://arxiv.org/html/2506.14223v1

32 34 35 37 dorienherremans.com
http://dorienherremans.com/sites/default/files/25.pdf

33 38 MIDI-to-Tab: Guitar Tablature Inference via Masked Language Modeling
https://arxiv.org/html/2408.05024v1

36 [PDF] Creating Tablature and Arranging Music for Guitar with Genetic ...
https://arti.franklin.uga.edu/sites/default/files/inline-files/tuohy_daniel.pdf

39 40 Tablature | MusicXML 4.0
https://www.w3.org/2021/06/musicxml40/tutorial/tablature/

41 42 PyGuitarPro — PyGuitarPro 0.9.3 documentation
https://pyguitarpro.readthedocs.io/en/stable/

43 45 46 Introduction | alphaTab
https://alphatab.net/docs/introduction/

44 A list of Sheet Music Display Libraries for Web browsers
https://opensheetmusicdisplay.org/blog/sheet-music-display-libraries-browsers/