AI x MUSIC

目指せAIミュージシャン！

人工知能作曲基礎講座

【VERSUSイノベーションスクール】

GoogleのAI音楽ライブラリーMagentaで体験する未来の音楽制作

第２回

Magenta
1.2.0にバージョンアップ
しました。

https://canplay-music.com/2019/11/28/magenta120/

MagentaでAI作曲

データの種類
MIDIについて
作曲と生成

データの種類について

本講義ではデータという名称で

・生成された音楽データ（MIDIファイル）

・音楽生成時にプログラムに与える音データ（生成コマンドやMIDIファイル）

・学習用の音楽データ（MIDIファイル）

・学習中のデータ（checkpointファイル.tar）

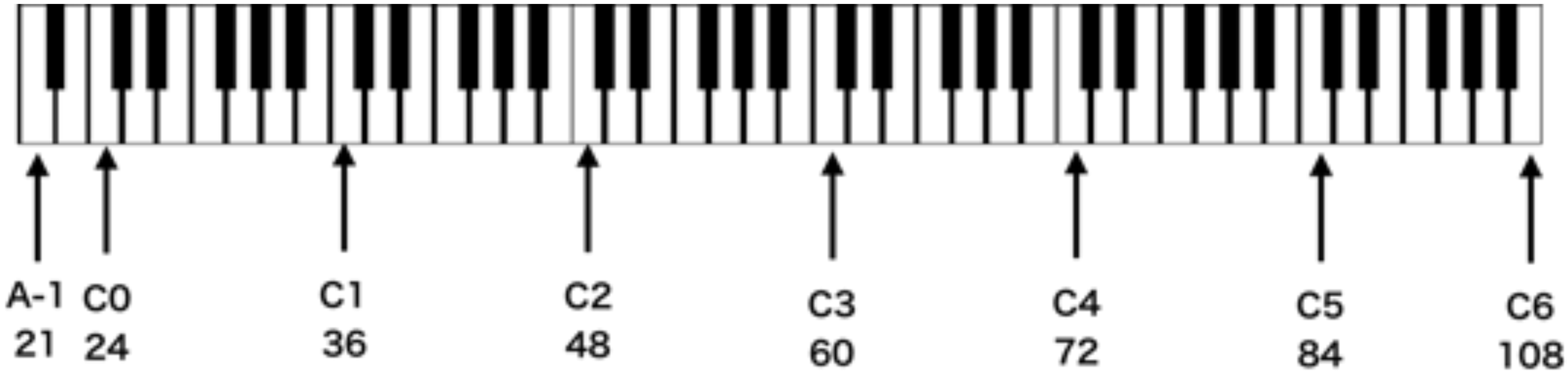
・学習済みデータ（magファイル）

など複数を取り扱います。

同じデータという表現でも用途により異なるものである事を把握の上で、混同しない様にお気を付けください。

MIDIとは、Musical Instruments Digital Interfaceの略です。
電子音楽楽器の共通規格となっており、メーカーを問わず、各電子楽器間で演奏の制御やデータの共有・使用を可能にしています。
様々な機能を持つMIDIですが、本講義で取り上げる内容に関わるのは主に２点です。

１・GM (General MIDI)
ドレミファソラソドなど、音の配列を数値に変換し、デジタルデータとして使用できる様にした規格です。
例えばピアノの丁度中間のあたりのドレミファソラシドだと
ド＝60 レ＝62 ミ=64 ファ=65 ソ=67 ラ=69 シ=71 ド=72
となります。
61や63はシャープやフラットのつく音名、黒鍵の部分です。
MIDIで取り扱えるのは0（C-2）～127（G8）までの128音。
ドのとても低い音～ソのとても高い音までです。
ピアノは21（A-1）～108（C6）までの88鍵ですから、ピアノよりも広い音域をカバーしている事になります。



この１２８の音名の番号をノートナンバーと呼び、MagentaのAI作曲では、生成時に与える音楽データの音名指定に使用します。

２・SMF (Standard MIDI File)

GMのノートナンバー配列などを使用し、再生や演奏に使用できるデータファイルの規格をSMF (Standard MIDI File)と呼びます。

拡張子は.midでMac、Windows、または各電子楽器など、機器を選ばずに利用可能です。

自身で作成する事はもちろん、インターネットで各楽曲のファイルが販売されていたり、カラオケデータとして活用されていたりします。

Magentaで生成される音楽データは、midiファイル、つまりこのSMFです。

それ以外にもMagentaでは、生成時のデータとして使用される他、学習の際の音楽データとしても利用されます。

作曲と生成

AIでの作曲においては、作曲という言葉以外に、生成という耳慣れない言葉が頻繁に出てきます。
生成という言葉は、通常の作曲行為では出てこない言葉ですが、人の知能ではない、人口の知能が音楽制作の一端を担うAI作曲では、この言葉が大きな役割を持ちます。
通常作曲と言えば、人が音楽を作りたいと言う意思を持ち、創作自体も人が行います。
しかしAI作曲では、人が音楽を作りたいと言う意思を持ち、創作は意思のない知能＝AIが行います。
創作が人の意図と離れた機能＝AI作曲では書き出し、になる訳です。

作曲：人の意思、判断や生成、作業を含む楽曲創作行為全般を指す
生成：AIによる音楽の書き出しとそれに伴う作業のみを指す

もう一つ生成の特徴は偶然性です。
生成はコントローラブルではありません。
意図を反映してくれる時もあれば、全く反映しない想定外の作品ができる事も多々あります。
しかしそれこそが生成の面白さです。
作者の想像を超える作品が生み出された時の驚きこそ生成音楽の醍醐味です。
そして今、そんな驚きをもっとも与えてくれる高度な生成を行えるのがAIと言えるでしょう。
作者とリスナーとの驚きの共有、そしてそれが未知の感動に至る事、それこそが生成音楽およびAI作曲の一つのゴールと考えています。

Magentaでできる音楽生成（本講座内で解説予定）

- 単音のシンプルなメロディー生成
- ドラムトラックの生成
- ３パート演奏（メロディー、ベース、ドラム）の楽曲生成
- コード進行に沿ったアドリブメロディー生成
- 単音メロディーにハーモニーを生成
- 表現力豊かなピアノ楽曲の生成
- Ableton Live（または他のDAW）での音楽生成プラグイン活用

単音のシンプルなメロディーを作るMelody RNN

まずはMagentaの中で最も基本となる単音のメロディーを作曲するMelody　RNNについて解説します。
Melody　RNNは後に解説させていただく他のモデルでもメロディー生成の機能として利用されています。
Magentaのメロディー生成のパートを担う主要モデルと言って良いでしょう。
このMelody　RNNの解説でMagentaの基本的な操作の解説を行いますので、少々長い解説となりますが、是非覚えてください。

Melody RNNでの作曲（音楽生成）について特徴をまとめてみます。

特徴１・単音のメロディーの生成（和音は作曲できません）
特徴２・指定された音に続くメロディーを生成（最初にガイドとなる音を最低１音以上与える必要があります）
特徴３・曲の長さを指定できます（１小節１６ステップとなり、８小節なら１２８ステップと指定）
特徴４・コードやスケール（使用する音の配列）を指定する事はできません。
特徴５・テンポの指定ができます
特徴６・数千曲を学習させた学習済みデータが４種類用意されている
特徴７・自分の音楽データを学習させて独自の学習済みデータ作りも可能

Melody RNNでの作曲の手順

手順１・学習済みデータのダウンロード

手順２・生成コマンドを入力し生成

手順３・生成された音楽を視聴確認、エディット

学習済みデータについて

学習済みデータおよび設定は４種類提供されています

Basic:
LSTMを使用した基本的な単音音楽生成モデル
ワンホットエンコーディングで学習データからメロディーを抽出し生成

Lookback:
音楽から、その楽曲のパターンをより簡単に抽出できる様にしたモデル

Attention:
より長い時間（過去の）データにアクセスする事ができるモデルで、時間的なメロディーの整合性をより高める事ができる

Mono
Basicを元にC-2～G8まで（MIDIの全音域）を使用できるモデル
（他のモデルではC２～C5（MIDIの４８～８４）に範囲が制限されています）

各学習済みデータの特徴と音域

名称	特徴	音域
basic	基本的な生成を行う	C2～C5 (MIDI 48～84)
lookback	パターンに沿ったメロディー	C2～C5 (MIDI 48～84)
attention	時間的整合性のあるメロディー	C2～C5 (MIDI 48～84)
mono	MIDIの全音域をカバー	C-2～G8 (MIDI 0～127)

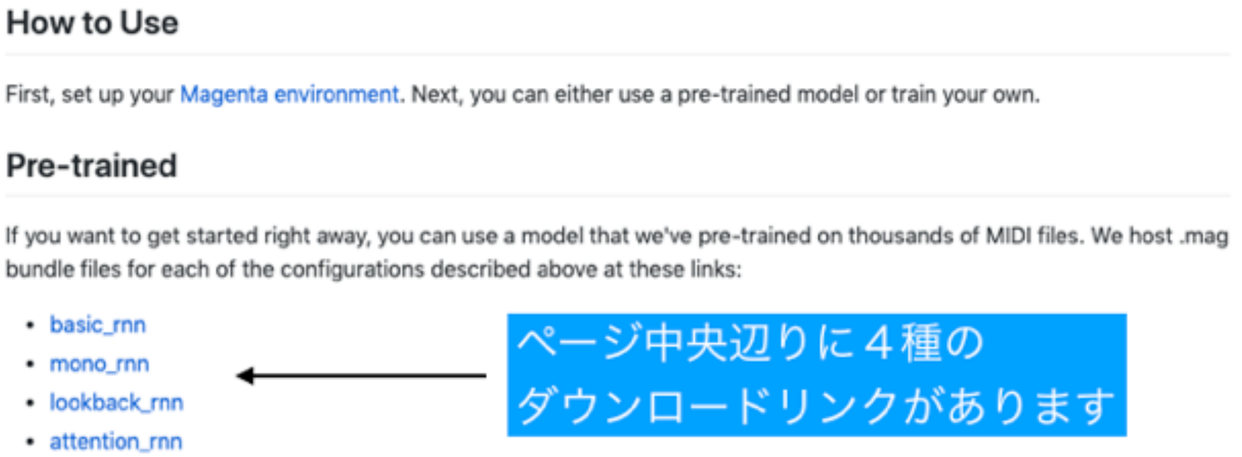
学習済みデータ（.magファイル）のダウンロード

basic_rnn

lookback_rnn

attention_rnn

mono_rnn



各ファイルをMelody RNNのgithubページからダウンロード

https://github.com/tensorflow/magenta/tree/master/magenta/models/melody_rnn

ダウンロードしたらご自身で呼び出しやすいディレクトリーに保存してください。

Melody RNNでの生成

メロディー生成実行コマンド MAC

```
melody_rnn_generate \
--config=４種類の設定から１つ選択 \
--bundle_file=ご自身の.magファイルへのパス \
--output_dir=任意で生成先ディレクトリーを指定 \
--num_outputs=10 \
--num_steps=128 \
--qpm=120.0 \
--primer_melody="[60]”
```

config（設定）と学習済みデータは同じものを選んでください。

basic, lookback, attention, monoの４種類があります

\ は改行できないコードを改行して表示するためのものです。

\ はMAC、Windowsは ^ を使用する。

–もないとエラーがでるので忘れない事

PDFのコピペはエラーが出る場合もあるので下記リンクのサンプルコードも参考に

https://github.com/tensorflow/magenta/tree/master/magenta/models/melody_rnn

メロディー生成実行コマンド Windows

```
melody_rnn_generate ^
--config=４種類の設定から１つ選択 ^
--bundle_file=ご自身の.magファイルへのパス ^
--output_dir=任意で生成先ディレクトリーを指定 ^
--num_outputs=10 ^
--num_steps=128 ^
--qpm=120.0 ^
--primer_melody="[60]”
```

config（設定）と学習済みデータは同じものを選んでください。

basic, lookback, attention, monoの４種類があります

^ は改行できないコードを改行して表示するためのものです。

–もないとエラーがでるので忘れない事

Windowsでは実行コマンドをコピペしても１行になります。

PDFのコピペはエラーが出る場合もあるので下記リンクのサンプルコードも参考に

https://github.com/tensorflow/magenta/tree/master/magenta/models/melody_rnn

＜メロディー生成の実行命令コマンド＞

```
melody_rnn_generate \
```

こちらがメロディー生成実行を命令するコマンドです。
この生成を開始するという命令を受け、以下に続く各パラメーターの設定に従ったメロディーが生成される訳です。
なお、末尾の\（バックスラッシュ）は音楽生成内容には関係ないのですが、特別な意味があり記述されています。
これは本来改行できないコードを改行するための特殊な文字です。
実はこの複数行に分けて記述されている今回の生成コマンドのコードは、本来改行できない１行のコードなのです。
しかしそれでは横幅が長く、画面からはみ出る長さになるので見づらいですね。
そんな時に\を末尾に記述すると、コードの途中でも繋がりを維持したまま次の行に改行して続きを記述する事ができます。

＜モデルの設定＞

--config=<'basic_rnn', ‘mono_rnn’, 'lookback_rnn', or ‘attention_rnn’,のどれかを選択>

読み込む学習済みデータと同じ名前を指定します。
違う設定を指定してもエラーにはなりませんが、学習データに合わせた最適な設定がプログラムしてあるので合わせた方が良い結果を得る事ができるはずです。
これでそれぞれのモデルの設定が完了となります。
例えばbasic_rnnを学習済みデータとして選択したのであれば、

—config=basic_rnn

となります。

＜学習済みデータの指定＞

```
--bundle_file=ご自分の.magファイルへのパス
```

ダウンロードした学習済みデータ（.magファイル）のパスを指定します。

例えばbasic_rnn.magファイルを、Rootディレクトリーの直下にあるDownloadsフォルダーに保管しているとすれば

```
--bundle_file=/mac-pro/Your-name/Downloads/basic_rnn.mag
```

となります。

前述した学習済みデータの保管先を呼び出しやすいディレクトリーにするのはこのためです。

今後他のモデルでも学習済みデータはたくさんダウンロードし保管する事になるので、専用のフォルダーを作って管理しても良いかもしれません。

＜保存先ディレクトリーの指定＞

```
--output_dir=/tmp/melody_rnn/generated
```

生成したMIDIファイルの書き出し先ディレクトリーを指定します。

初期値はtempフォルダー（一時保管フォルダー）になる様プログラムされているため、何も指定しない場合は自動的にtempフォルダーが作成されそこに保管されます。

しかし保管先がわかりづらいですし、せっかく任意で好きな保管先を指定できるのですから管理しやすいディレクトリーを指定するのが良いでしょう。

今後たくさんの音楽を生成する事になるはずですので専用のフォルダーを作成するのもお勧めです。

＜生成MIDIファイル数の指定＞

```
--num_outputs=10
```

このコマンドで生成MIDIファイル数を指定できます。

初期設定は１０になっているので何も指定しない場合は１０曲生成されます。

筆者の実験では１度に最高５００曲ほどの生成を行なった事があります。

さすがにそれ以上の曲数の生成は試していませんが、１曲～何万曲でも好きなだけ生成が行えるはずです。

＜曲の小節数・長さの指定＞

```
--num_steps=128
```

Melody RNNではステップ数で曲の小節数と、それによって決まる長さの指定をします。
１ステップが１６分音符、１小節が１６ステップと非常にわかりやすくなっています。
初期設定は１２８ステップで、これは８小節です。
１６小節なら２５６ステップ、４８小節なら７８４ステップとなります。
なお、最低でも１小節以上の音楽生成という設定を満たす必要があるのでステップ数は１６以上の指定が必須です。
またステップの単位の都合上３連やスイング、３２分音符などの音符を含む楽曲の生成はできません。
タイミングはジャストです。
揺れやノリ、合わせてステップとは直接関係ないですが、音の強弱（ベロシティ）といった表現もされませんので覚えておいていただければと思います。

＜楽曲のテンポの指定＞

```
--qpm=120.0
```

楽曲のテンポはqpmというパラメーターで指定します。

音楽では通常BPM（Beat　Per　Minutes）でテンポの指定をしますが、名前がqpmとなっただけで通常通り１分間になる四分音符の回数を表します。

初期設定は１２０ですので、何も指定しなければ１２０で生成されます。

なおfloat型（少数型）ですので少数での指定が可能です。

少数以下は必須ではないので整数、例えば１２０などでも問題なく生成できますが、細かい指定が必要な場合は少数で指定できる事を覚えておいてください。

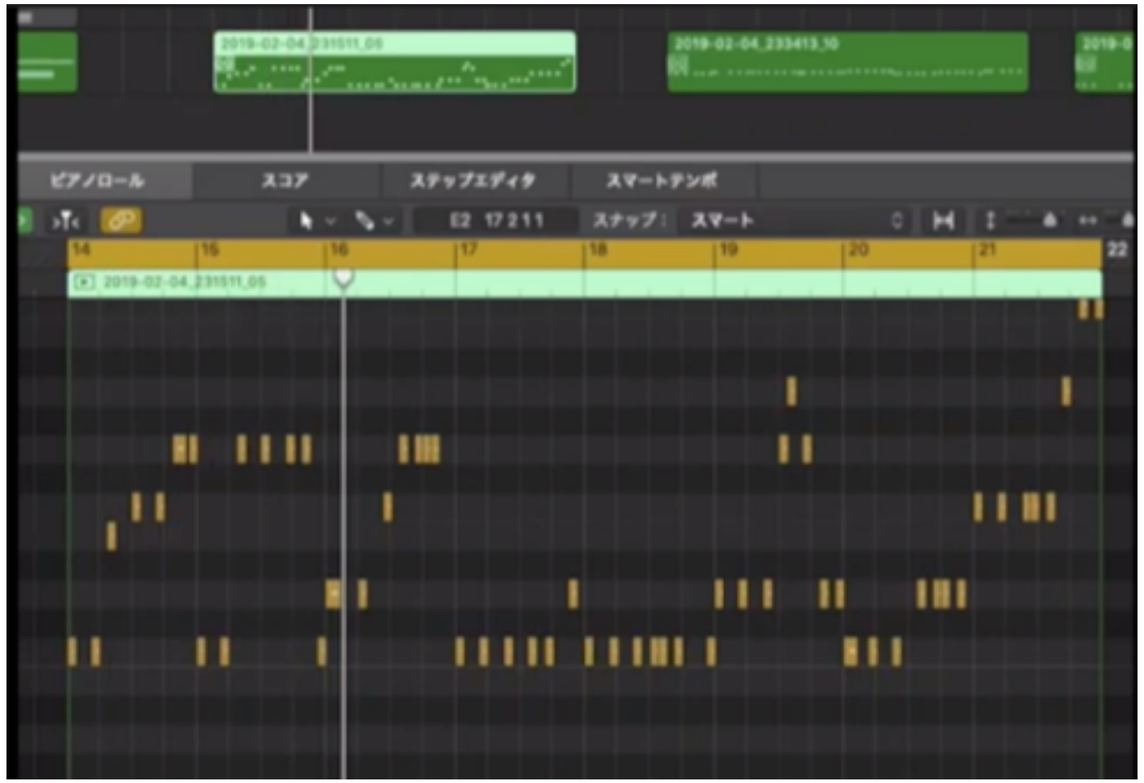
＜生成用の音楽データを与えます＞

```
--primer_melody="[60]”
```

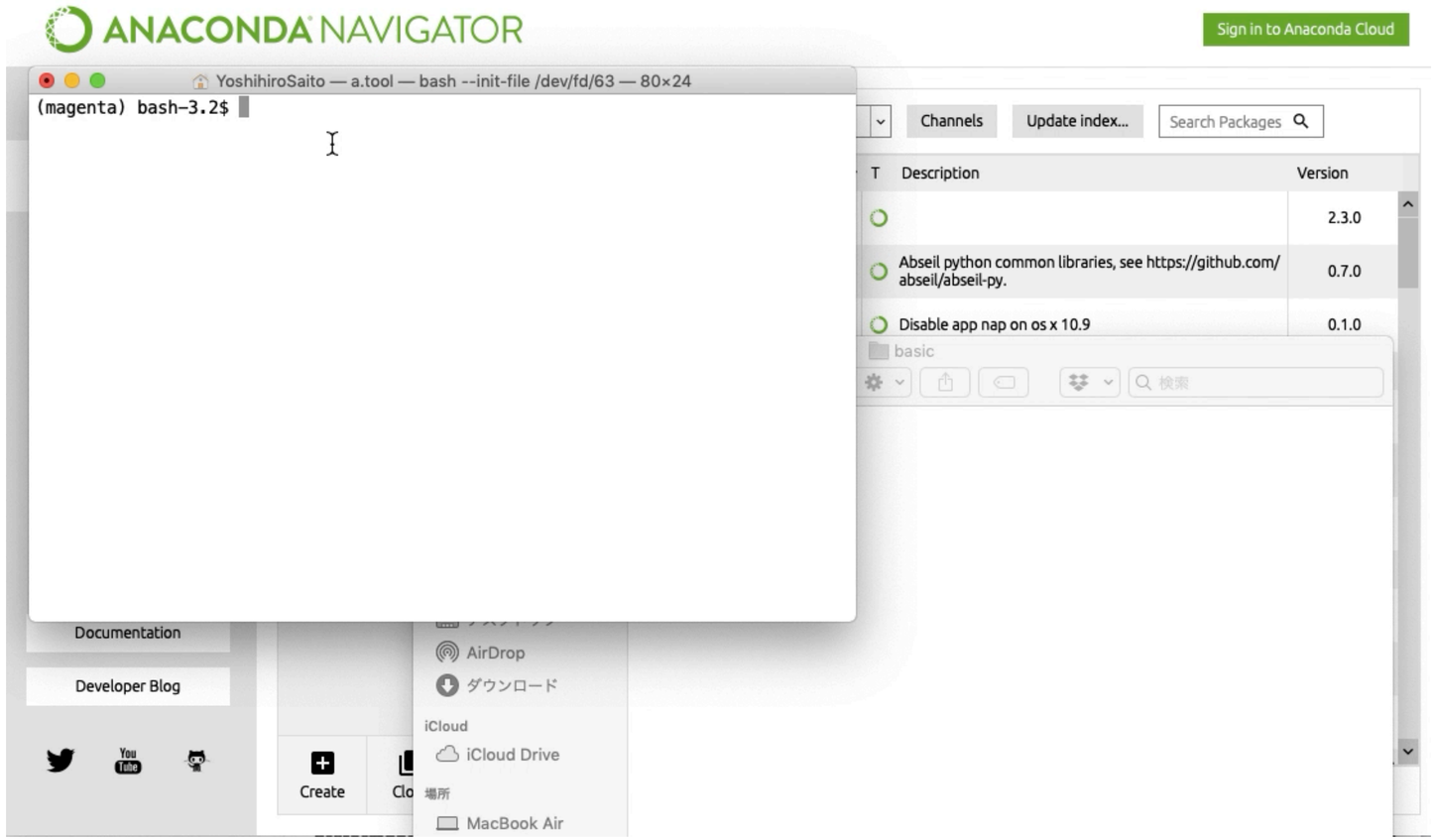
特徴２で解説した通り、Melody　RNNは、生成時に与えられた音に続く音を予測する事でメロディーを生成します。
そのため最低１音以上の音符を指定して与える必要があります。
そのパラメーターとなるのがprimer_melodyです。
全てMIDIのノートナンバーで数値として指定します。
記述の方法は、リスト形式で”[ ]”で囲んだ中に該当ノートナンバーの数値を記述する事で行います。
初期値は60（C３）になっており、何も指定しなければC3から始まるメロディーが生成されます。
C3はちょうピアノの鍵盤の真ん中あたりのドの音です。
ミの音から始めたければ64、ソ＃からはじめたければ68など、任意の音を数値で指定できます。

生成した音楽データの試聴とエディット

音楽生成できたらお持ちのDAW（コンピューター音楽制作ソフト）で視聴してみましょう！



MagentaのMelody RNNで自動作曲してみましょう



※資料では動画は再生できません

Melody RNN でMagentaの実践的生成手法を解説

primer_melody

Melody　RNN（や他の多くのMagentaモデル）は、生成楽曲の開始に使用する音符データを１音以上与える必要があります。
その時に使用するのはprimer_melodyです。
単音、C3（ド）の場合のコードは

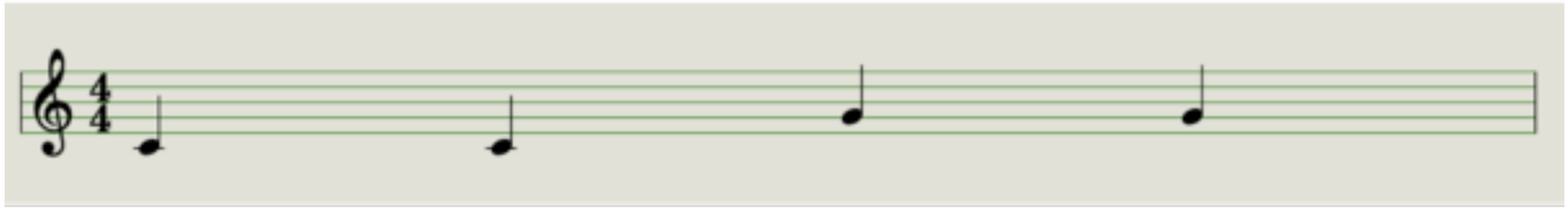
--primer_melody=“[60]”

でした。
複数の音が連なった音符データ、つまりメロディーの場合はコンマで区切って指定します。

--primer_melody="[60, 60, 67, 67]”

上記は60がC3（ド）で67はG３（ソ）です。
ドドソソ、きらきら星の始まりと一緒ですね。
しかし実はきらきら星と同じ様には発音されません。
というのもMelody　RNNでの音価（音の長さ）は何も指定していないと１ステップ分、つまり１６分音符になるのです。
きらきら星は４分音符ですので音価は４倍です。

４分音符のドドソソ

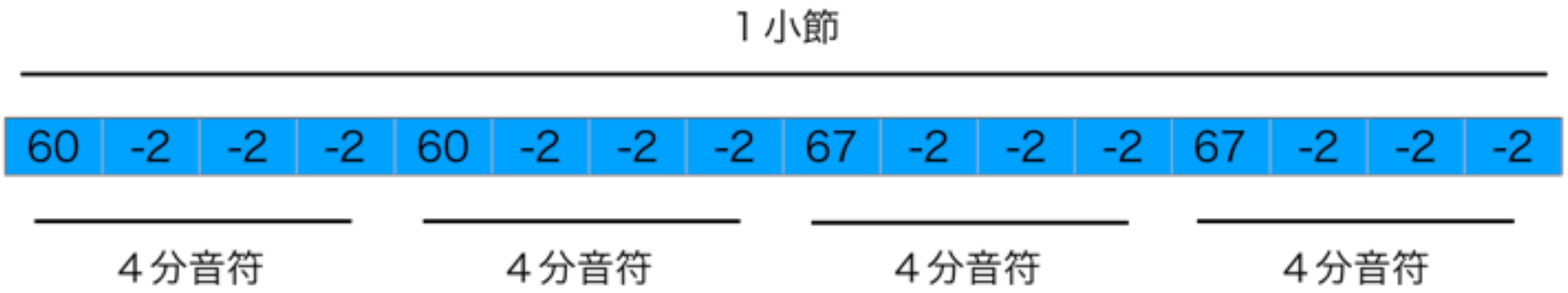


１６分音符のドドソソ

Melody RNNで４分音符の指定するのはどうすれば良いのでしょうか？
次のページののコードをご覧ください。

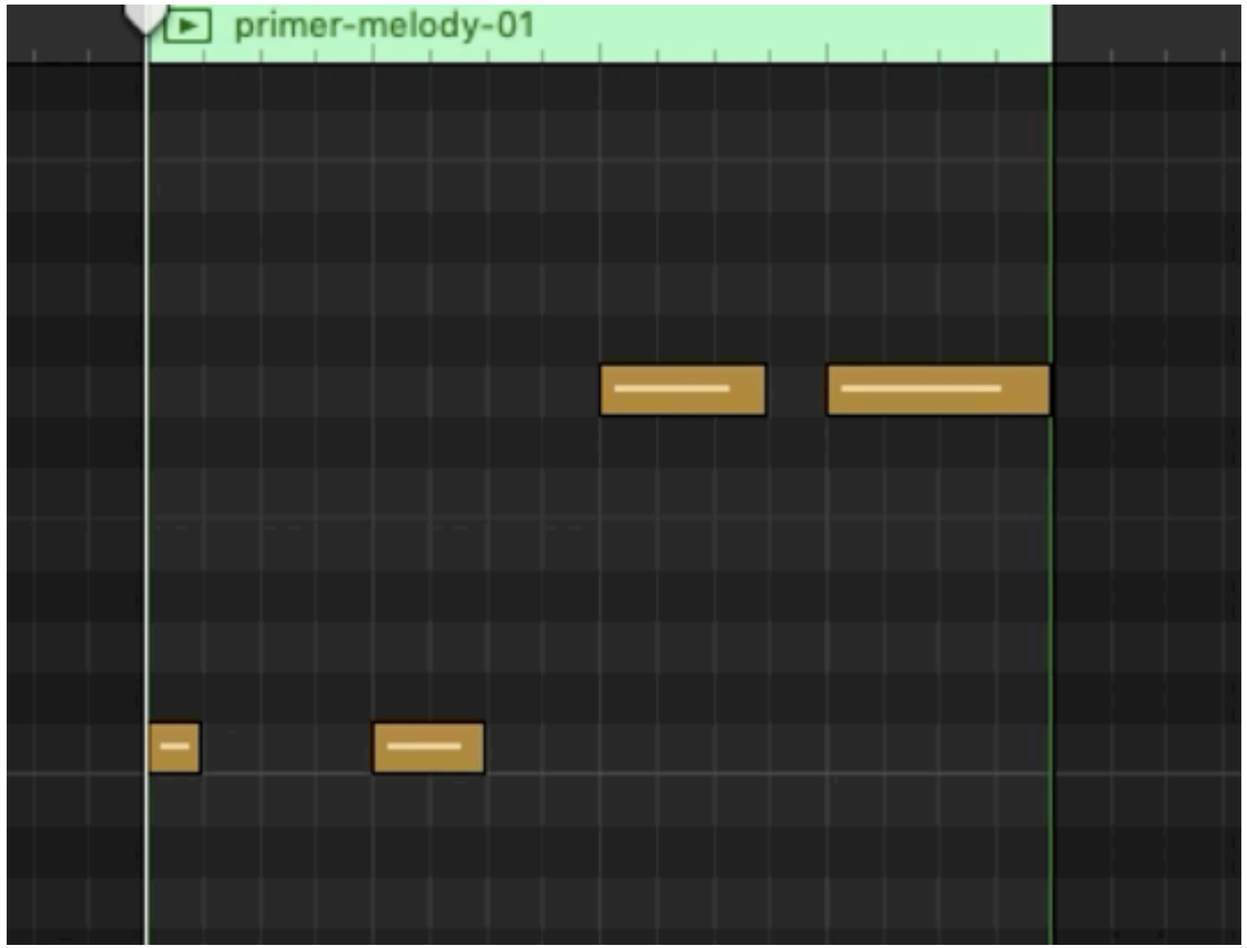
--primer_melody="[60, -2, -2, -2, 60, -2, -2, -2, 67, -2, -2, -2, 67, -2, -2, -2]”

－２が数値データとして登場しましたが、これは何もしないという実行命令になります。
この場合の何もしないは、発音されている音がある場合は、その音に何も操作を加えない＝音をそのまま伸ばす、音がない場合は何も行わない＝発音しない、という実行を行います。



これできらきら星の最初のドドソソ４音をきちんとデータとして与える方法がわかりました。

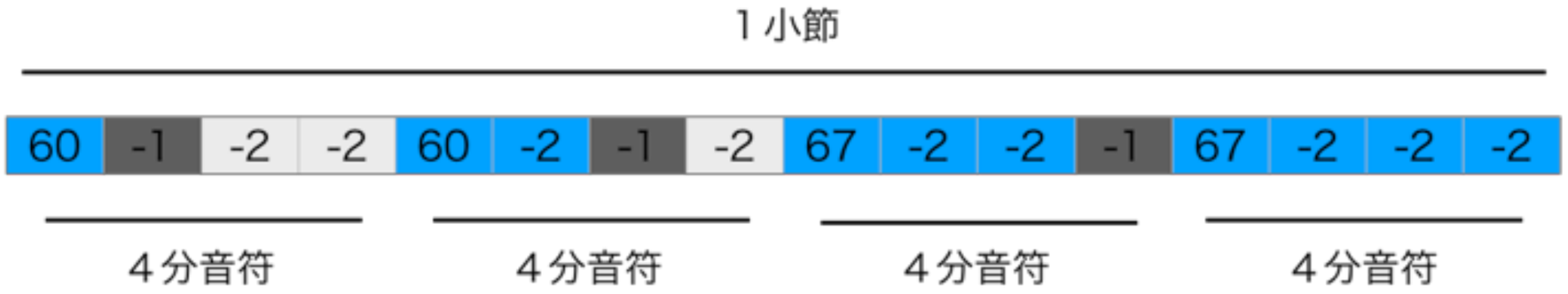
音価の変化や、休符（発音しない）を含むメロディーの場合はどの様に行えば良いのでしょうか？



第１拍：１６分音符（１ステップ）のドの音、その後付点８分休符（３ステップ）
第２拍：８分音符（２ステップ）のドの音、その後８分休符（２ステップ）
第３拍：付点８分音符（３ステップ）のソの音、その後１６分休符（１ステップ）
第４拍：４分音符（４ステップ）のソの音

このメロディーを指定するために必要なのは音を止める命令コマンドです。
そこで－１が使用されます。
図4.7のメロディーを表したコードは

--primer_melody="[60, -1, -2, -2, 60, -2, -1, -2, 67, -2, -2, -1, 67, -2, -2, -2]”



-1の部分で発音が止められており-2では発音も休符もそのまま実行されています。
青の色の部分が発音されている長さとなります。
0~127がC-2～G8までのノートナンバーとなりますのでまとめると

0~127： 該当するMIDIノート。0＝C-2～127=G8（例：60はC3です）
-1：ノートオフ（音を止めます。）
-2：イベントなし（発音はそのまま、休符もそのまま実行）

です。
-2、-1、0~127までを使用して、１６分音符を元にしたあらゆるメロディーの指定が可能です。

primer_midi

Melody RNNでは、生成時に与える音符データを数値だけではなくMIDIファイルで与える事もできます。
指定の方法は非常に簡単で、使用したいmidiファイルのパスを指定するだけです。
例えばRootディレクトリー直下のダウンロードフォルダーに保管してあるsong001.midを使用する場合

--primer_midi=Users/Downloads/song001.mid（Mac or Ubuntu）

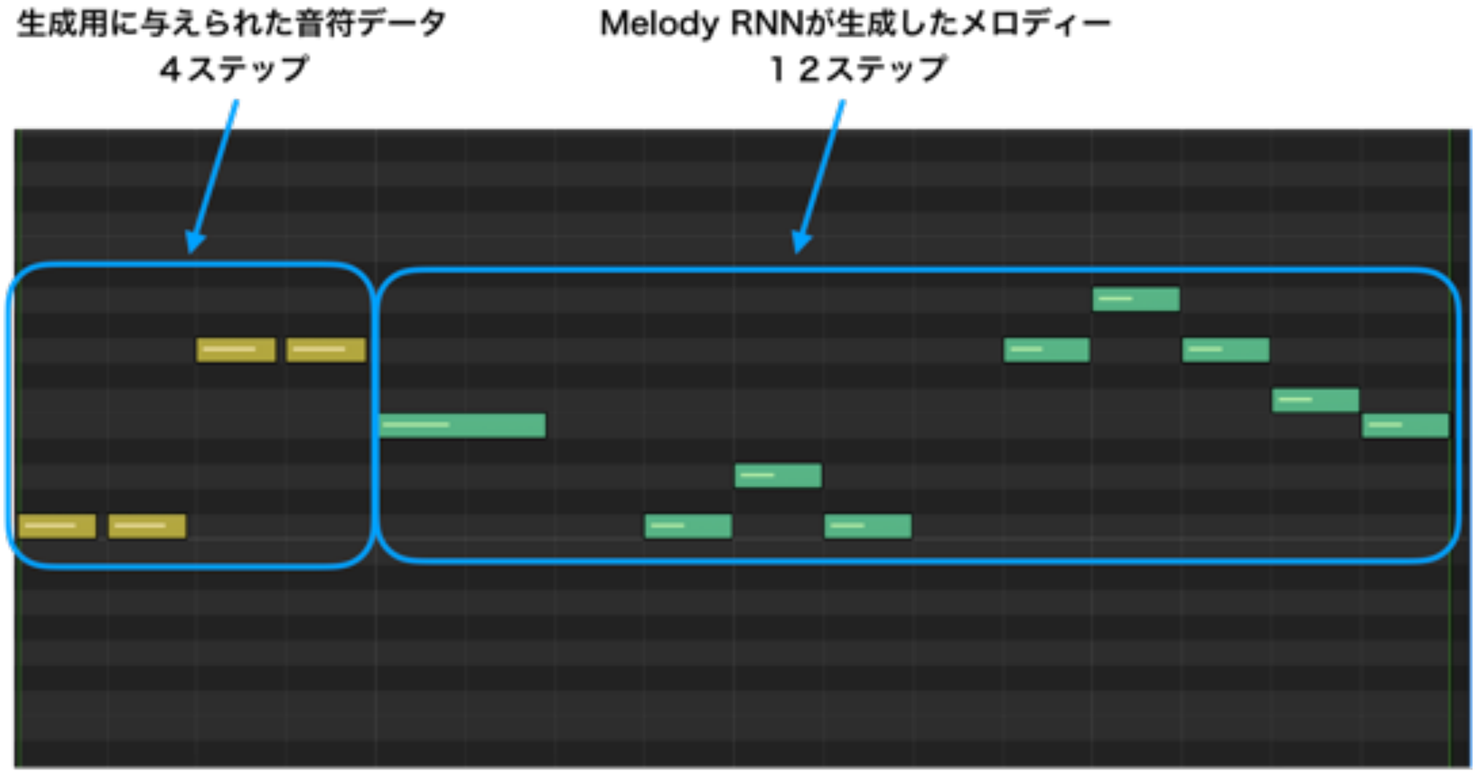
や

--primer_midi=Users¥user-name¥downloads¥song001.mid（Windows）

などの様になります。

primer_melodyも含め、生成時に与える音楽データで留意すべき事、共通のルールは、その与えられたデータに続く音符が選択されメロディが作られるという点です。
チューリップのメロディー１曲分最後までをmidiデータとして与えた場合、Melody RNNが生成するのは、チューリップの既存メロディー終了後の続きのメロディーという事になります。
もしステップ数を１６＝１小節のメロディーを生成する場合、primer_midi（primer_melodyも同様）で４ステップの生成音楽データを与えると、Melody RNNが生成するのは１２ステップです。
合わせて１６ステップ＝１小節のメロディーとなります。

生成曲のステップ数は、与えた音符データよりも長いものにする必要があります。
もし短い場合は、与えた音符データしか書き出されず、一切の生成が行われない事になります。



１６ステップの生成曲

他のモデルの使用方法

これまでbasicの使用のみ解説してきましたが、Melody　RNNには他に３つの学習済みデータがあり、合計すると４つのモデルを使用する事ができます。
（Magentaでは、学習済みデータを設定として利用し、音楽生成AIとして使用できるようにしたものがモデルとお考えください。）
各学習済みデータについては解説済みですが、それぞれの生成時の読み込み方と設定について説明します。
と言っても、方法は非常に簡単で

・bundle-fileにダウンロードした他の学習済みデータのパスを指定
・bundle-fileに合わせてconfigの名前を変更する

以上の２点の変更だけです。

Melody RNN全４つのモデルの記述方法

```
–config=basic_rnn \
--bundle_file= ご自身のディレクトリー/basic_rnn.mag \

--config=lookback_rnn \
--bundle_file= ご自身のディレクトリー/lookback_rnn.mag \

--config=attention_rnn \
--bundle_file= ご自身のディレクトリー/attention_rnn.mag \

--config=mono_rnn \
--bundle_file= ご自身のディレクトリー/mono_rnn.mag \
```

Monoでベース
のフレーズ生成

Melody RNN

LSTM（Long Short Term Memory）を使用した、言語モデル応用の単音自動作曲プログラム

Basic
Lookback
Attention

生成される音域がC2~C5に設定されており範囲外のメロディー（フレーズ）生成ができない。

C-2～G8まで（MIDIの全音域）を使用できる新しいモデルが

Mono

mono_rnnを使用してベースフレーズ生成

```
melody_rnn_generate \
--config=mono_rnn \
--bundle_file=絶対パス/mono_rnn.mag \
--output_dir=自身の生成したい希望のディレクトリーを指定 \
--num_outputs=5 \ #生成したい曲数
--num_steps=16 \ #生成したいフレーズの長さ16ステップで1小節
--primer_melody="[36]"
```

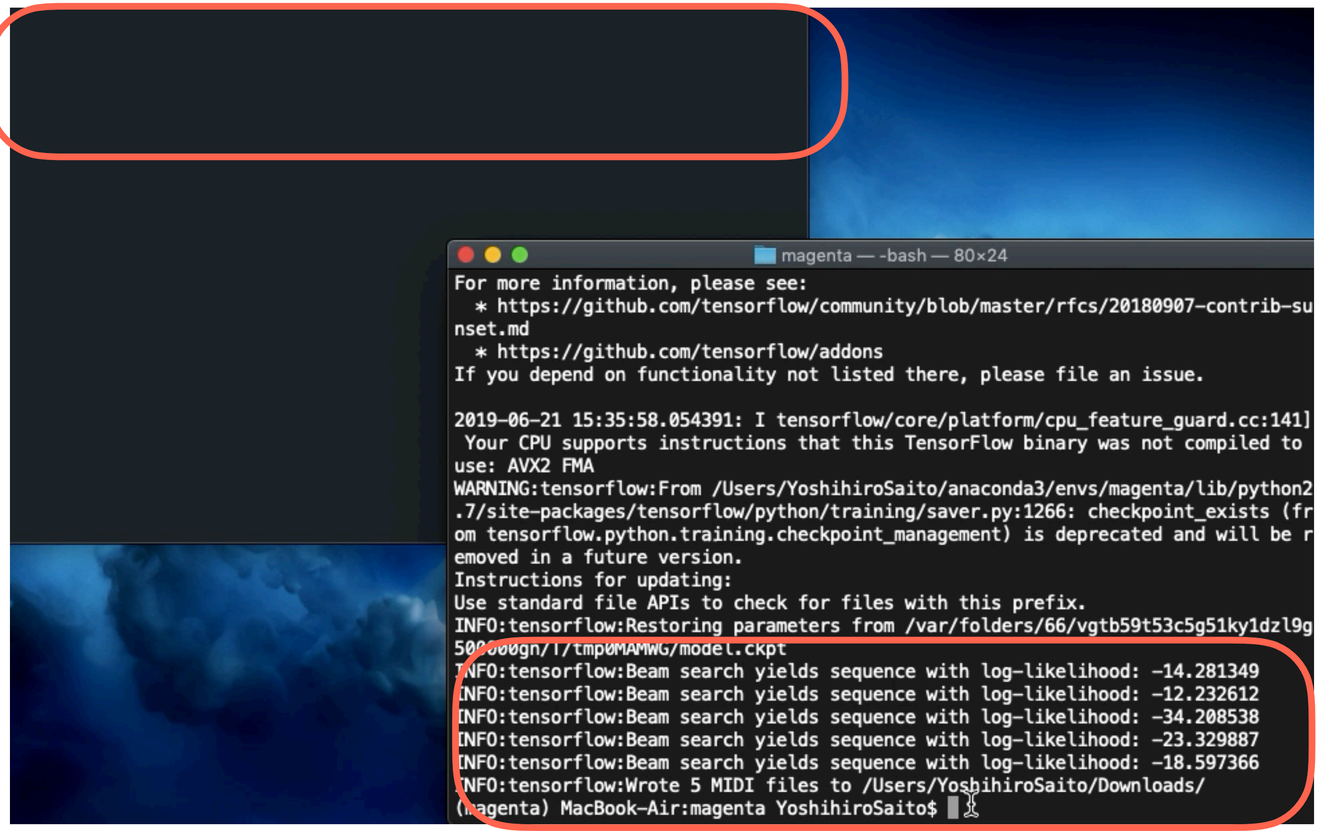
バックスラッシュは本来改行できないコードを改行するためにつける
windowsは^

ベースのフレーズ生成のためPrimer melodyを36（C1）に設定（他の音高でも構いません）
primer_midiでMIDIファイルも使用できます

mono_rnnでベース
のフレーズ生成

mono_rnnでベースのフレーズ生成

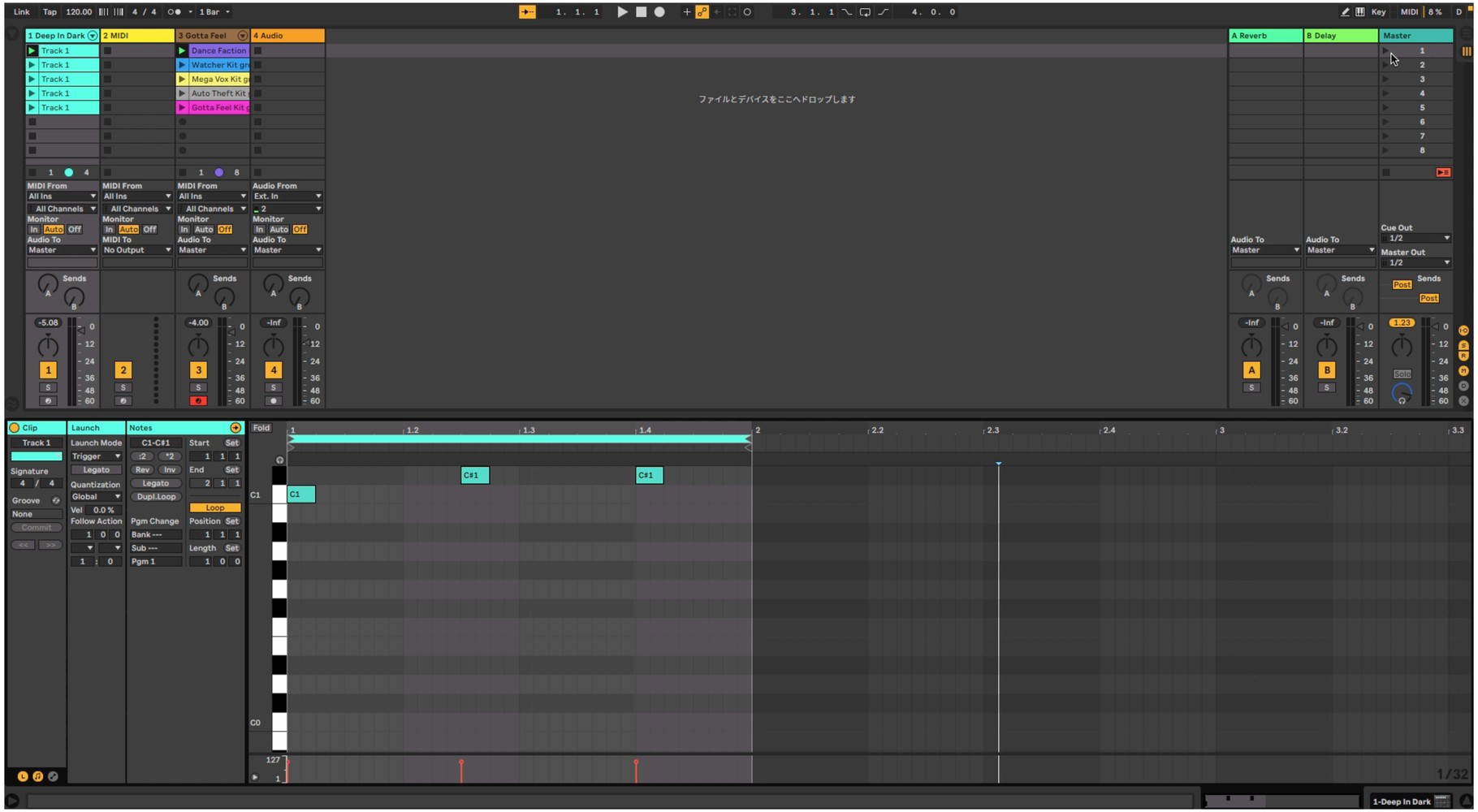
ここにMIDIファイル
が生成されます



生成コマンドの実行

※資料では動画は再生できません

mono_rnnでベースのフレーズ生成



※資料では動画は再生できません

生成したベースフレーズにドラムを重ねて再生

Magentaでできる音楽生成（本講座内で解説予定）

・単音のシンプルなメロディー生成
・ドラムトラックの生成
・３パート演奏（メロディー、ベース、ドラム）の楽曲生成
・コード進行に沿ったアドリブメロディー生成
・単音メロディーにハーモニーを生成
・表現力豊かなピアノ楽曲の生成
・Ableton Live（または他のDAW）での音楽生成プラグイン活用

Drums RNN

ドラム演奏の作曲を行う Drums RNN

Magentaでは音楽に、特に現代の音楽にとって非常に重要なパートであるドラム演奏の作曲に特化したモデルもあります。
それがDrums RNNと命名されているモデルです。
Drums　RNNは、他のMagentaモデルのドラム演奏作成用にも使用されており、Magentaのドラム作曲担当と言える存在です。
ここではドラム演奏作曲用の様々な特徴を持つDrums　RNNについて解説させていただきます。

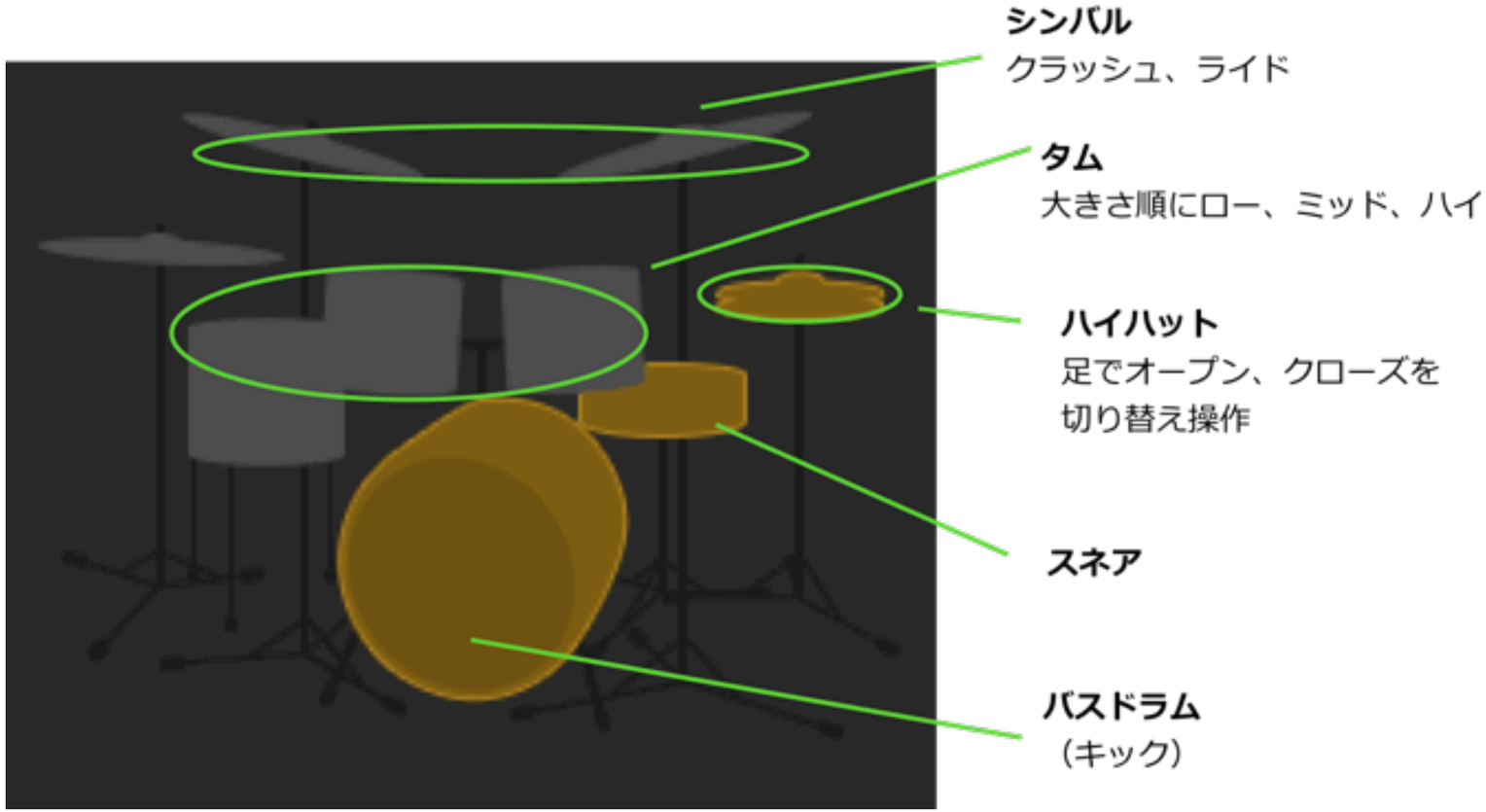
Drums RNNの特徴

特徴１・使用できる音はドラムセット用のノートナンバーに指定された9音のみ

特徴２・複数の音を同時に発音可能

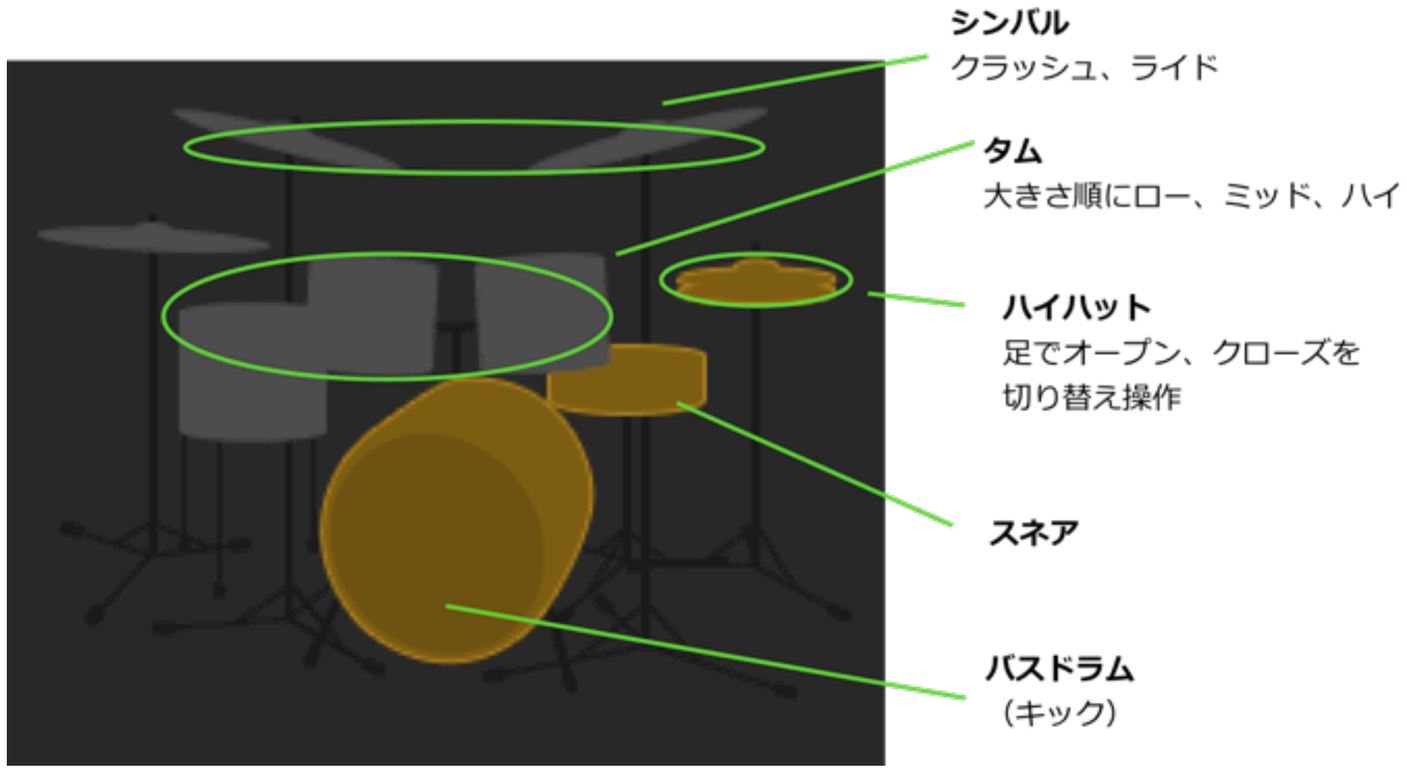
特徴３・学習データは１種類のみ提供（数千曲のドラム演奏を学習させたもの）

特徴１・使用できる音はドラムセット用のノートナンバーに指定された9音のみ



通常の楽曲で特にメインとなって使用されるのが図で個別に名称を記載した９種類です。
Drums RNNでは、この９種類のドラム音のみを使用する事ができます。
他のドラム楽器は使用できません。

特徴１・使用できる音はドラムセット用のノートナンバーに指定された9音のみ

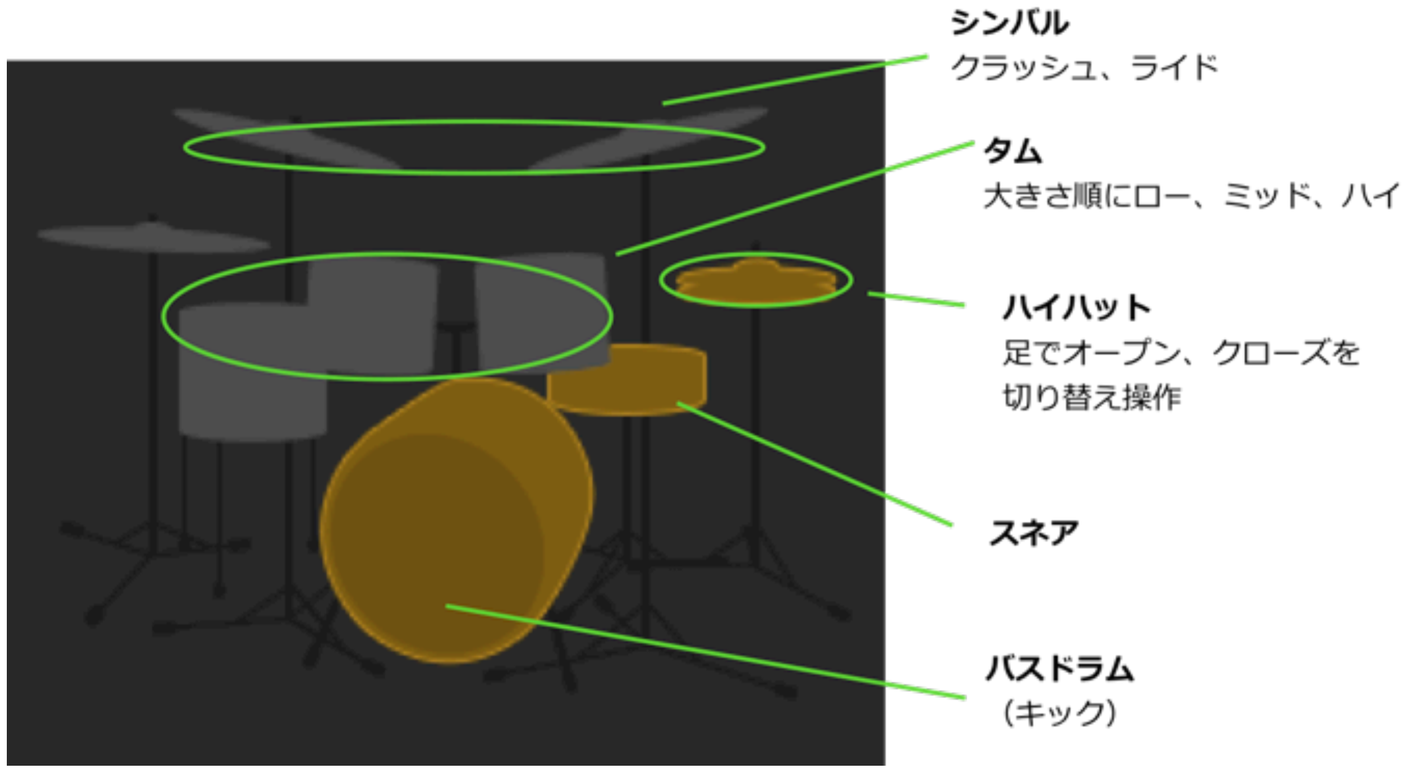


- バスドラム

キックともよびます。大太鼓です。

低音を担当するメインドラムの一つで主に小節の始まりなど、リズムのアクセント、開始時に使用します。

特徴１・使用できる音はドラムセット用のノートナンバーに指定された9音のみ

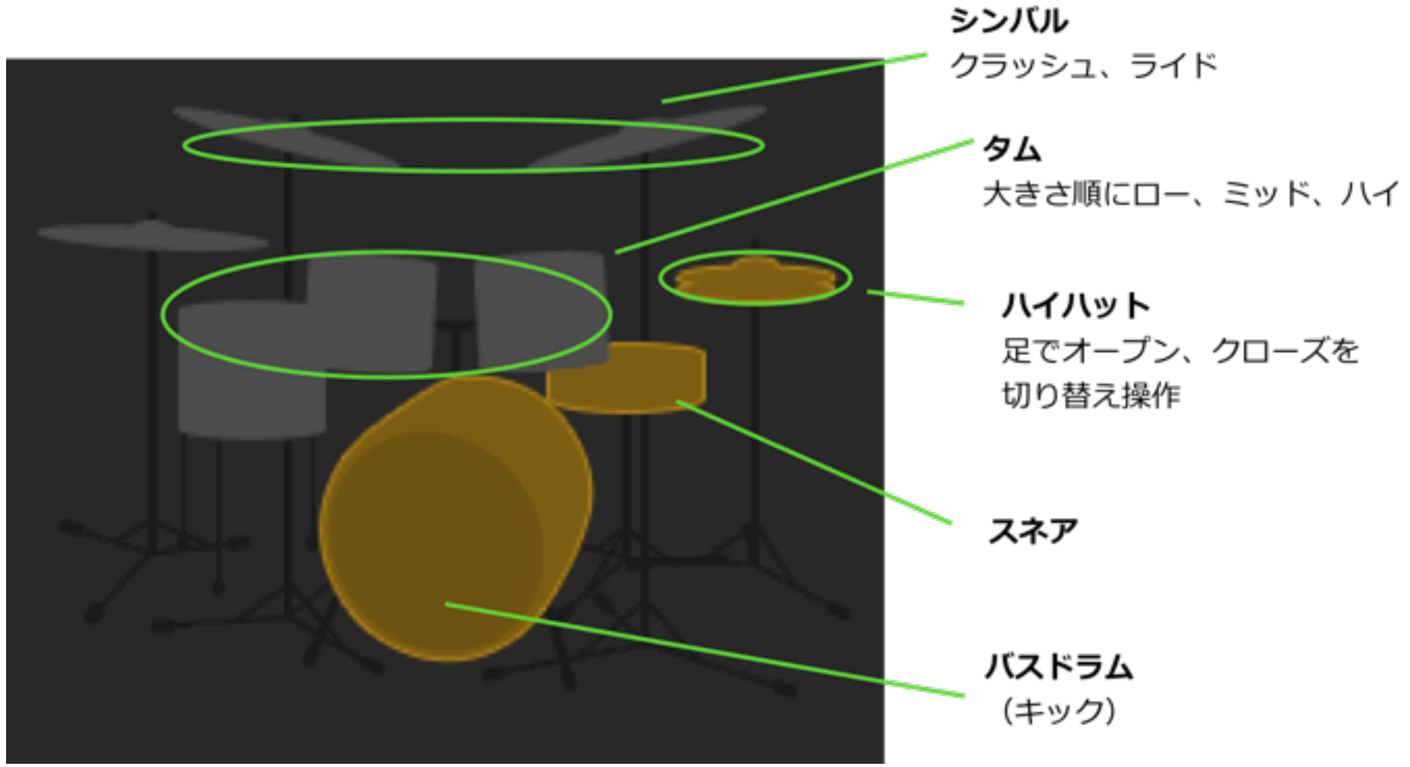


- スネアドラム

小太鼓です。中音を担当するメインドラムの一つです。

主に２、４拍目などバスドラムと共に強いアクセントの部分に使用します。

特徴１・使用できる音はドラムセット用のノートナンバーに指定された9音のみ



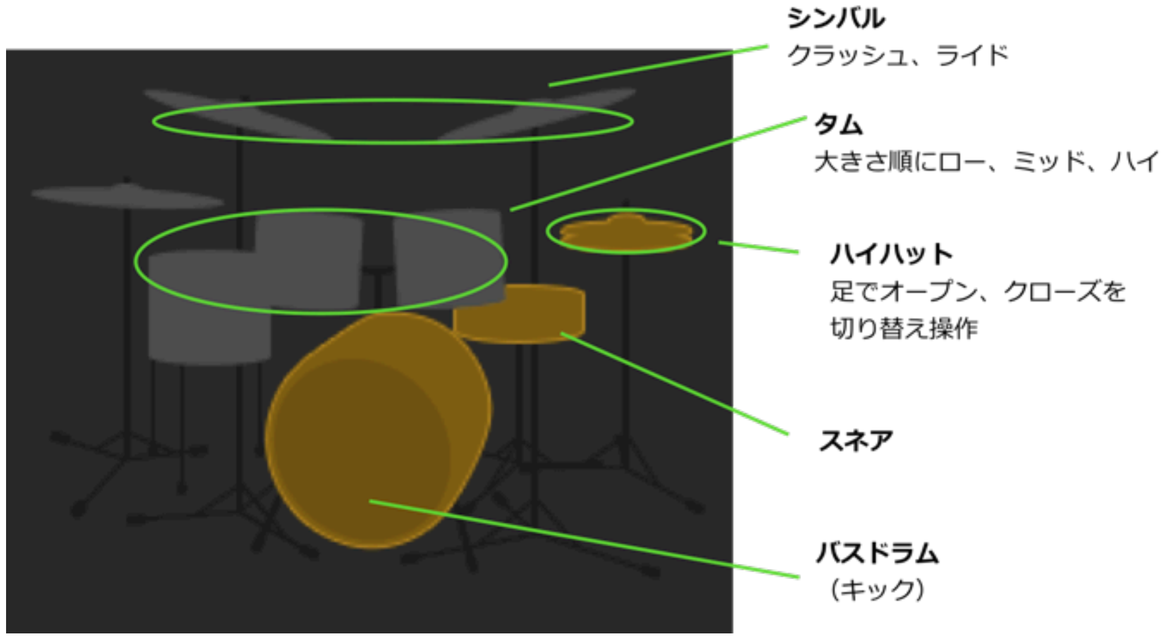
- ハイハット

ハイハットは金属系のドラム楽器でもっとも通常利用されるもので、高音域を担当するメインドラムの一つです。

比較的小さな音量で細かいリズムを刻む事が多いです。

２枚の金属板が重なる様な作りになっており、足の操作で閉めて短い音にしたり（クローズ ）、開いた状態で長めの音（オープン）にしてノリを生み出します。

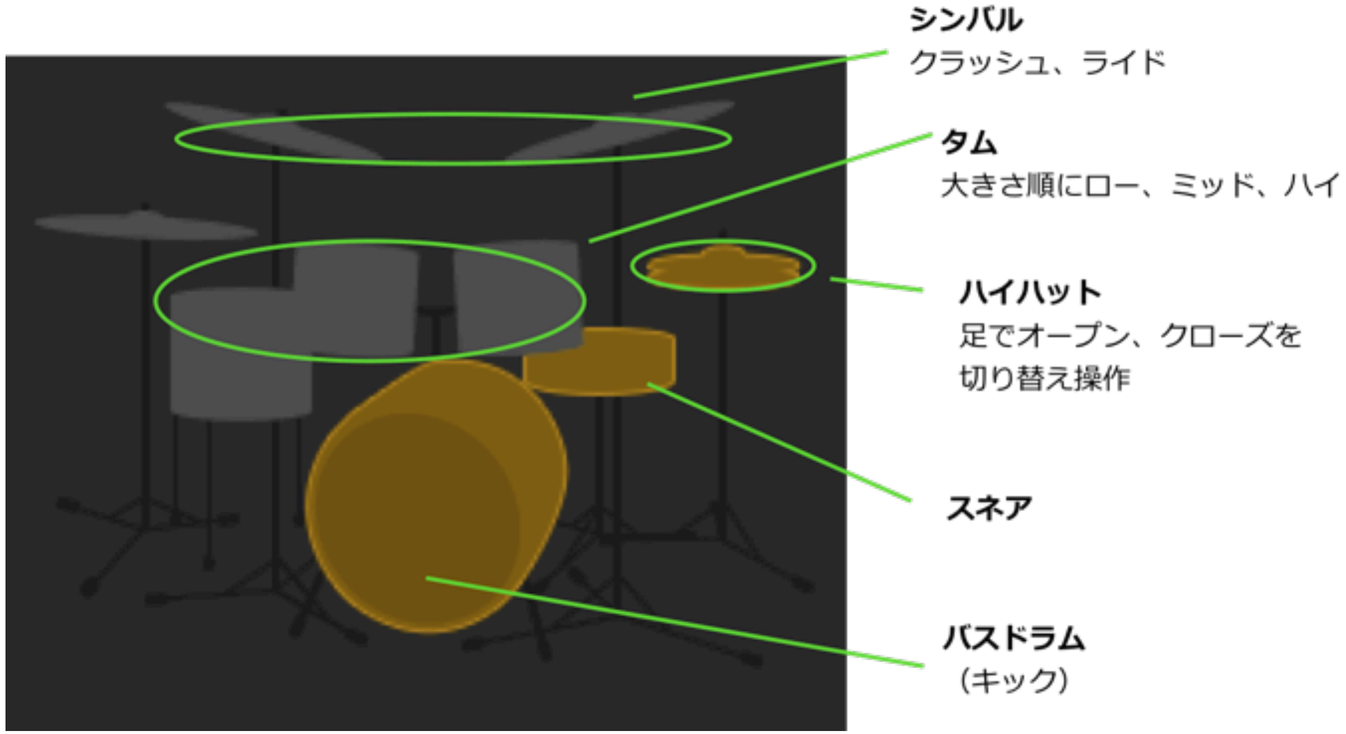
特徴１・使用できる音はドラムセット用のノートナンバーに指定された9音のみ



- タム

バスドラムとスネアドラムの中間の大きさの太鼓で、使用するドラマーによって３個～１０個以上をセットにして使用する場合もあります。
特徴は比較的ピッチ（音程）が明確な事で、主にフィルインやオカズと呼ばれるリズムのバリエーションを作るために使用されます。
フロアタム、ロータム、ハイタムなど呼び方も色々ありますが、MIDIでは５個が使用され、その中でもMagentaで使用されているのはハイ、ミッド、ローの３種類です。

特徴１・使用できる音はドラムセット用のノートナンバーに指定された9音のみ



- シンバル

高い音域で大きな音の金属音を発する打楽器です。
クラッシュシンバルは、より薄く、口径も小さく、小節の始まり等で使用されます。
ライドシンバルは、より厚く、口径も大きく、リズムのアクセントなどで使用される事が多いです。

Drums RNNのノートナンバー

Instrument	MIDI ノートナンバー
Bass drum/Snare drum	36/38 （C1/D1)
Closed/Open hi-hat	42/46 (F#1/A#1)
Low/Mid/High tom	45/48/50 (A1/C2/D2)
Crash/Ride cymbal	49/51 (C#2/D#2)

特徴２・複数の音を同時に発音可能
ドラムというのは、バスドラム（やスネアドラム）、ハイハット、シンバルなど、複数のドラム楽器が同時に重なって発音される事によって演奏される事が多いです。
前述した通り、Drums RNNでは複数のドラム音が重なった発音も、単音のデータと同じ１つのデータとして取り扱う事で生成を実現しています。
図は実際に生成したドラム演奏曲です。
単音しか生成できないMelody RNNに対し、Drums RNNでは、複数の音が重なるリアルなドラム演奏の生成を実現しています。

特徴３・学習データは１種類のみ提供（数千曲のドラム演奏を学習させたもの）
用途の異なる複数の学習済みデータが用意されていたMelody RNNと異なり、Drums RNNではdrum_kitと命名された１種類のみ学習済みデータが用意されています。
こちらも数千曲のドラム演奏のMIDIファイルを学習しているとの事です。
自身で膨大なデータを作成し、学習をさせなくても、この学習済みモデルを使用すればすぐにAIドラム作曲を実践していただけます。
ただし、Melody RNNと異なり、学習済みデータ１つに対し、config（設定）は２種類用意されています。
そのためMelody RNNと違い、学習済みデータと設定名を合わせない生成も実践できます。

Drums RNN
の生成方法

Drums RNNの作曲 ドラム演奏の生成方法解説
Drums RNNの音楽生成方法は基本的に
Melody RNNと同様です。

手順１・学習済みデータのダウンロード
手順２・生成コマンドを入力し生成

まず学習済みデータのダウンロードです。
学習済みデータはMagentaのGithubレポジトリーのDrums　RNNページ中央辺りに、drum_kit
という名称でダウンロードリンクがあります。

https://github.com/tensorflow/magenta/tree/master/magenta/models/drums_rnn#pre-trained
こちらからダウンロードし、ご自身で管理しやすい、呼び出しやすいディレクトリーに管理して
ください。

＜設定ファイルの指定＞
CONFIG=<one of 'one_drum', or 'drum_kit', matching the bundle>

Drums RNNでは、学習済みデータが１つだけ提供されているのですが、設定ファイルは２種類用意されています。
one_drumとdrum_kitの２種類です。
Melody RNNでは４種類の学習済みデータの対し、該当する同じ名前の設定を使用しましたが、Drums RNNでは１つの学習済みデータdrum_kit_rnn.mag に対し、２種類の設定両方を使用する事になります。

```
CONFIG= one_drum
```

または

```
CONFIG=drum_kit
```

となります。

one_drum
すべてのドラムを単一のドラムクラスにマッピングしている設定

drum_kit
すべてのドラムを、バスドラム、スネアドラム、クローズド＆オープンハイハット、ロー&ミッド&ハイの３つのタム、クラッシュシンバル、ライドシンバルで構成される9ピースドラムキットにマッピング

とあります。
どちらもワンホットエンコーディングを使用しているのですが、ドラム音データの取り扱い方が異なっています。
２種類の設定でそれぞれ生成したドラム演奏ファイルを何度も比較していますが、正直一聴してわかるほどの違いはない様に思えます。
ただone_drumの方が音数が少々多くなる傾向があるかもしれません。
読者の皆様も是非ご自身でそれぞれ生成実験し、比べてみてください。

Macの場合

```
drums_rnn_generate \
--config=one_drumかdrum_kit \
--bundle_file=ご自身のディレクトリーを指定/drum_kit_rnn.mag \
--output_dir=ご自身の希望で任意に指定 \
--num_outputs=10 \
--num_steps=128 \
--qpm=120.0 \
--primer_drums=“[(36,)]”
```

１・改行には \ を使用

２・ディレクトリーの階層の区切りは / を使用

３・BUNDLE_PATHとCONFIGは直接指定

Windowsの場合

```
drums_rnn_generate  ^
--config=one_drumかdrum_kit ^
--bundle_file=¥ご自身のディレクトリーを指定¥drum_kit_rnn.mag ^
--output_dir=¥ご自身の希望で任意に指定 ^
--num_outputs=10 ^
--num_steps=128 ^
--qpm=120.0 ^
--primer_drums=“[(36,)]”
```

１・改行には ^ を使用

２・ディレクトリーの階層の区切りは ¥ を使用

３・BUNDLE_PATHとCONFIGは直接指定

primer_drums

＜primer_drumsの操作方法＞

単音のみ生成するMelody RNNと異なり、Drums RNNは複数のドラム楽器音が同時に発音する音楽を生成します。
また、常にリズムを刻むドラム演奏と言う特性上、休符（音のない部分）のより容易な取り扱いが求められます。
そこでprimer_drumsは

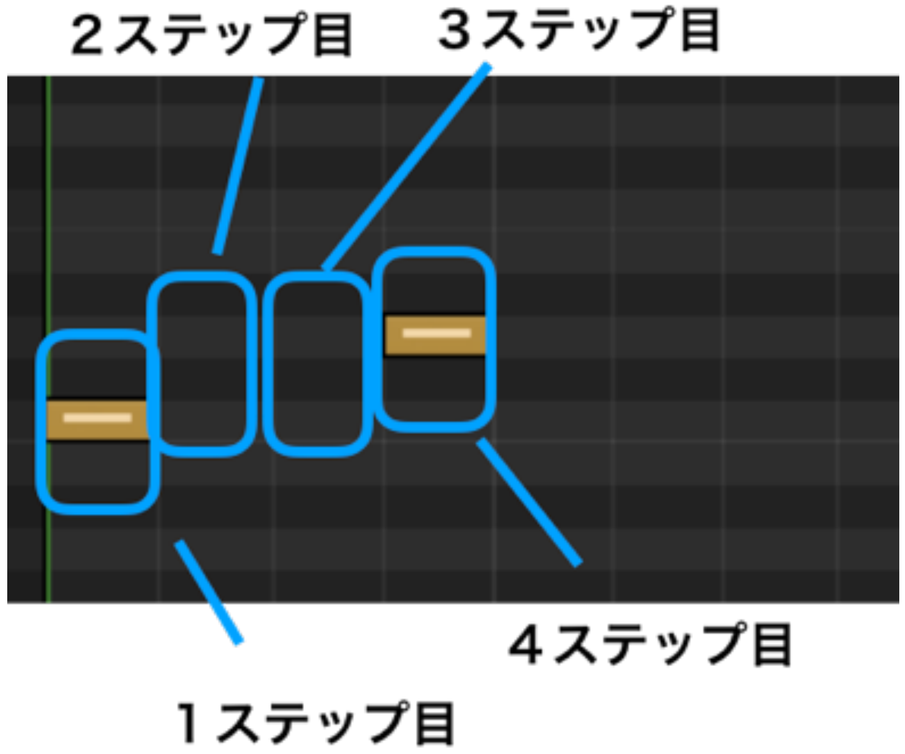
・発音しないステップは()で記述し休符を表現
・()内に，カンマで区切った複数の同時発音するノートナンバーを記述し指定
・それぞれの ()で区切られた発音セットも，で区切る

全体はMelody RNN同様にリスト形式で”[ ]”の中に音符データを記述していきます。

全体はMelody RNN同様にリスト形式で”[ ]”の中に音符データを記述していきます。
例えば4ステップ分指定し、最初と最後のステップにそれぞれバスドラムとスネアドラムを指定するとしましょう。
その場合は次の様に記述します。

```
--primer_drums="[(36,), (), (), (38,) ,]"
```

これがどの様に発音されているかと言うと



ここで注目していただきたいのは発音されている１、４ステップではなく、発音のない２、３ステップの部分です。
先のコマンドの（）がきちんと無音のステップとしてカウントされ、休符が表現されているのがお分かりいただけると思います。
Melody RNNでは休符を-1と-2で表現していました。
-1は音を止める、-2は単にノーイベントです。
これは音をどこで止めるのか、という音の長さ＝音価の指定の必要があったからです。
しかしドラム楽器は基本叩いたら短い音がなるだけですので音価の調整は必要ありません。
一方先に述べた通り、ドラムはリズム楽器の特性上、常に細かいステップで発音と無音（休符）が繰り返されます。
そのため、単純に休符のみをより容易な方法で表現できる方が便利です。
つまり（）のみで簡単に休符を表現できるのはDrums RNNにとってはリーズナブルな方法と言えるのです。

複数のドラムの同時発音方法

複数のドラム楽器を同時に発音させる方法はどの様になっているのでしょうか？
複数のドラム音の発音は (36,42,) の様に( )内に , カンマで区切って複数の音を記述する事で実行する事ができます。
(36,42,)はバスドラムとクローズドのハイハットが同時に発音されます。
もしも使用可能な９音全てを同時に発音させたい場合は

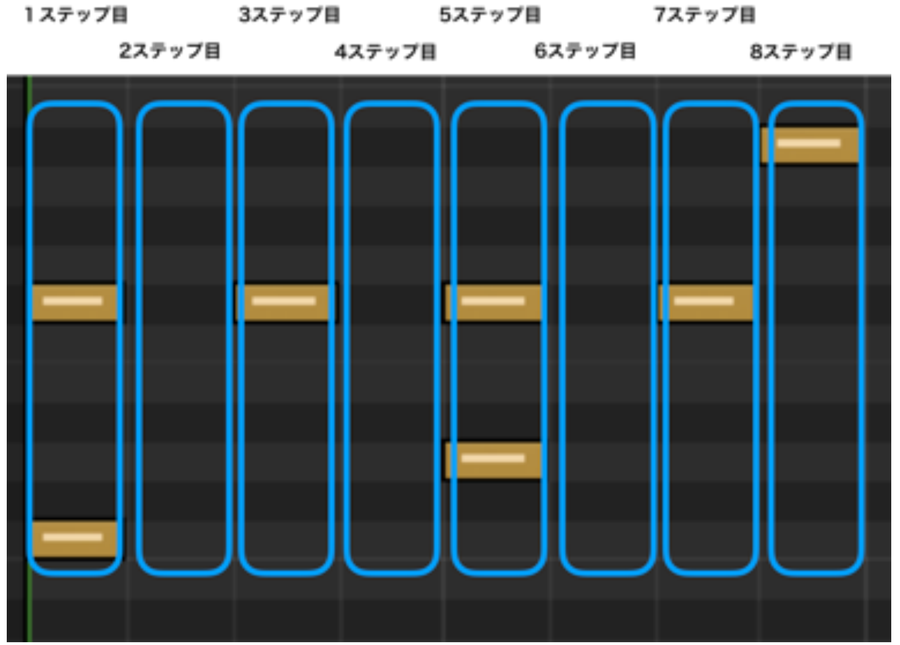
```
“[(36,38,42,45,46,48,49,50,51)]"
```

と記述する事で指定可能です。（通常のドラム演奏で９音同時発音はないとは思いますが、、、）

複数ドラム音の同時発音と休符も組み合わせ、よりドラムらしい演奏データになる様に記述してみます。

```
--primer_drums="[(36,42,), (), (42,), (), (38,42), (), (42,), (46,),]"
```

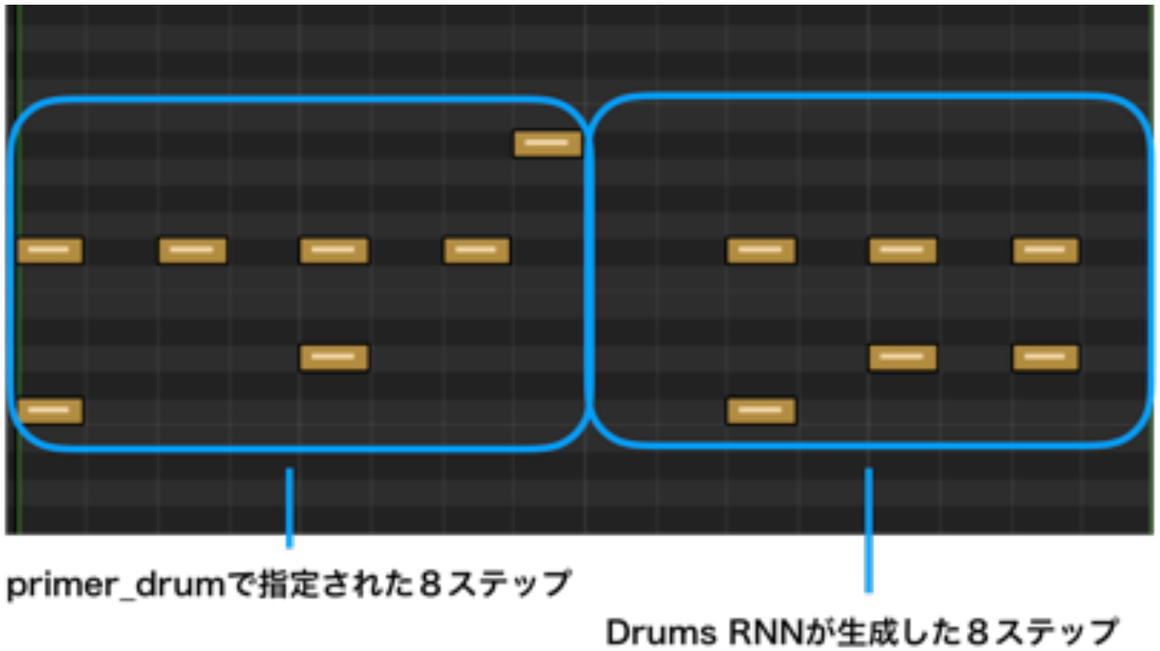
トータルで８ステップですが、これがどの様に発音されるかと言うと下図です。
１ステップ目がバスドラムとクローズドハイハット、５ステップ目がスネアドラムとクローズドハイハットの２つのドラム音が重なって発音されています。



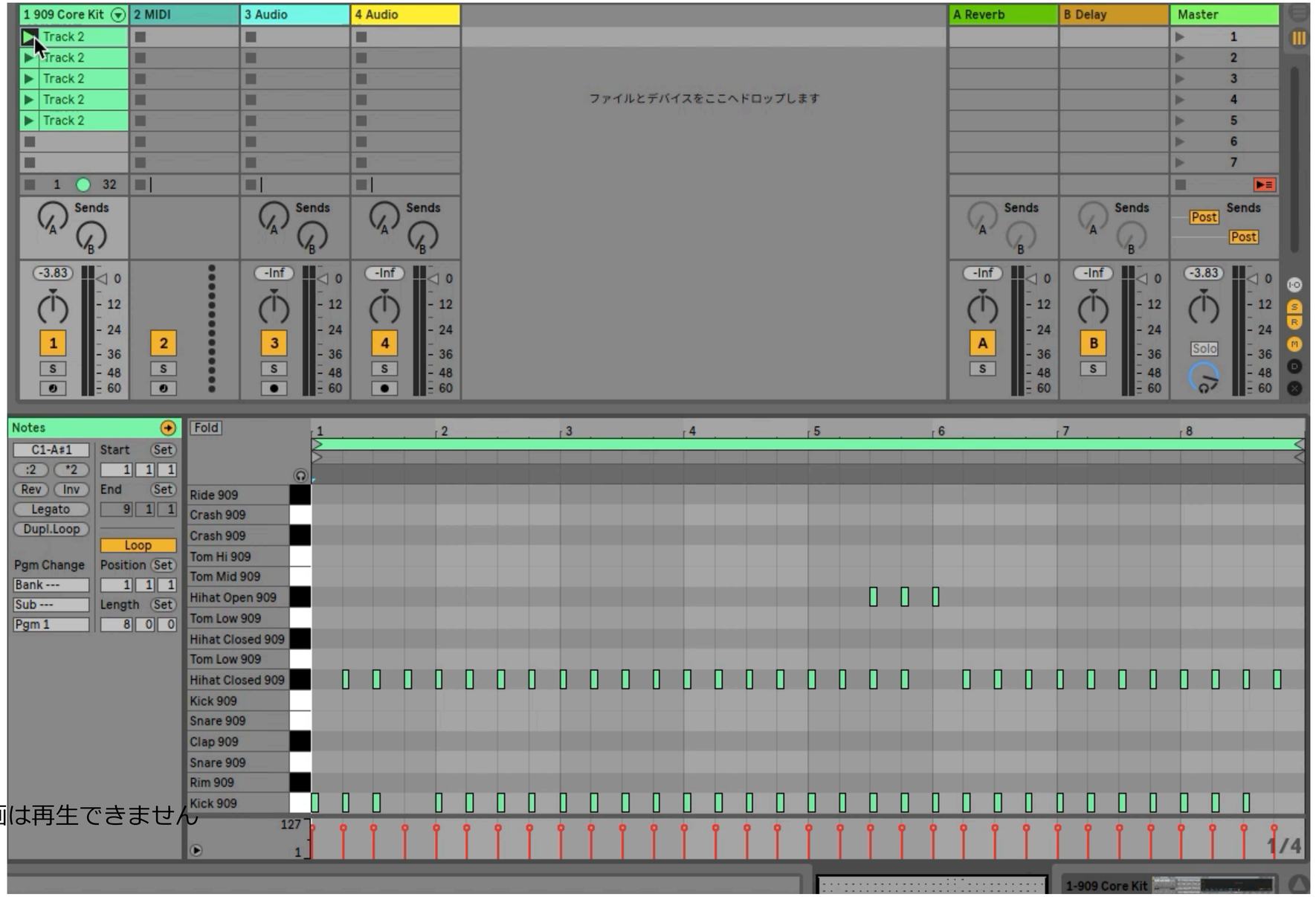
では、この音符データを与えて、１小節（１６ステップ）のドラム演奏を生成してみましょう。

```
--primer_drums="[(36,42,), (), (42,), (), (38,42), (), (42,), (46,),]"
```

Magentaでは生成時に指定し与えた音符データの後に続く音符を生成しますので今回は前半８ステップが指定した音符データ、後半がDrums　RNNが生成した音符データ、トータル１６ステップのドラム演奏生成となります。



Drums_RNN primer-drumsでC1(36)だけを与えて生成(動画)



※資料では動画は再生できません

primer_midi

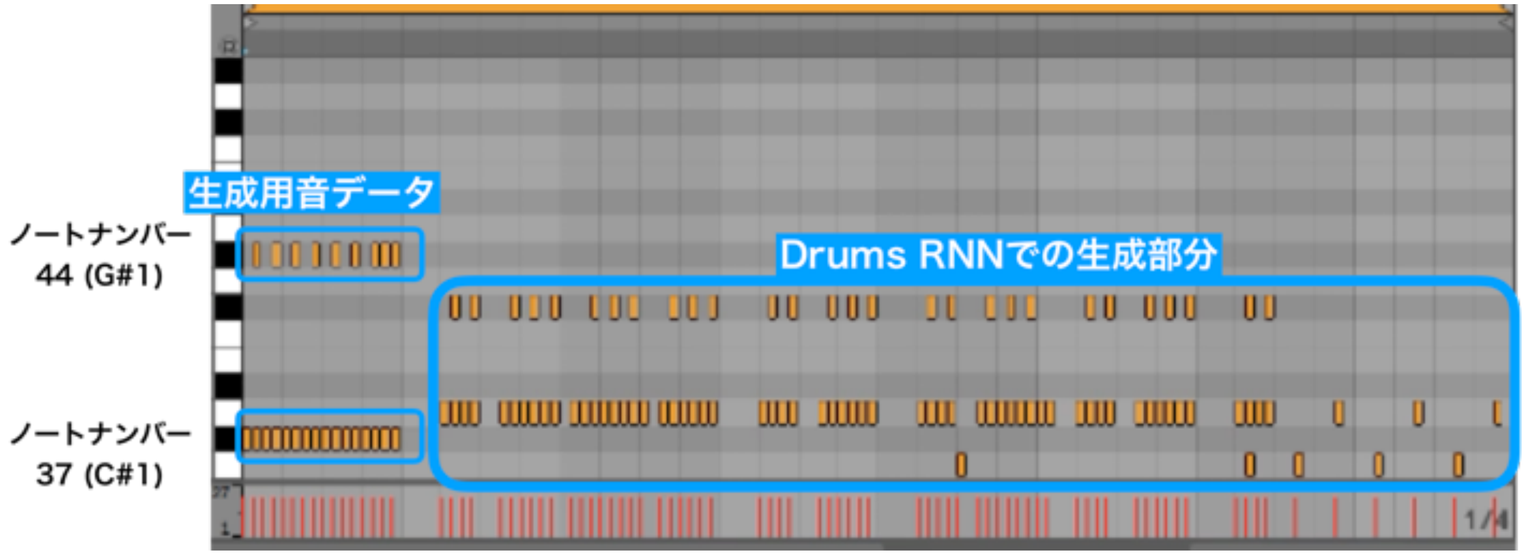
＜primer_midiを使用したmidiファイルでの生成方法＞

Drums RNNでも生成にMIDIファイルを使用することが可能です。
方法はMelody RNNのprimer_midiの方法と全く同様です。

drums001.midを使用する場合
`Users/Downloads/drums001.mid`（Mac or Ubuntu）
や
`Users¥user-name¥downloads¥drums001.mid`（Windows）

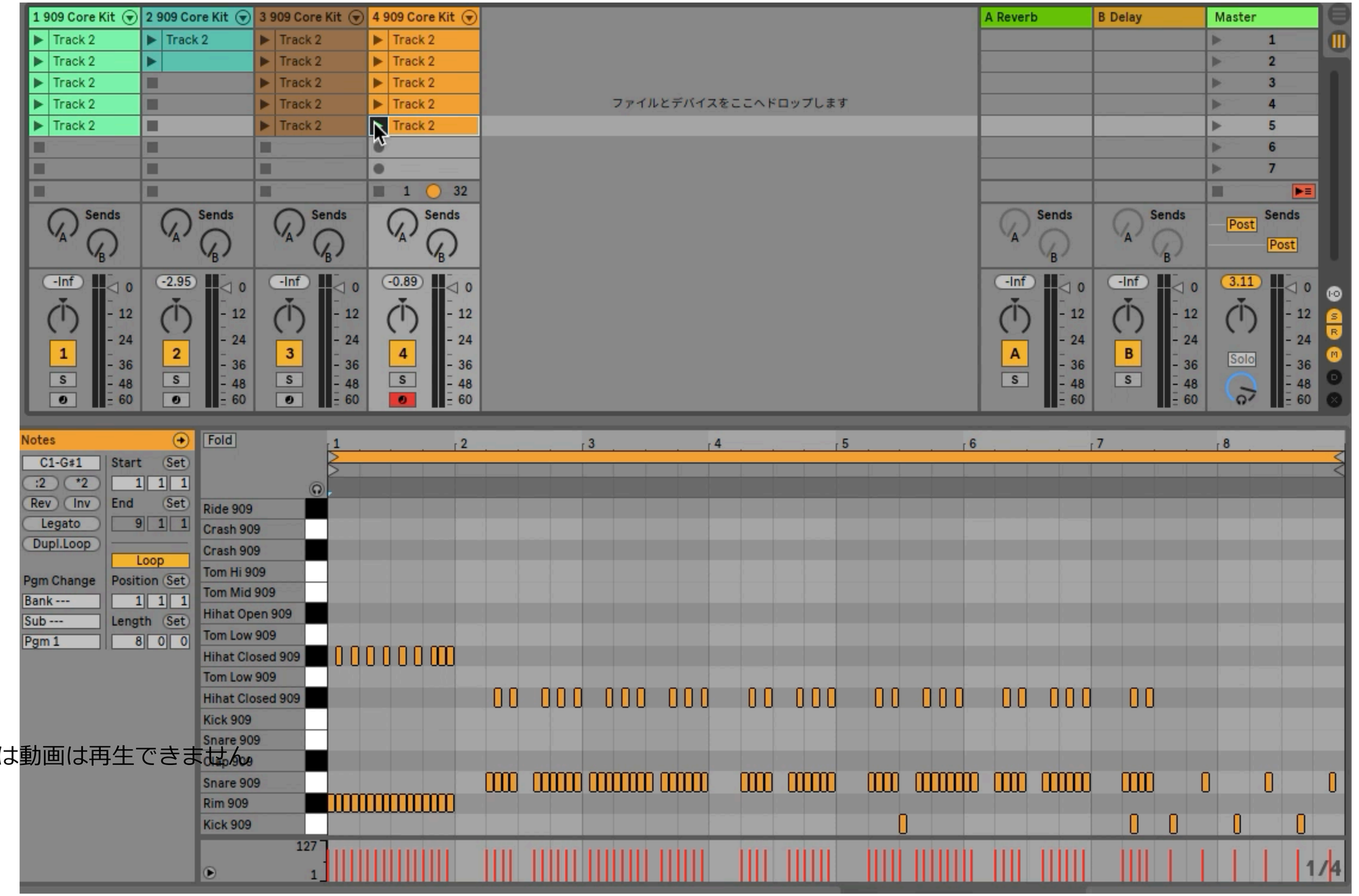
留意すべき点は、先ほどのprimer_drumsで解説した通り、９つのドラム音以外のノートナンバーは無視されます。
MIDIファイルの中に９つの音以外の音があってもエラーにはならず読み込みはされますが、生成には全く反映されません。

９つのドラム音以外の音、今回は37　(C#1)と44　(G#1)だけを使用した１小節のMIDIファイルを使用して生成してみました。



図でご確認いただける通り、ノートナンバー37と44だけで作られた生成用のMIDIの音は、エラーにはならずに読み込まれていますが、Drums　RNNでの生成の際には一切反映されず、使用可能な９音のみで生成されているのがわかります。
これはprimer_drumsで生成する場合も同様です。
Drums　RNNでドラム演奏を生成する場合は、使用可能な９音のみを使用して生成する事が基本的ルールと覚えておいてください。

Drums_RNN 意図的に範囲外のノートのMIDIを使用しての生成



※資料では動画は再生できません

Macの場合

```
drums_rnn_generate \
--config=one_drum \
--bundle_file=ご自身のディレクトリーを指定/drum_kit_rnn.mag \
--output_dir=ご自身の希望で任意に指定 \
--num_outputs=10 \
--num_steps=128 \
--qpm=120.0 \
--primer_midi=Users¥user-name/downloads/drums001.mid  #MIDIファイルへのパス
```

１・改行には \ を使用

２・ディレクトリーの階層の区切りは / を使用

３・BUNDLE_PATHとCONFIGは直接指定

Windowsの場合

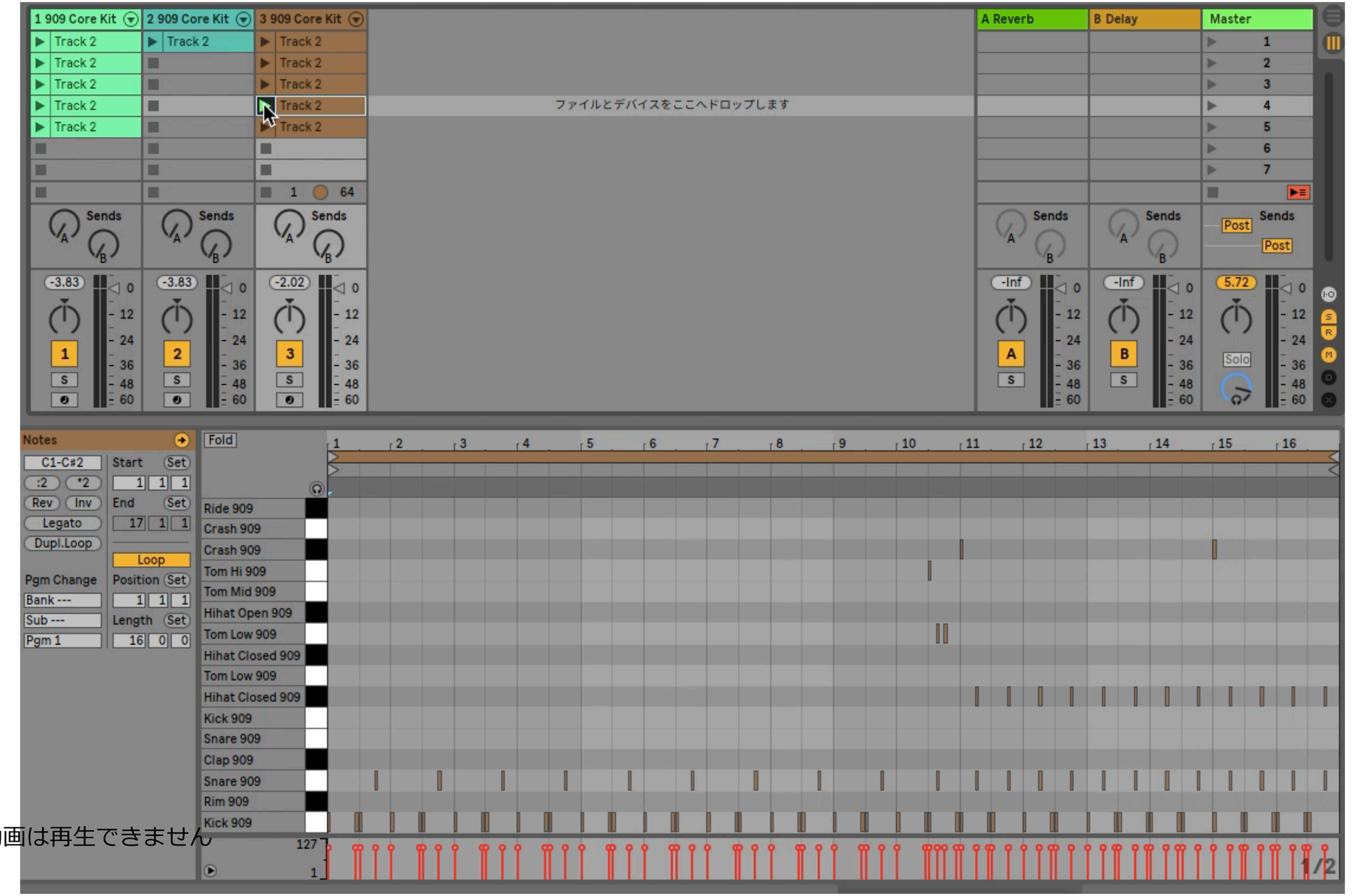
```
drums_rnn_generate ^
--config=one_drum ^
--bundle_file=¥ご自身のディレクトリーを指定¥drum_kit_rnn.mag ^
--output_dir=¥ご自身の希望で任意に指定 ^
--num_outputs=10 ^
--num_steps=128 ^
--qpm=120.0 ^
--primer_midi=Users¥user-name¥downloads¥drums001.mid  #MIDIファイルへのパス
```

１・改行には ^ を使用

２・ディレクトリーの階層の区切りは ¥ を使用

３・BUNDLE_PATHとCONFIGは直接指定

Drums_RNN primer-midiでの生成



※資料では動画は再生できません

Magentaでできる音楽生成（本講座内で解説予定）

・単音のシンプルなメロディー生成
・ドラムトラックの生成
・３パート演奏（メロディー、ベース、ドラム）の楽曲生成
・コード進行に沿ったアドリブメロディー生成
・単音メロディーにハーモニーを生成
・表現力豊かなピアノ楽曲の生成
・Ableton Live（または他のDAW）での音楽生成プラグイン活用

MusicVae

３トラックの演奏作曲を行う MusicVae

MusicVaeは３トラック
・メロディー
・ベース
・ドラム
の楽器パート演奏から成る音楽を生成できるモデルです。（他の生成もできますが代表例）

与えられた音楽データからランダムにサンプリングして、２つの音楽の間にある潜在的な音空間（latent constraint model）を補完し操作する事で新たな音楽を生成します。

メロディーとベースの生成にはMelody RNNが、ドラムの生成にはDrums RNNが用いられています。

MusicVaeの仕組み

この２つの黒いシーケンス（音楽データ）がオリジナル

２つのシーケンスを徐々につなぐ音楽生成を行う

この生成（と学習）を行う際に、２つの音楽の間にある潜在的な音空間（latent constraint model）を埋める手法としてLatent Spacesという手法が用いられています。

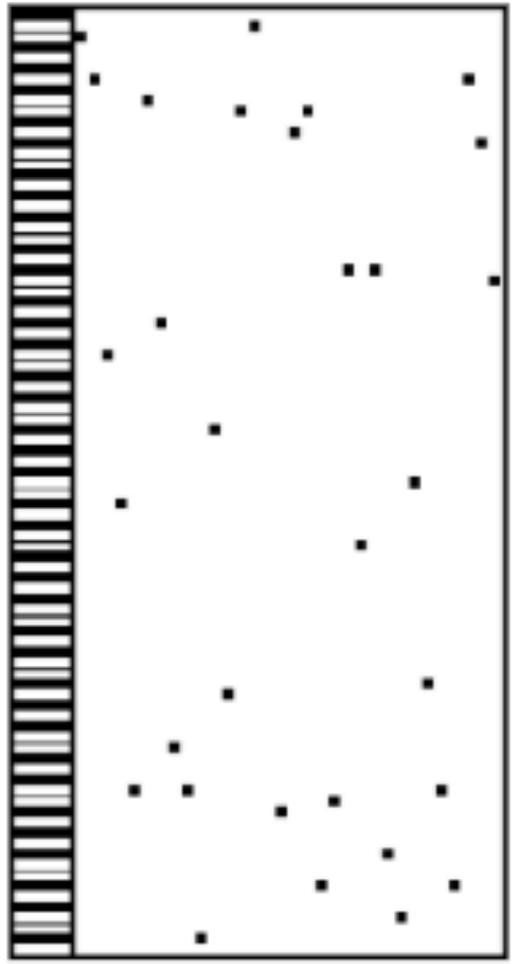
Latent Spacesについて

MusicVaeでは音楽の学習と生成の際、Latent Spacesという手法を採用しています。

音楽データを取り扱う際、８８音（ピアノの鍵盤数と同じ数）に無音、音を止めるという事も含めると、９０のデータを取り扱う必要があります。

１６分音符（１小節に１６音）で２小節、これらを全て配列の組み合わせとして取り扱った場合、９０の３２乗となり、１６小説だと９０の２５６乗となります。

仮にこれをそのままランダムに学習＆生成してみると右図の様になります。



Latent Spacesを使用すると

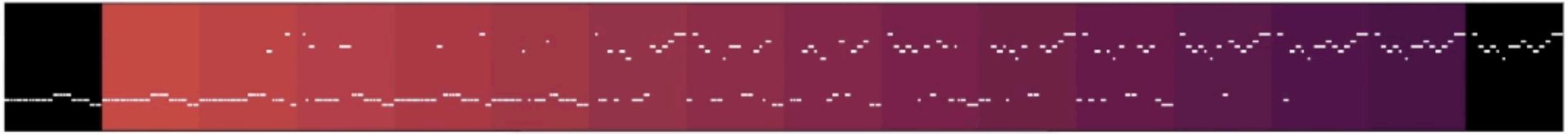
Latent　Spacesモデルでは、その音楽にある特徴を学習し、特徴外の音の可能性を排除します。

これにより、より音楽的な生成を行う事が可能になっています。
また異なる楽器パートにおいても、ランダムな生成では難しい音楽の統一感を生む事にも貢献します。

右図でご確認いただける様に、先ほどのランダム生成に比べ、整然とした音楽データが生成が行われています。

MusicVaeを使用していない

Data Space Interpolation (*not* MusicVAE)



0:00 / 1:06

MusicVaeを使用

Latent Space Interpolation (MusicVAE)

0:00 / 1:06

MusicVae
の学習データ

MusicVae特徴の一つは非常にたくさんの学習データが提供されている事です。

cat-mel_2bar_big	**2-bar melodies**
hierdec-mel_16bar	**16-bar melodies**
hierdec-trio_16bar	**16-bar "trios" (drums, melody, and bass)**
cat-drums_2bar_small.lokl	**2-bar drums w/ 9 classes trained for more *realistic* sampling**
cat-drums_2bar_small.hikl	**2-bar drums w/ 9 classes trained for *better reconstruction and interpolation***
nade-drums_2bar_full	**2-bar drums w/ 61 classes**
groovae_4bar	**4-bar groove autoencoder.**
groovae_2bar_humanize	**2-bar model that converts a quantized, constant-velocity drum pattern into a "humanized" groove.**
groovae_2bar_tap_fixed_velocity	**2-bar model that converts a constant-velocity single-drum "tap" pattern into a groove.**
groovae_2bar_add_closed_hh	**2-bar model that adds (or replaces) closed hi-hat for an existing groove.**
groovae_2bar_hits_control	**2-bar groove autoencoder, with the input hits provided to the decoder as a conditioning signal.**

MelodyRNNなどと違い.magファイルではなく.tarファイルです。容量が大きいものがあるので注意

学習済みデータの特徴を知り使い分けてください

cat-mel_2bar_big	単一トラックの２小節のメロディーを生成します
hierdec-mel_16bar	単一トラックの１６小節のメロディーを生成します
hierdec-trio_16bar	３パート（ドラム、メロディー、ベース）の１６小節の演奏曲を生成します
cat-drums_2bar_small.lokl	２小節、９つのドラム楽器を使用したドラムトラック。よりリアルな演奏を生成します
cat-drums_2bar_small.hikl	２小節、９つのドラム楽器を使用したドラムトラック。２つの音楽データをミックスするのに適しています
nade-drums_2bar_full	６１のドラム音を使用した２小節のドラムトラック生成を行います

学習済みデータの特徴を知り使い分けてください

groovae_4bar	４小節のグルーブのある（タイミングを細かくした）ドラム生成を行います
groovae_2bar_humanize	２小節のジャストタイミング、強弱なしのドラムトラックを人間的なノリのあるものに変換します
groovae_2bar_tap_fixed_velocity	２小節の強弱なしのtap（単音ドラムトラック）を人間的なノリのあるものに変換します
groovae_2bar_add_closed_hh	２小節のドラムトラックにクローズドハイハットの演奏を追加、もしくは既存のものを変換します
groovae_2bar_hits_control	入力されたドラムデータを基にした２小節のグルーブを生成します

GrooVae補足

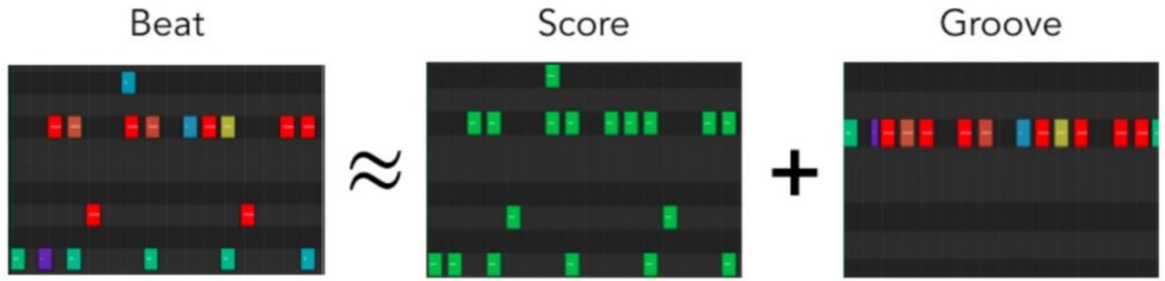
GrooVAE

GrooVAEはMusicVAE(マルチトラックの音楽生成モデル)の中のドラムトラック生成モデル。

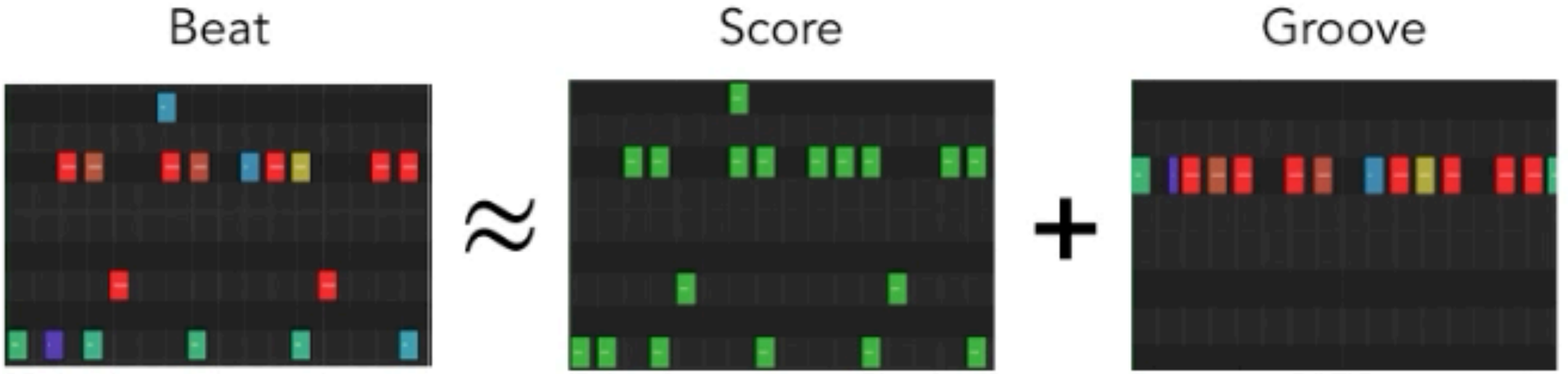
学習済みデータとしてGroove MIDI Datasetが使用されています。

GrooVAEではドラムのパターンを”楽譜”と”Groove”を2つに分けて学習します。

そして楽譜上の演奏データ(どのドラム楽器を楽譜的にどの拍で演奏するか)にGroove(強弱やタイミングのズレ)を加え、Beatとしてノリのあるドラムトラックを生成します。



/e can visualize drums in piano roll format: each row shows one piece of a drum kit (kick, snare, etc.), velocit
own in color (with louder notes in red and quieter notes in blue), and microtiming can be seen as deviations
the vertical grid lines that represent 16th notes.



We can visualize drums in piano roll format: each row shows one piece of a drum kit (kick, snare, etc.), velocity is shown in color (with louder notes in red and quieter notes in blue), and microtiming can be seen as deviations from the vertical grid lines that represent 16th notes.

GrooVAE

データ内容

・収録時間13.6時間

・1150曲のドラムパターンMIDIファイル、約22000小節以上

・10人のドラマーが即興で演奏、全パターンの80%がプロフェッショナルのドラマーの演奏

・ほとんどの曲は4/4で演奏され、Fill inや短いビートの演奏も含む

・MIDIだけでなく音声合成したオーディオデータも学習データとしてリリース

Split	Beats	Fills	Measures (approx.)	Hits
Train	378	519	17752	357618
Validation	48	76	2269	44044
Test	77	52	2193	43832
Total	**503**	**647**	**22214**	**445494**

GrooVAE

使用ドラム音マップ

Pitch	Roland Mapping	GM Mapping	Paper Mapping	Frequency
36	Kick	Bass Drum 1	Bass (36)	88067
38	Snare (Head)	Acoustic Snare	Snare (38)	102787
40	Snare (Rim)	Electric Snare	Snare (38)	22262
37	Snare X-Stick	Side Stick	Snare (38)	9696
48	Tom 1	Hi-Mid Tom	Mid Tom (48)	13145
50	Tom 1 (Rim)	High Tom	High Tom (50)	1561
45	Tom 2	Low Tom	Low Tom (45)	3935
47	Tom 2 (Rim)	Low-Mid Tom	Mid Tom (48)	1322
43	Tom 3 (Head)	High Floor Tom	Low Tom (45)	11260
58	Tom 3 (Rim)	Vibraslap	Low Tom (45)	1003
46	HH Open (Bow)	Open Hi-Hat	Open Hi-Hat (46)	3905
26	HH Open (Edge)	N/A	Open Hi-Hat (46)	10243
42	HH Closed (Bow)	Closed Hi-Hat	Closed Hi-Hat (42)	31691
22	HH Closed (Edge)	N/A	Closed Hi-Hat (42)	34764
44	HH Pedal	Pedal Hi-Hat	Closed Hi-Hat (42)	52343
49	Crash 1 (Bow)	Crash Cymbal 1	Crash Cymbal (49)	720
55	Crash 1 (Edge)	Splash Cymbal	Crash Cymbal (49)	5567
57	Crash 2 (Bow)	Crash Cymbal 2	Crash Cymbal (49)	1832
52	Crash 2 (Edge)	Chinese Cymbal	Crash Cymbal (49)	1046
51	Ride (Bow)	Ride Cymbal 1	Ride Cymbal (51)	43847
59	Ride (Edge)	Ride Cymbal 2	Ride Cymbal (51)	2220
53	Ride (Bell)	Ride Bell	Ride Cymbal (51)	5567

Music VAEで音楽生成
sampleモード

Mac sampleモード例

```
music_vae_generate \
--config=学習データと合わせる事 \
--checkpoint_file=/任意のパス/任意の学習データ \
--mode=sample \
--num_outputs=5 \ #任意の曲数
--output_dir=/任意の出力ディレクトリー
```

config（設定）と学習データは合わせないとエラーが出ます

MelodyRNNなどの学習済みデータ（.mag）ではなくチェックポイントファイル（.tar）です

tarファイルは解凍せずに使用

Windows sampleモード 生成例

```
music_vae_generate ^
--config=学習データと合わせる事 ^
--checkpoint_file=¥任意のパス¥任意の学習データ ^
--mode=sample ^
--num_outputs=5 ^ ＃任意の曲数
--output_dir=¥任意の出力ディレクトリー
```

１・改行には ^ を使用

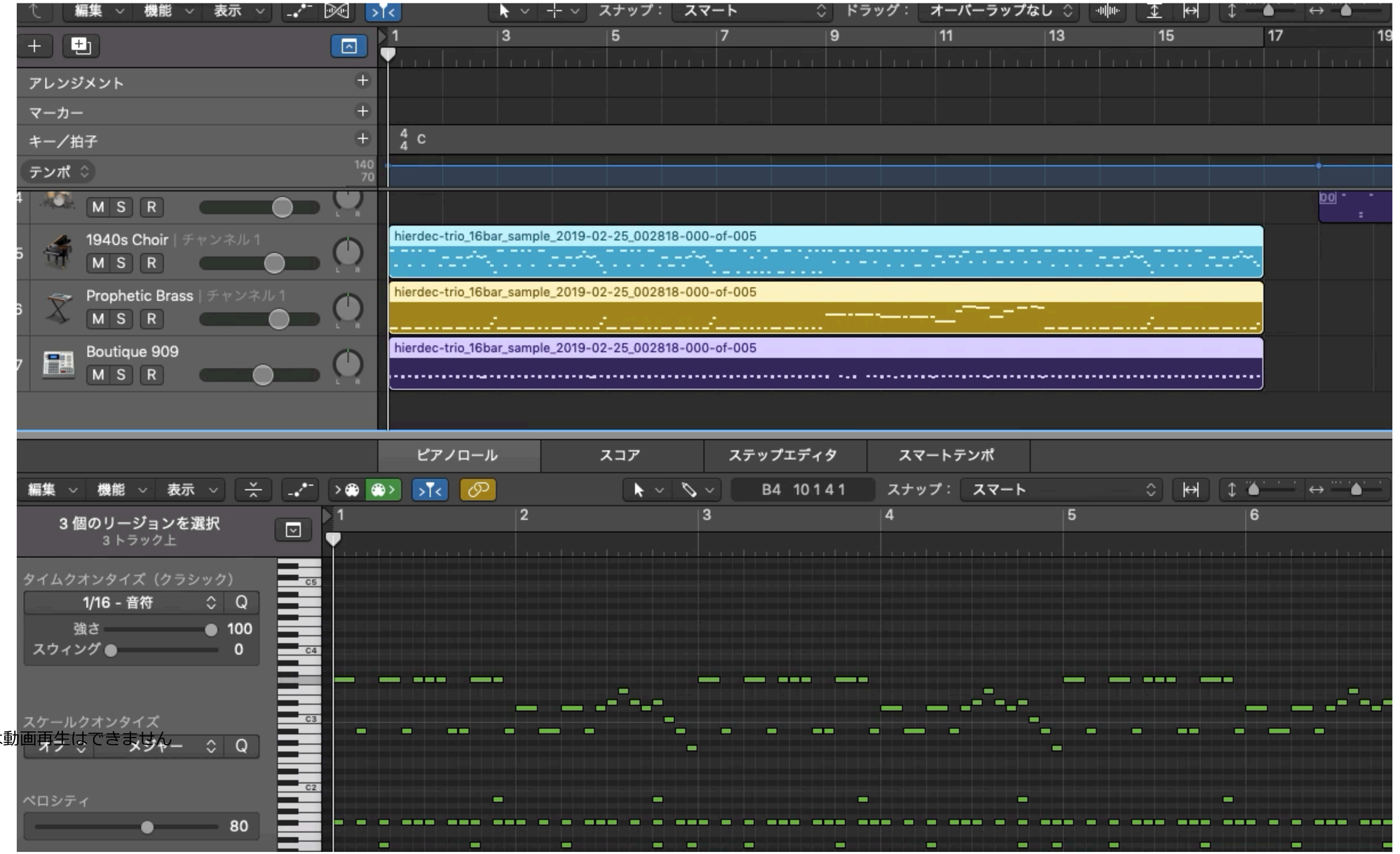
２・ディレクトリーの階層の区切りは ¥ を使用

３・checkpoint_fileとCONFIGは直接指定

Music VAEの生成方法

```
music_vae_generate \

--config=hierdec-trio_16bar \
#config今回ははhierdec-trio_16barを指定（３パートの演奏）

–checkpoint_file=/任意のパス/hierdec-trio_16bar.tar \
#ダウンロードした学習データhierdec-trio_16bar.tarのPath

--mode=sample \
#モードは２種類
学習データを元に生成する場合はsample
自身のMIDIファイルを２曲混ぜ合わせる場合はinterpolate

--num_outputs=5 \
#生成ファイル数

--output_dir=/任意の出力ディレクトリー
```

Music VAEで生成した音楽

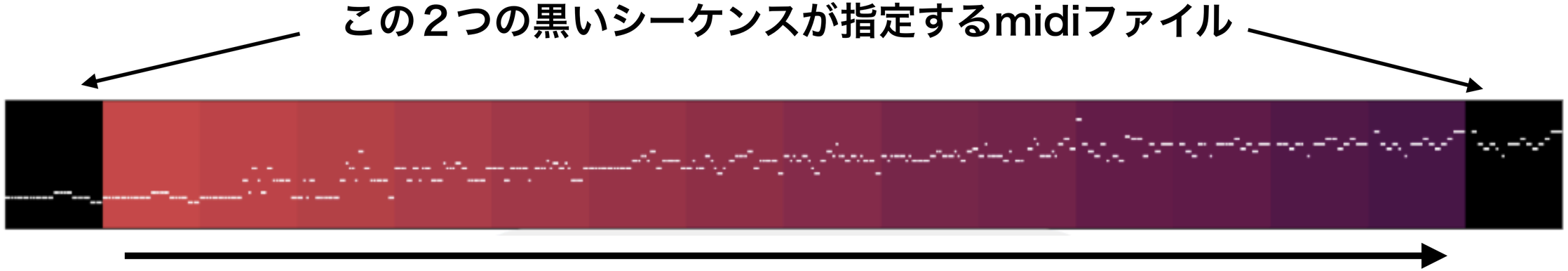


※資料では動画再生はできません

MusicVAEで音楽生成
interpolateモード

interpolateモードとは

この様に２曲のmidiファイルをつなぐ生成を行います



２つのシーケンスを徐々につなぐ音楽生成を行う

この生成（と学習）を行う際に、２つの音楽の間にある潜在的な音空間（latent constraint model）を埋める手法としてLatent Spacesという手法が用いられています。

Mac MusicVae interpolateモード例

```
music_vae_generate \
--config=hierdec-trio_16bar \
--checkpoint_file=/任意のパス/hierdec-trio_16bar.tar \
--mode=interpolate \
--num_outputs=5 \
--input_midi_1=/ご自身のmidiファイル1指定 \
--input_midi_2=/ご自身のmidiファイル2指定 \
--output_dir=/任意の出力ディレクトリー
```

この様なコマンドで生成します

（注意）
メロディー（単音）、ベース（単音）、ドラム（和音OK）の３トラック、１６小節の長さを合わせるの必要があります。
メロディー、ベースは和音でもエラーは出ませんが、単音しか出力されません。

Windows MusicVae interpolateモード例

```
music_vae_generate ^
--config=hierdec-trio_16bar ^
--checkpoint_file=/任意のパス/hierdec-trio_16bar.tar ^
--mode=interpolate ^
--num_outputs=5 ^
--input_midi_1=/ご自身のmidiファイル1指定 ^
--input_midi_2=/ご自身のmidiファイル2指定 ^
--output_dir=/任意の出力ディレクトリー
```

この様なコマンドで生成します

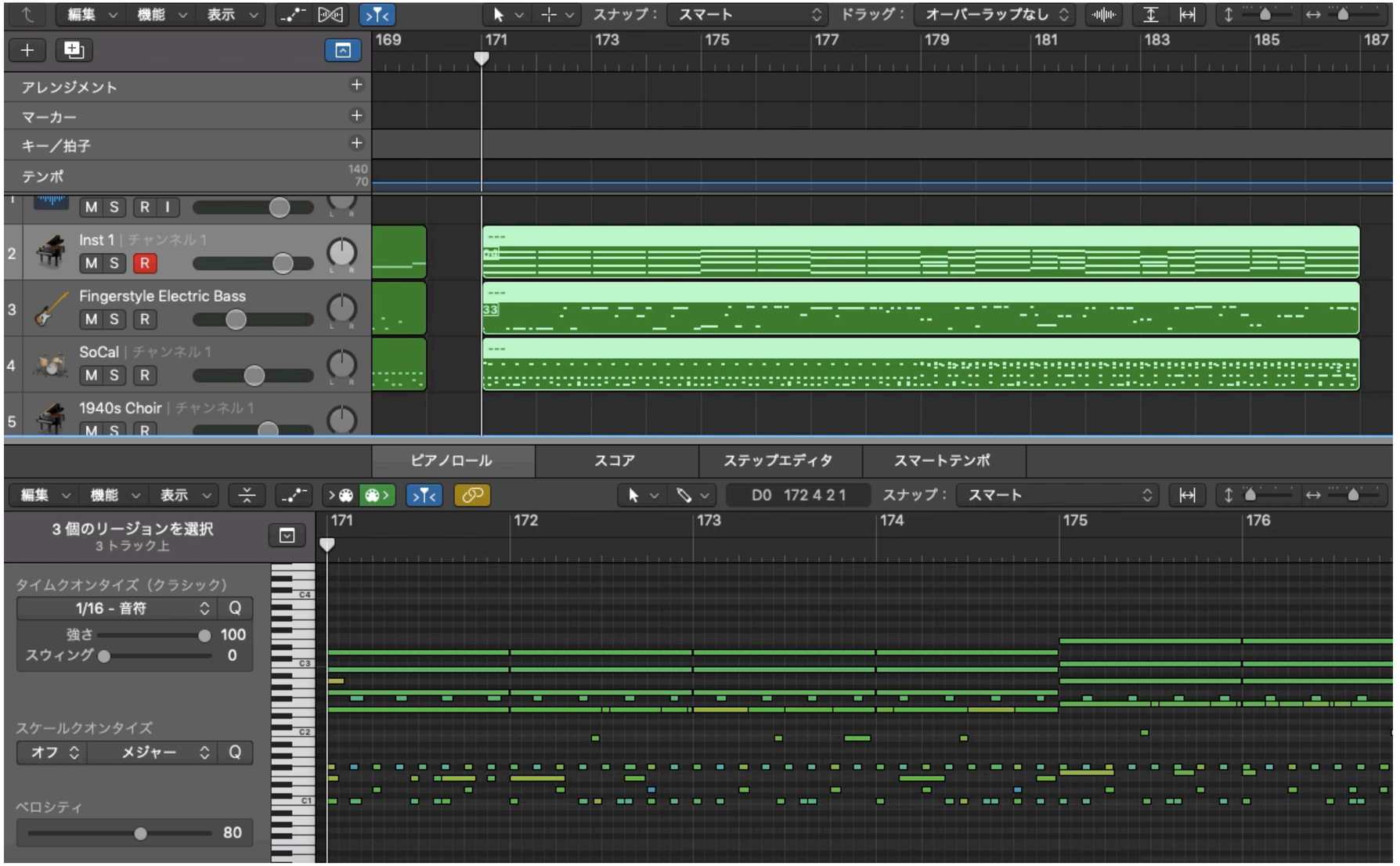
（注意）
メロディー（単音）、ベース（単音）、ドラム（和音OK）の３トラック、１６小節の長さを合わせるの必要があります。
メロディー、ベースは和音でもエラーは出ませんが、単音しか出力されません。

MusicVae interpolateモード例

```
music_vae_generate \
--config=hierdec-trio_16bar \
--checkpoint_file=/任意のパス/hierdec-trio_16bar.tar \
#ダウンロードした学習データhierdec-trio_16bar.tarのPath


--mode=interpolate \
#モードは２種類
学習データを元に生成する場合はsample
自身のMIDIファイルを２曲混ぜ合わせる場合はinterpolate


--input_midi_1=/ご自身のmidiファイル1指定 \
--input_midi_2=/ご自身のmidiファイル2指定 \
#混ぜ合わせる２曲のMIDIファイルを指定


--num_outputs=5 \
#生成ファイル数


--output_dir=/任意の出力ディレクトリー
```

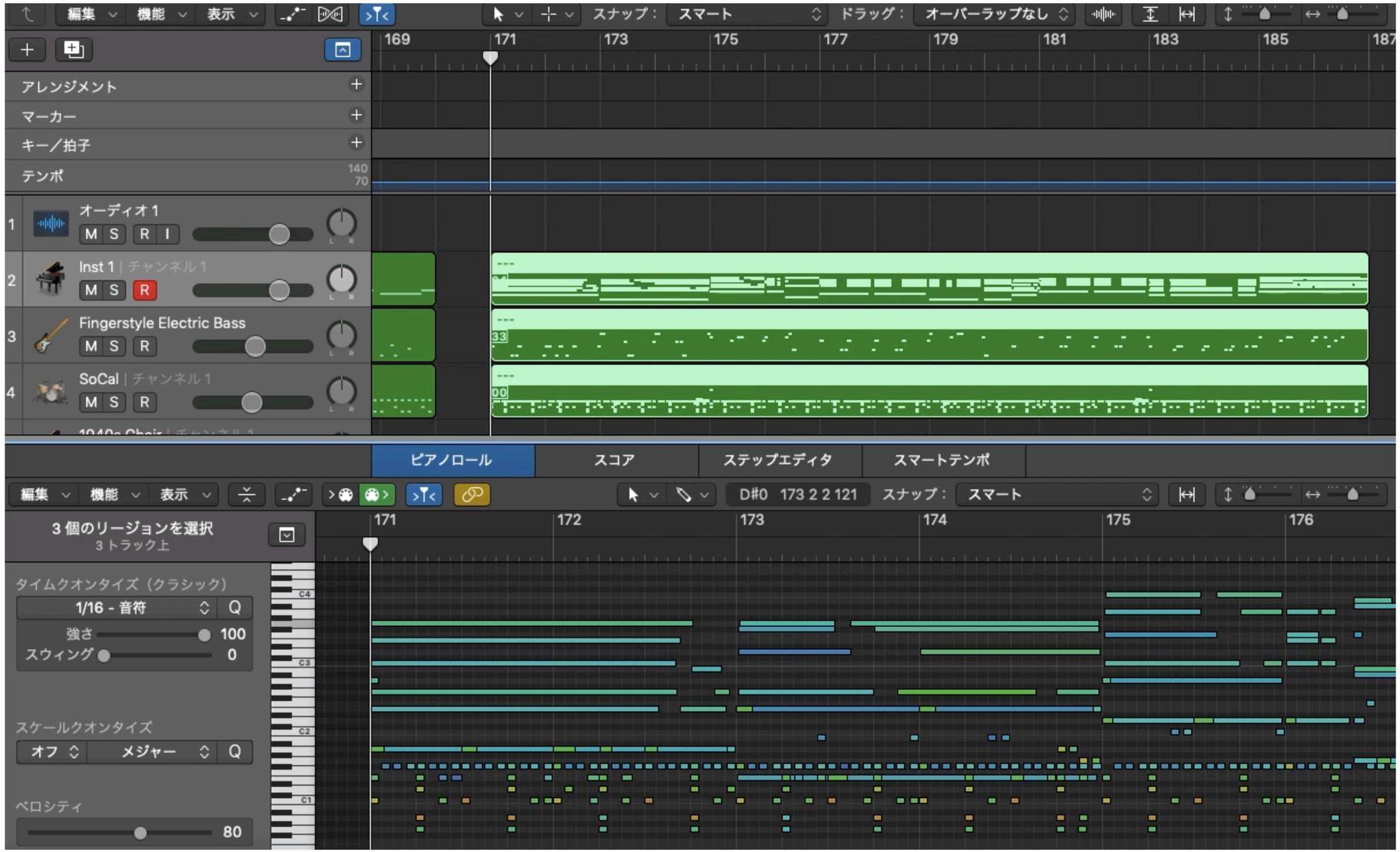
Music VAE interpolateモード
の生成曲を聴いてみましょう

元になったMIDIファイル１



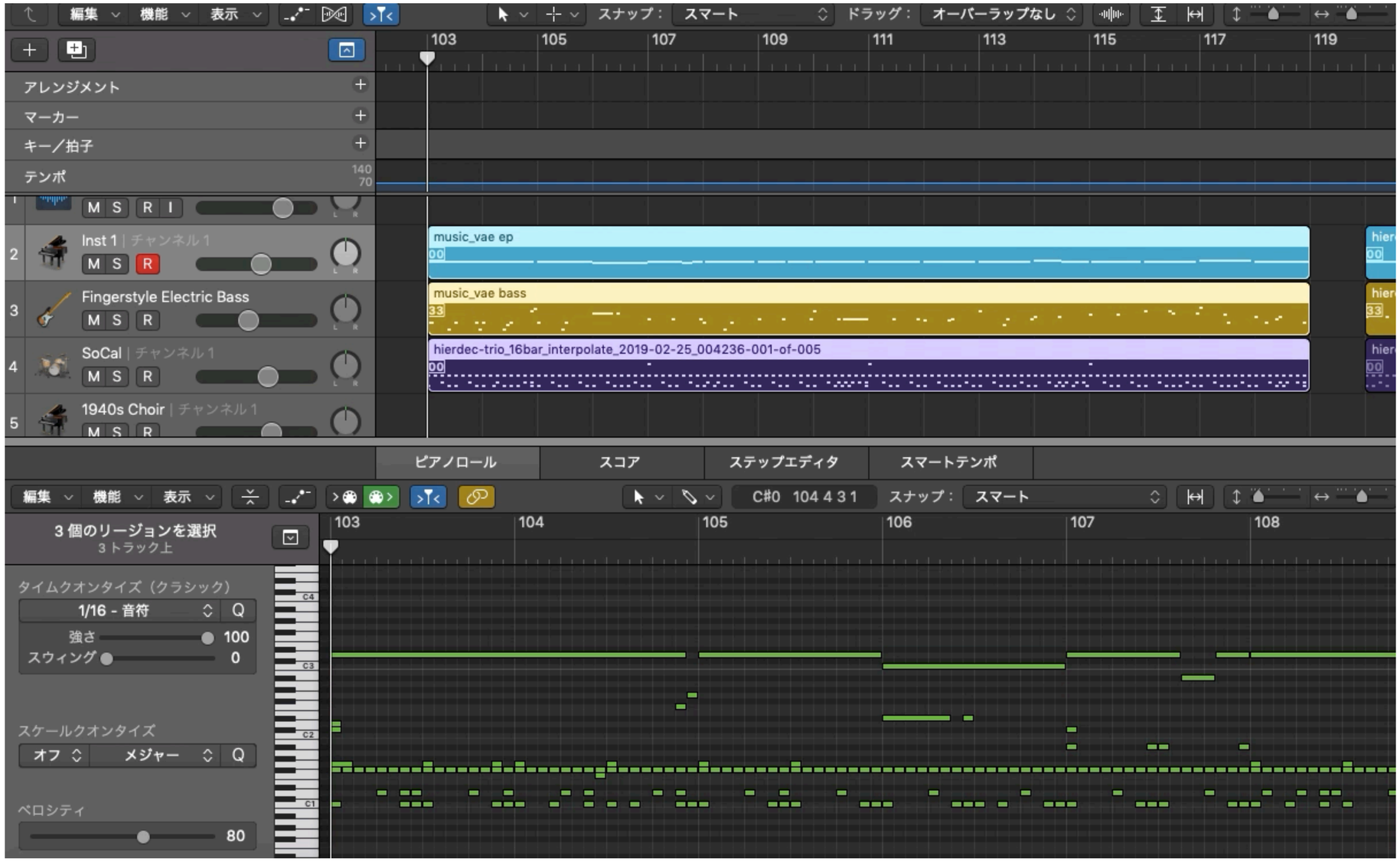
※資料では動画再生はできません

元になったMIDIファイル2



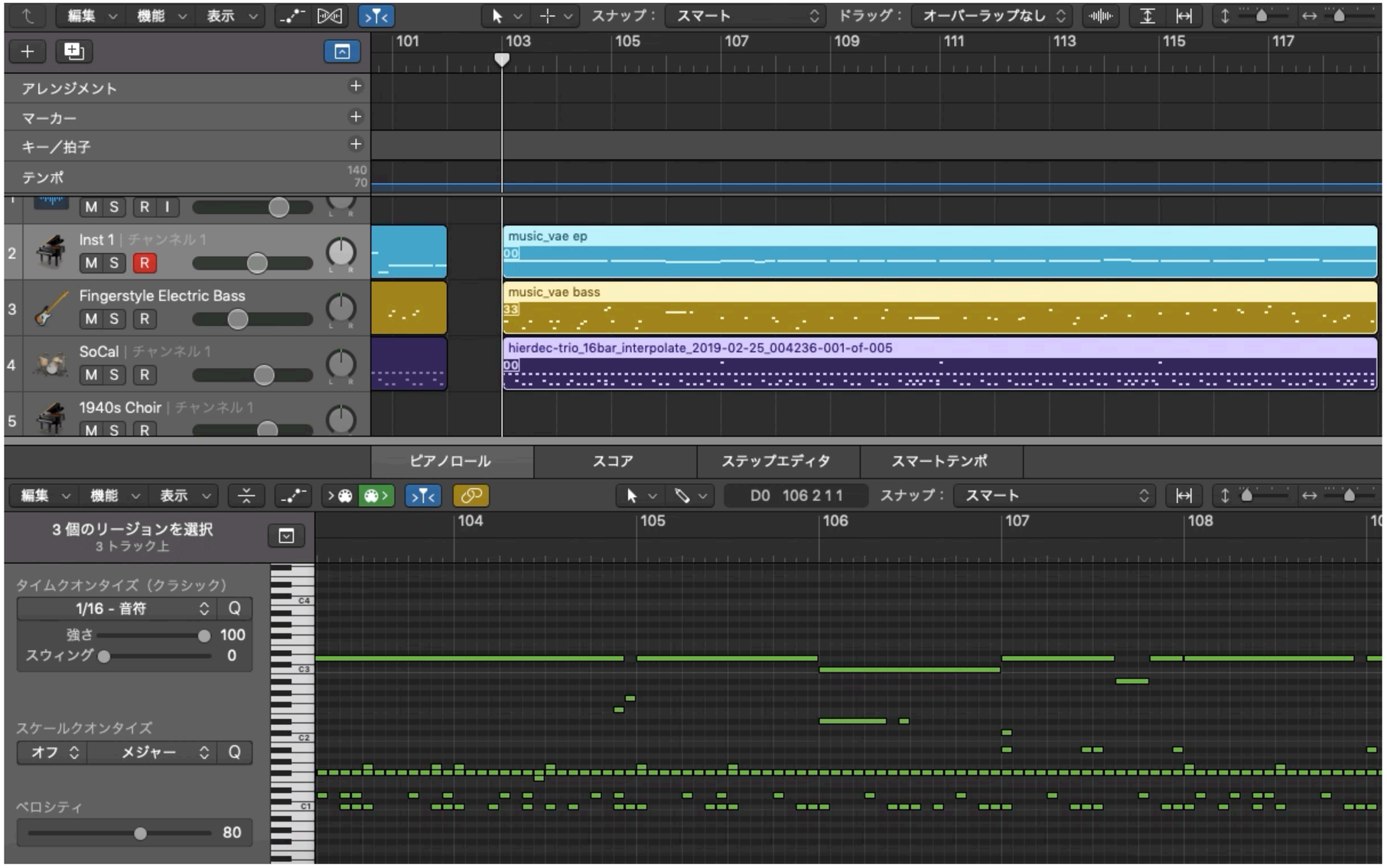
※資料では動画再生はできません

interpolateモード生成曲



※資料では動画再生はできません

interpolateモード生成曲　コード生成プラグインで和音に



※資料では動画再生はできません