

# 音樂生成基礎

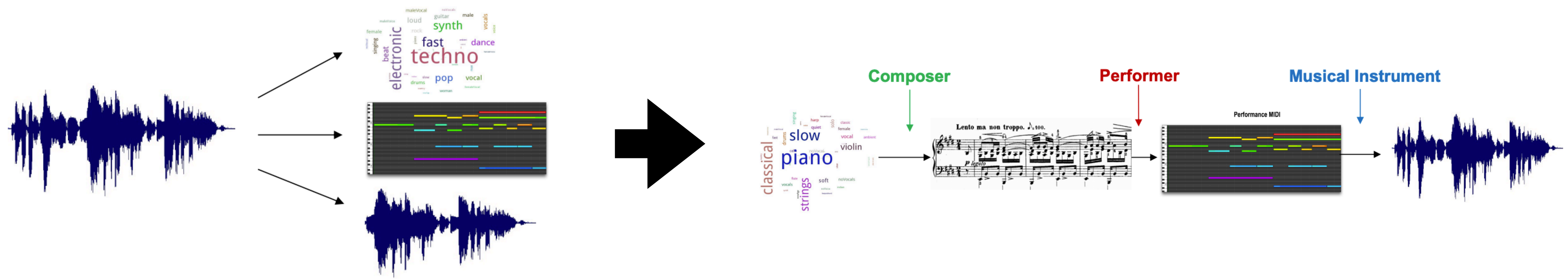
Deep-people #9

- **AutoEncoder**
  - 音->音の変換タスクに用いる
  - 情報を圧縮してまた復元というプロセスを辿る
  - 特徴抽出や音源分離・合成に用いられる
  - エンコーダが畳み込みの場合はデコーダでは転置畳み込みを用いる

- Juhanの資料
- [https://mac.kaist.ac.kr/~juhan/gct634/Slides/\[week12-1\]%20symbolic%20music%20generation.pdf](https://mac.kaist.ac.kr/~juhan/gct634/Slides/[week12-1]%20symbolic%20music%20generation.pdf)

- 音楽を生成する方法

- いままでは存在する音響信号から情報を抽出・分析したり音を変換する方法にフォーカス
- 0から新しく音楽を作り出すという問題も存在
- Label-to-Score, Score-to-MIDI, Midi-to-Audio etc...



- **Symbolic music generation; 楽譜ドメインでの生成**

今回はこっち

- 音楽を楽譜の形で生成（ほとんどはMIDI）
- 音楽言語モデル（Musical LM）に従って生成する -> 自然言語処理の方法が応用される
- RNNやTransformer等，系列を扱うモデルがよく用いられる

- **Audio generation; 音響信号ドメインでの生成**

- 音楽を波形 or スペクトログラムの形で生成
- 連続的な値をとる波形 or スペクトログラムを生成 -> 画像処理の方法が応用される
- クオリティが大事；GANやDiffusion model等が用いられる

## 言語モデルで捉えることができる

自然言語処理  
における言語モデル

The sky is so \_\_\_\_\_  
 $x_1 \ x_2 \ x_3 \ x_4 \dots$

→ blue  
→ beautiful  
→ dark

ある単語の次の単語を予測

音楽における言語モデル



$x_1 \ x_2 \ x_3 \ x_4 \dots$

ある音符の次の音符を予測

## 言語モデル

$$p(x_t | x_1, \dots, x_{t-1})$$

$x_t$ : input representation vector

音/単語の並べ方の  
ルール（文法）を  
表現するもの



## 音楽の場合， 多声を扱うことも

- メロディと伴奏が存在， 依存関係も当然存在
- MIDIだと伴奏があろうが1次元で表現
- 同時発音をどう考慮すればいいのか？



Time (Ticks)	Message	Channel	Note Number	Velocity
60	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	63	100
0	NOTE ON	2	51	100
0	NOTE ON	2	39	100
240	NOTE OFF	1	63	0
0	NOTE OFF	2	51	0
0	NOTE OFF	2	39	0

## スケール，リズム，ハーモニーという制約条件

- グローバルな調性が存在，音符の調の中に存在する音の方が頻度が高くなる傾向
- 同時に発音される音高はコードに依存
- リズムパターンの中で音符は配置される



Scale

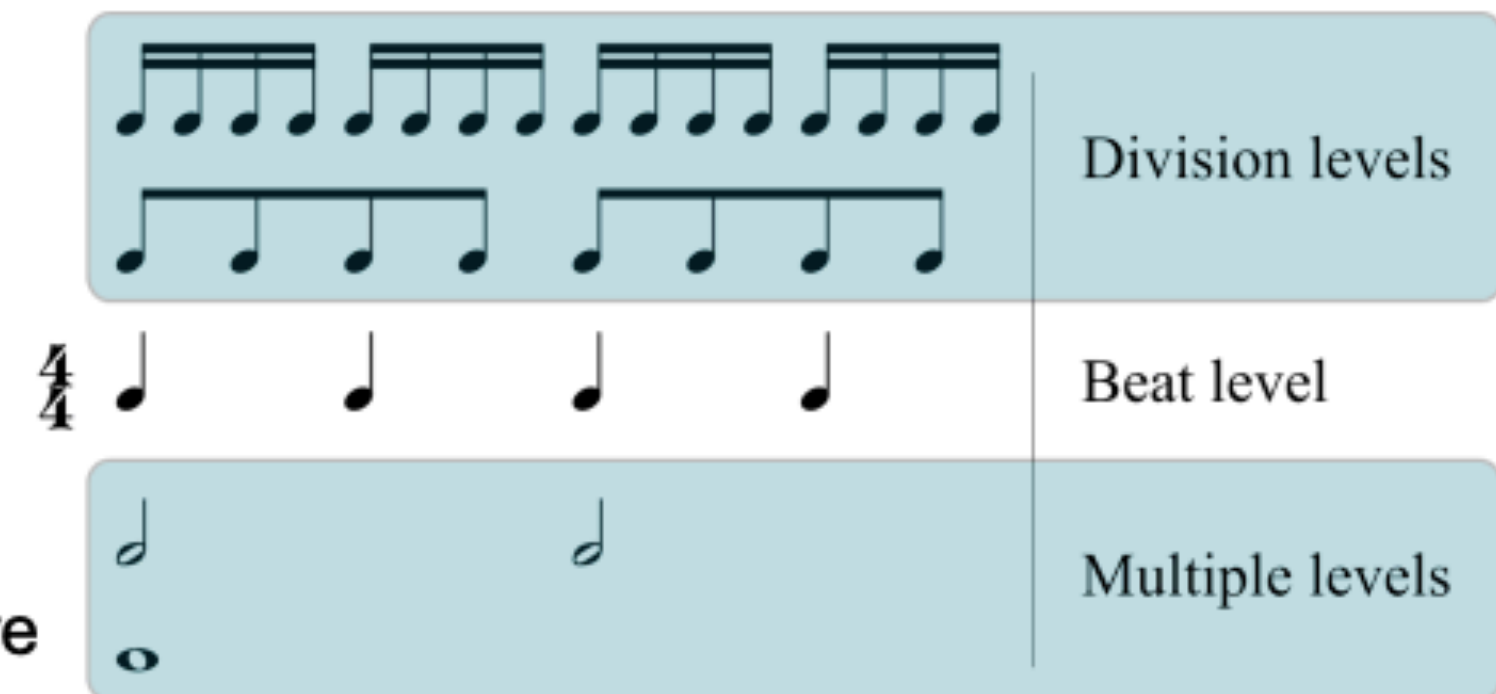


Harmony

Tick

Beat

Measure



Rhythm




## 楽曲の構成（特に長期にわたる繰り返し）の存在

- 音楽は構造を持つことが多い（ジャンルにも依るけど）
  - 繰り返しとその変形
  - ポピュラーならAメロ-Bメロ-サビ，クラシックならAABB...等
  - ソナタ形式，ロンド形式等，特有の構造
- このような長期の構造の学習はとてもチャレンジング



# モデルへの入力表現

## MIDI



Time (Ticks)	Message	Channel	Note Number	Velocity
60	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE ON	1	67	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	63	100
0	NOTE ON	2	51	100
0	NOTE ON	2	39	100
240	NOTE OFF	1	63	0
0	NOTE OFF	2	51	0
0	NOTE OFF	2	39	0

各音符のON/OFF情報を  
時系列にしたファイル

## Music XML

```
<note>  
  <pitch>  
    <step>E</step>  
    <alter>-1</alter>  
    <octave>4</octave>  
  </pitch>  
  <duration>2</duration>  
  <type>half</type>  
</note>
```



各音符の情報を  
マークアップ形式で記述

## ABC

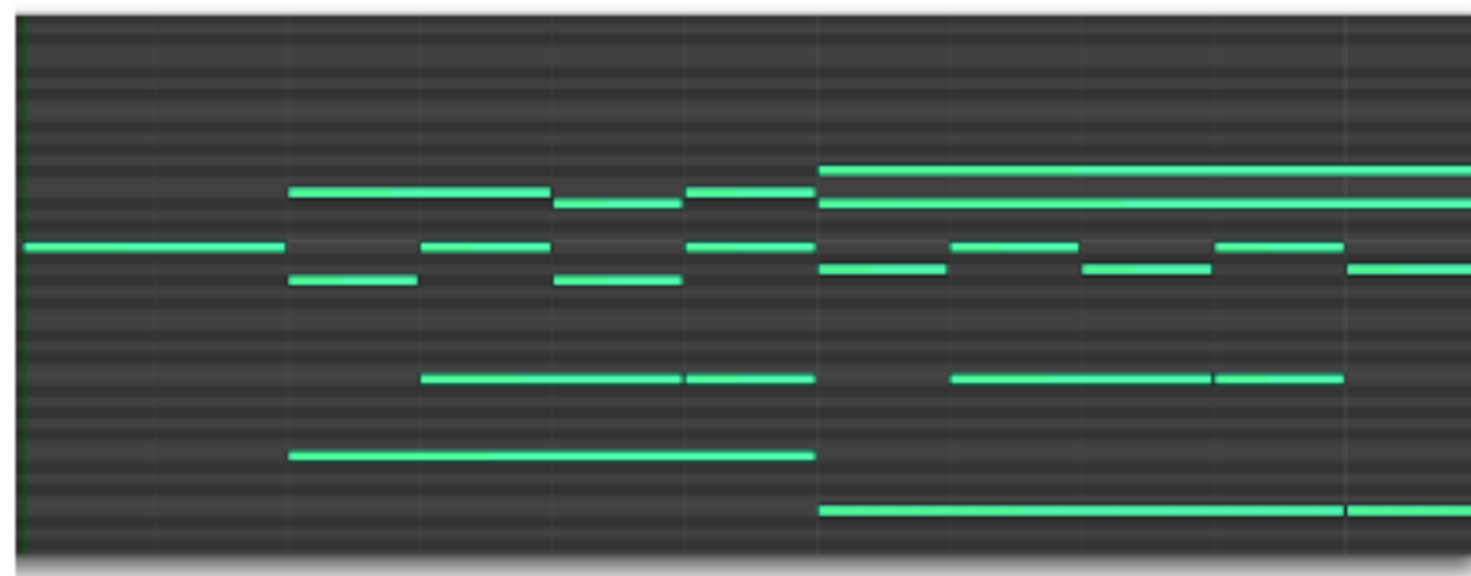
```
X:1  
T:sooranbushi  
M:2/4  
L:1/8  
K:F  
C2 DF | A2 GF | A2 GF | G2 FC | D2 FC | D2 D2 | z2 z2 ||  
zG AA | GA AA | GA AA | GF D2 | zA, CA, | CD GF | zG AF | DC FD | D z CC |  
DF A2 | A3 c | G F2 C | D2 D2 | D2 z2 ||
```

[https://ja.wikipedia.org/wiki/  
ABC%E8%A8%98%E8%AD%9C%E6%B3%95](https://ja.wikipedia.org/wiki/ABC%E8%A8%98%E8%AD%9C%E6%B3%95)

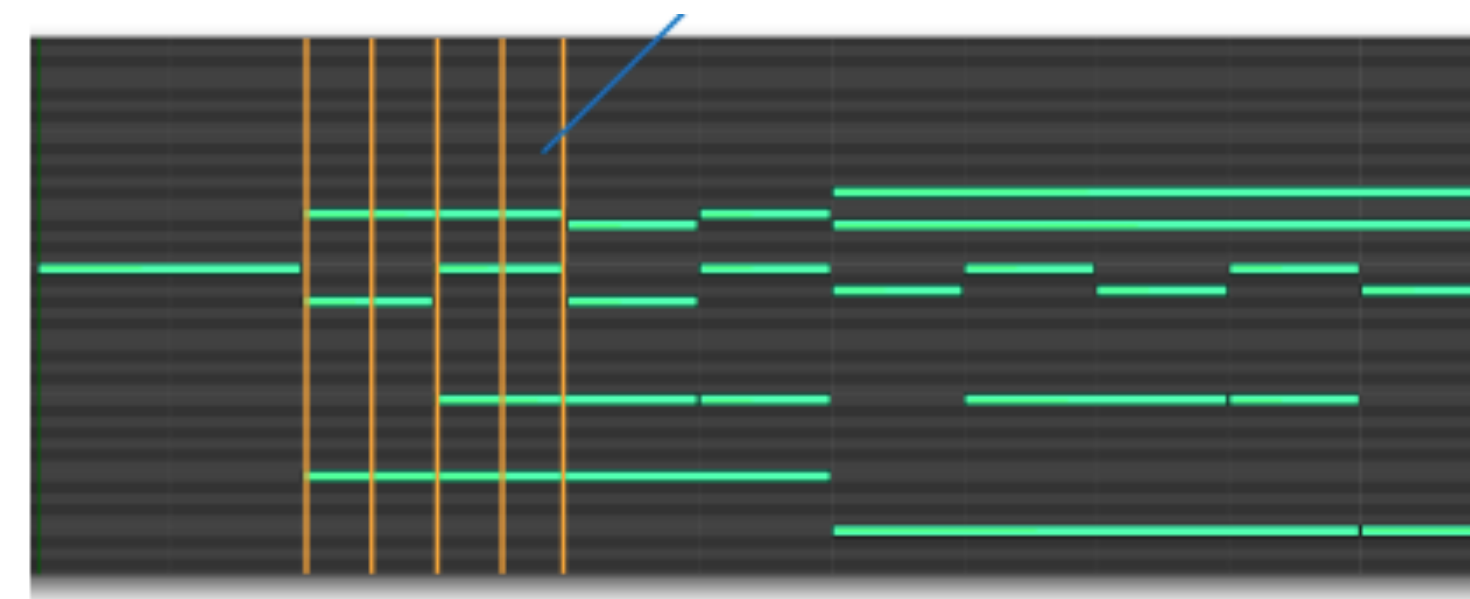
文字を並べる記法

他にもあるけど滅多に見ないので割愛

- 音符を2次元のバイナリ行列（or 1次元のマルチホットベクトル）として表現
  - 直感的でわかりやすく，多声の表現も簡単
  - 冗長であるが，16分音符ごとに量子化すると減らせる
  - 量子化は実演奏とアライメントを取ったものには有効でない



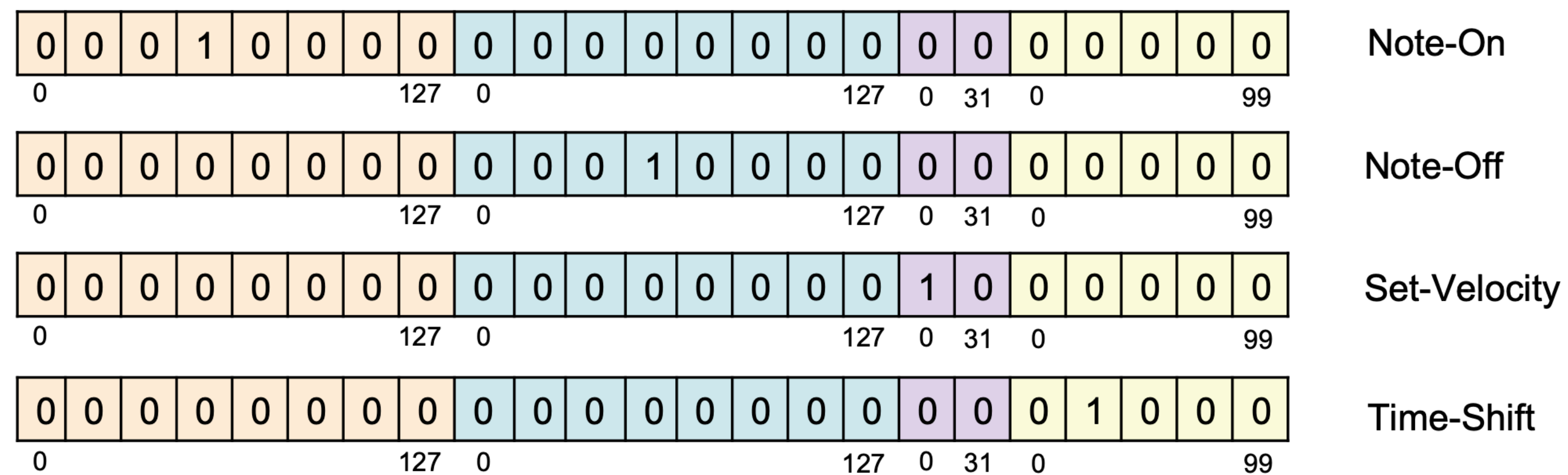
Piano roll



Quantization

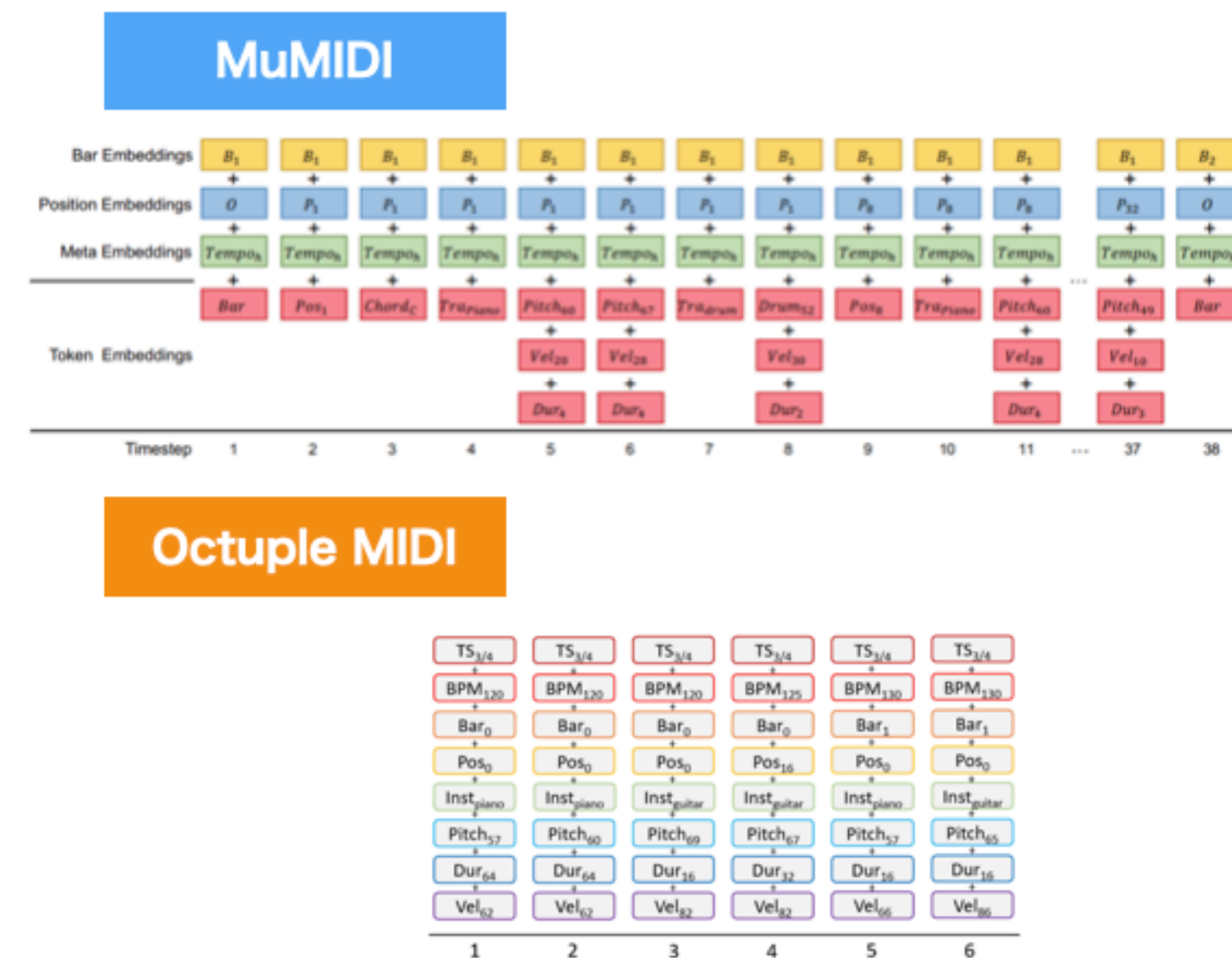
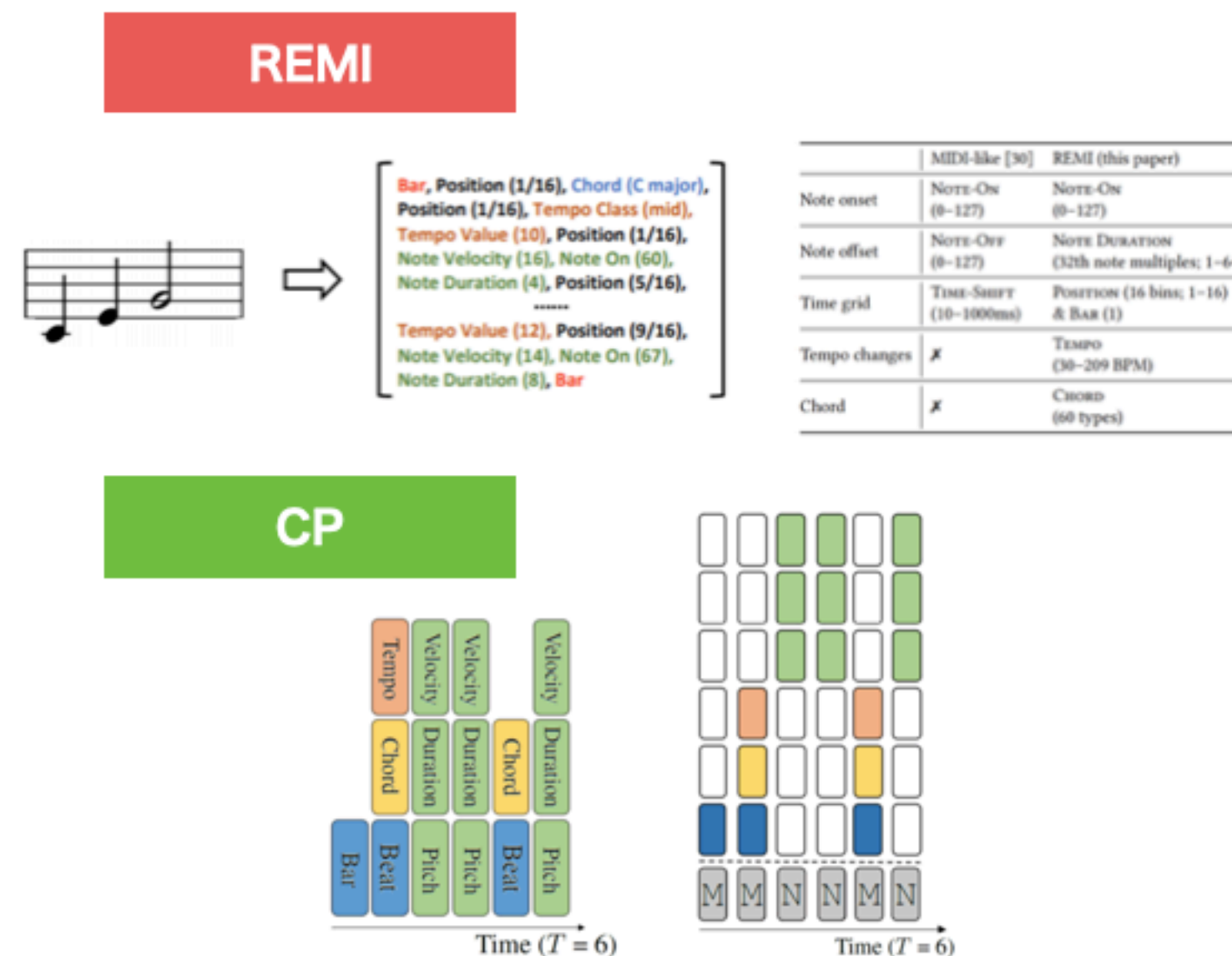


- **MIDIのイベントを388次元のワンホットに押し込む**
  - ノートオン (128音高) + ノートオフ (128音高) + ヴェロシティ (32レベル) + タイムシフト (100シフト : 10ms ~ 1000ms)
  - 多声音楽の表現も容易



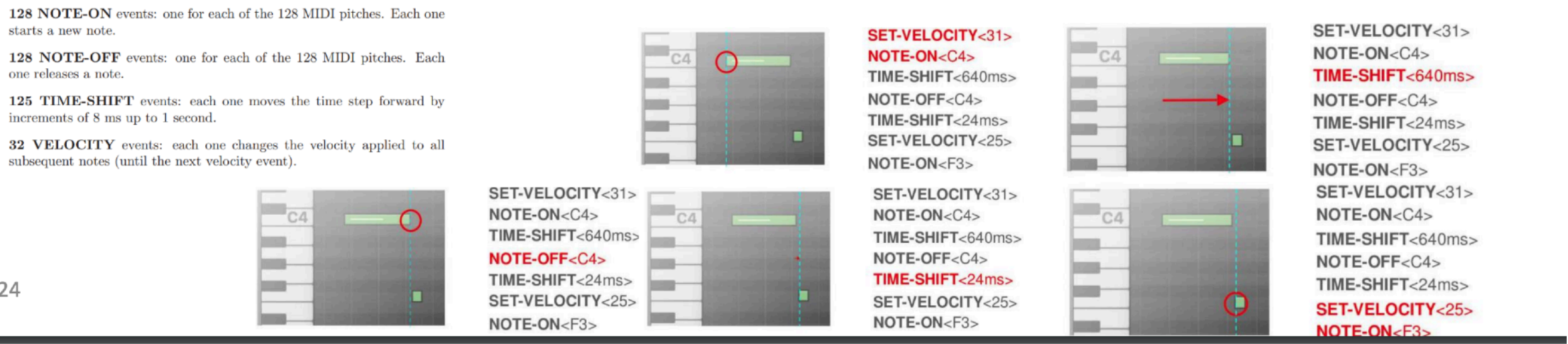


- 音符を音楽要素の情報を保持してパースしたもの
  - 拍子, 調, テンポ, 音符の情報等を保持した系列
  - 情報量が多く, より音楽的に自然な生成が可能に
  - 現在急成長中で, あまり標準化されていない



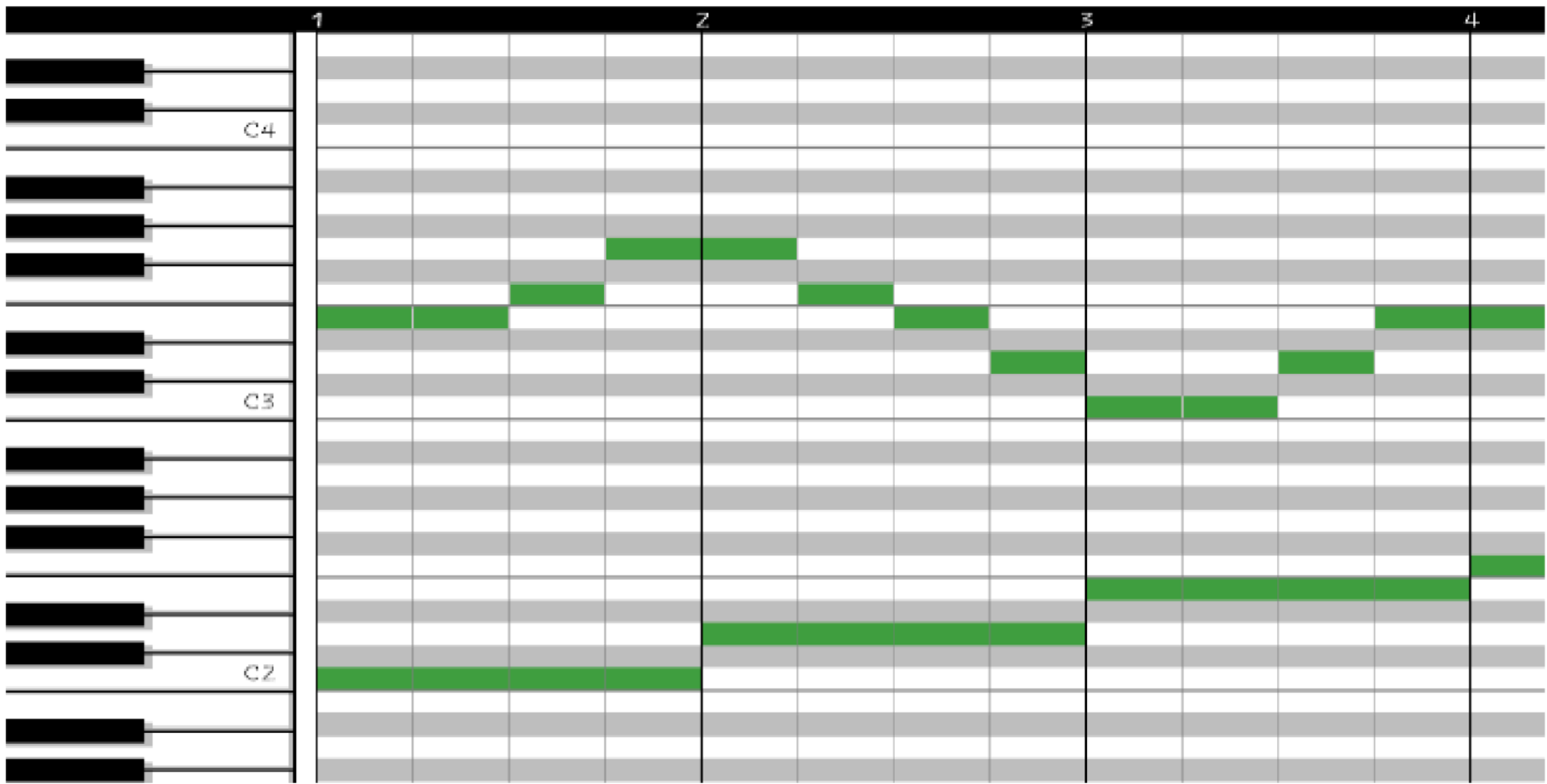
## MIDIトークン

音高・音量・音長  
ノートON/OFFのイベント列



## ピアノロール

音高×時間のバイナリ行列

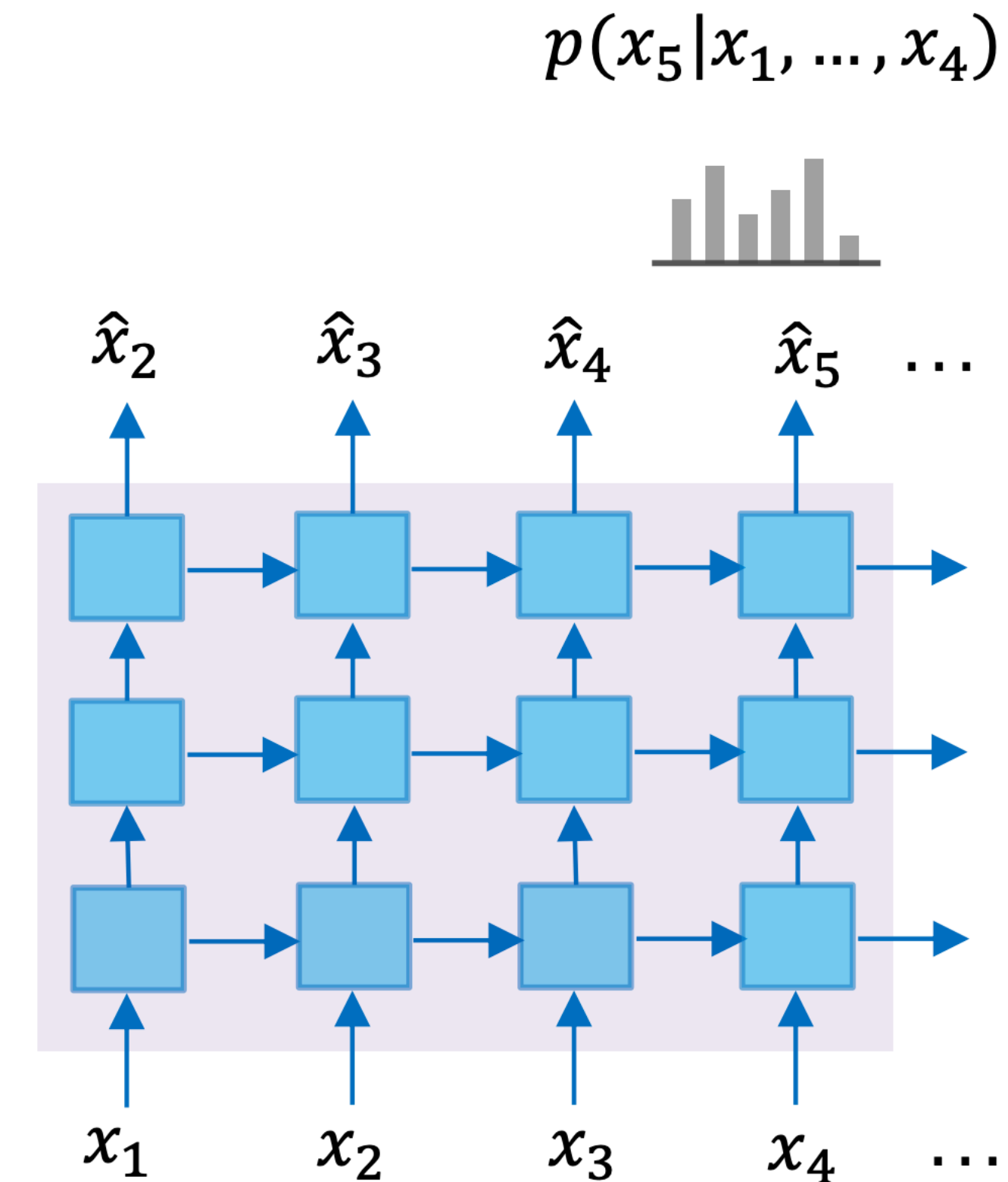


😊：音符ごとの記述（処理で有用なことが多い）  
😓：拍の概念と相性×，同時に多数の音が鳴る場合に効率が悪い，音符の長さがわかりにくい

😊：直感的  
😓：密になる，長い音なのか短い音の連発なのかの区別ができない

# モデル・評価方法

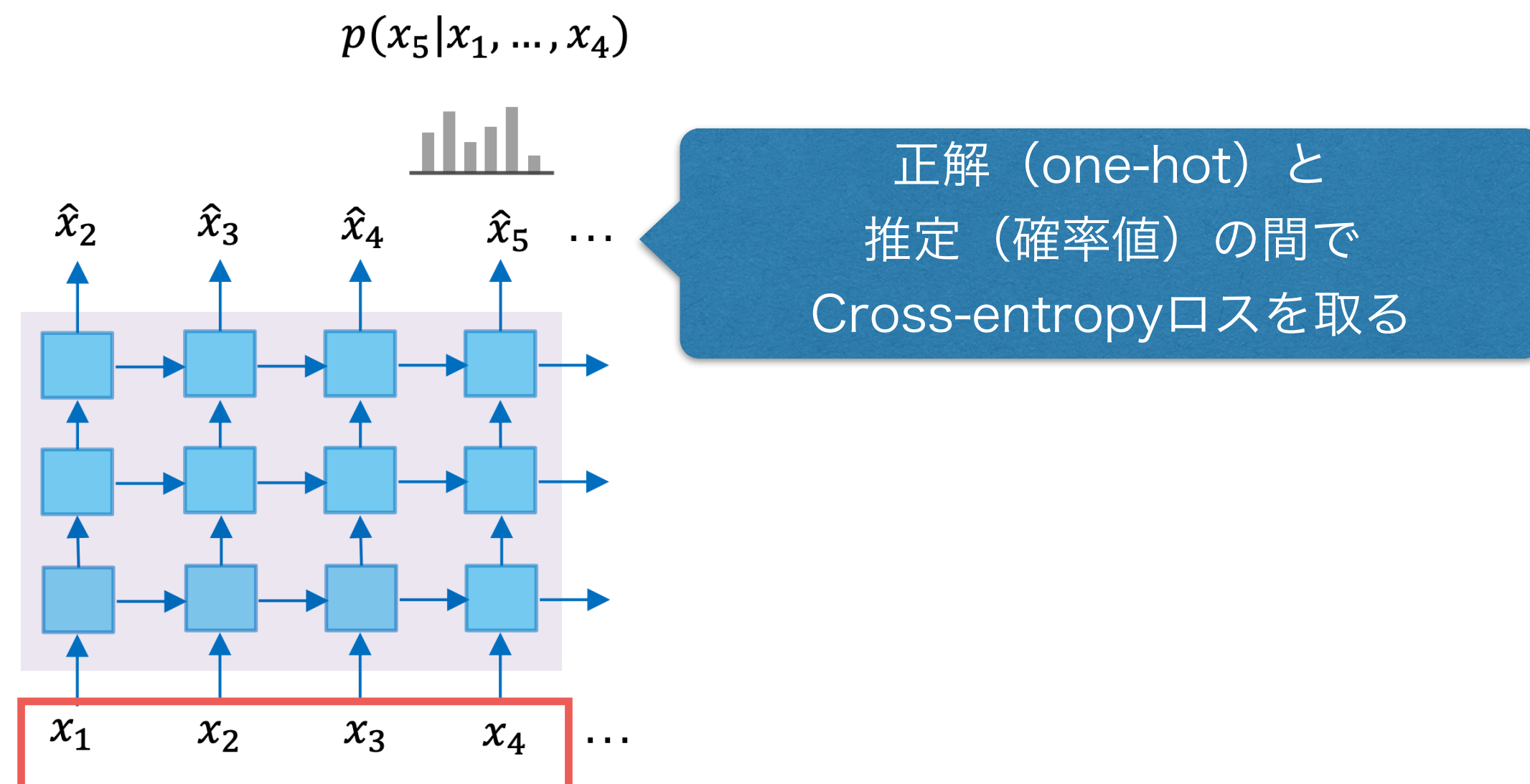
- RNNベースのモデルの代表的なモデル
  - 3層のLSTMの後, softmaxによってどの単語（音符）が次に来るかを随時予測
  - 予測の“分布”を, 正解音符によって最適化していく
- Magenta形式のMIDI表現が入力
- テンポの変更と移調によりデータ拡張を行う



## 訓練時 (train) と推論時 (test) で挙動が異なる

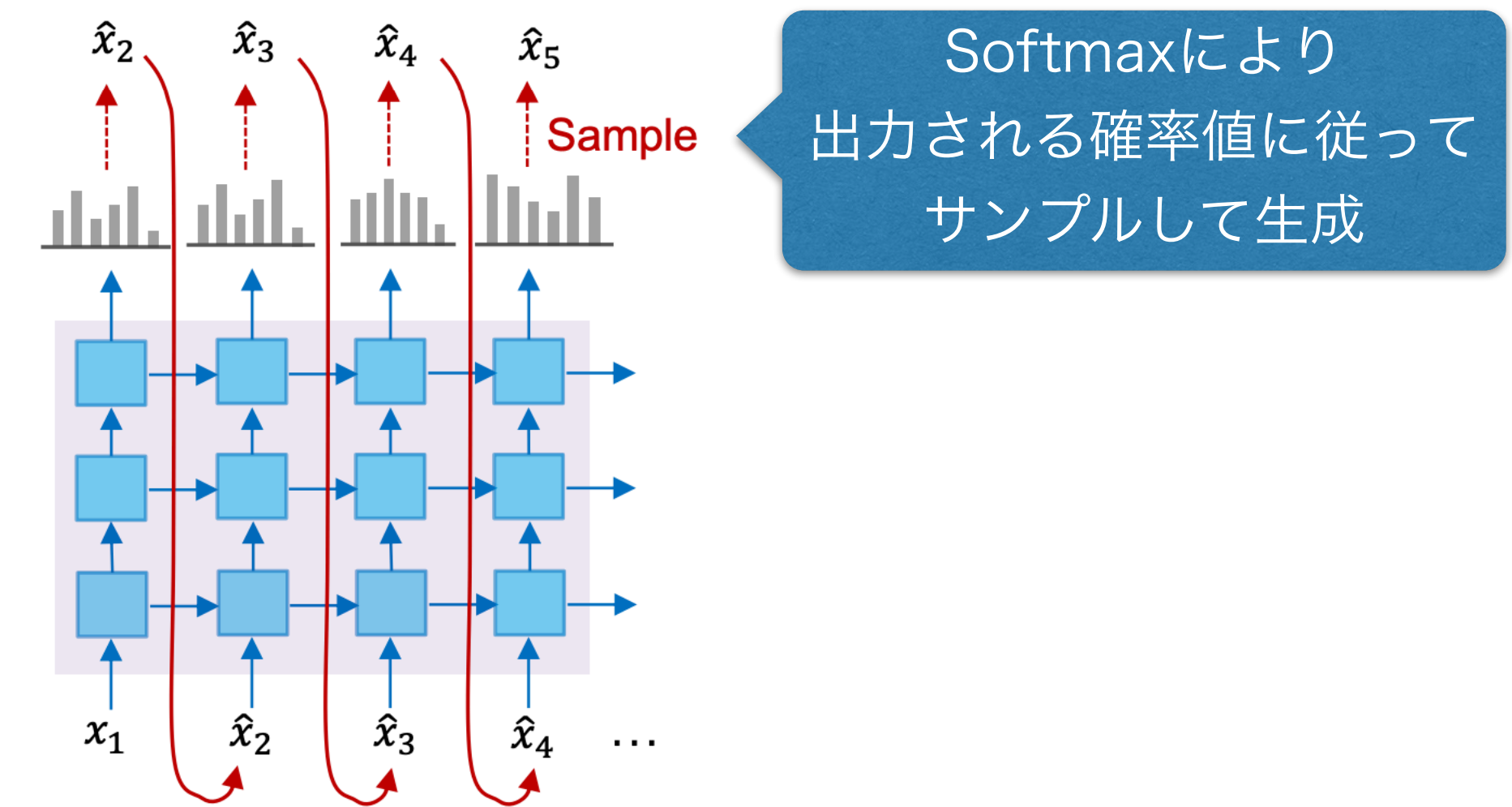
### 訓練時

入力データをそのまま次の入力にする  
(teacher-forcing)



### 推論時

その時点での出力が次の入力になる





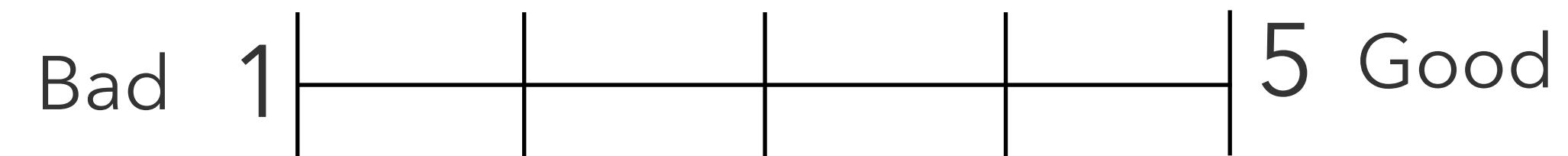
- 定量的評価：
  - パープレキシティ (PPL)：確率の逆数
    - ざっくり言うと「次の予測がどれくらい難しいか」  
= 「モデルが文法を捉えられていない度」
  - 自然言語処理では、一般的に低いほどよい
  - ↔音楽では、一概にそうとは言えない  
(多様性を求めるとPPLは低くなる)

$$P(X) = \prod_{t=1}^T \left( \frac{1}{P_{LM}(x_t | x_{<t})} \right)^{1/T}$$

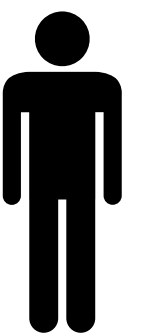
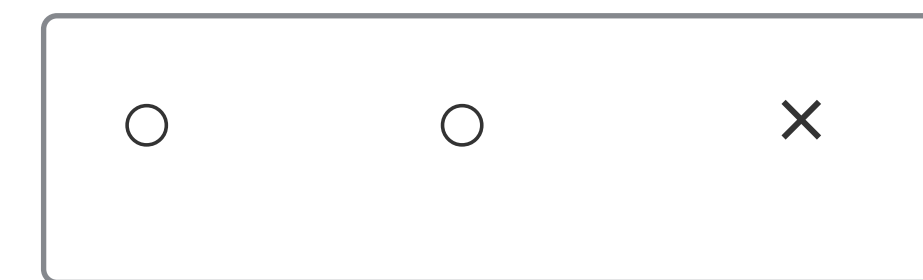
T: 系列長

- 定性的評価

- Mean opinion score (MOS)：被験者実験を行い、5段階のリッカート尺度で評価をとって平均する
- Discrimination test：本物と生成物を混ぜて聴かせ、区別できるかをテスト
- 特定ドメインの音楽生成において行われる  
(例：Deepbach [Hadjeres 17])

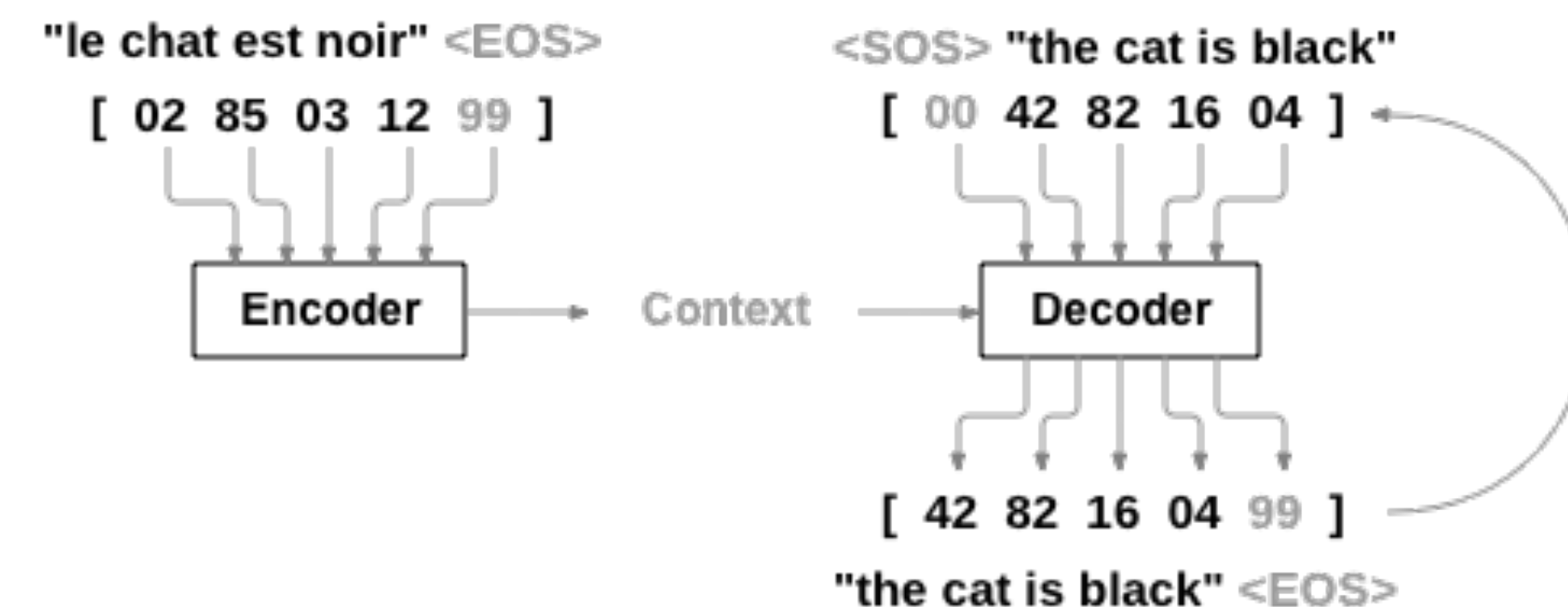
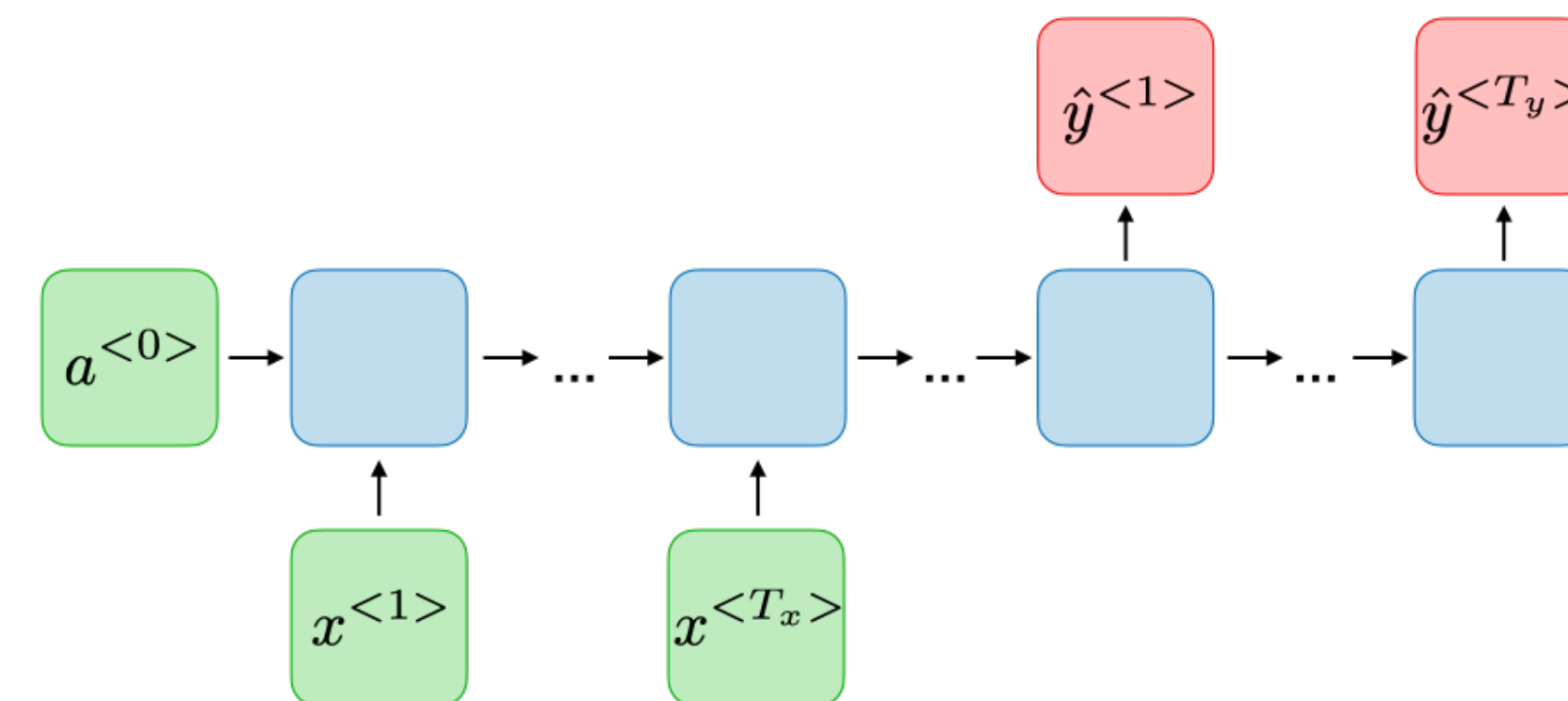


Q:これは本物？



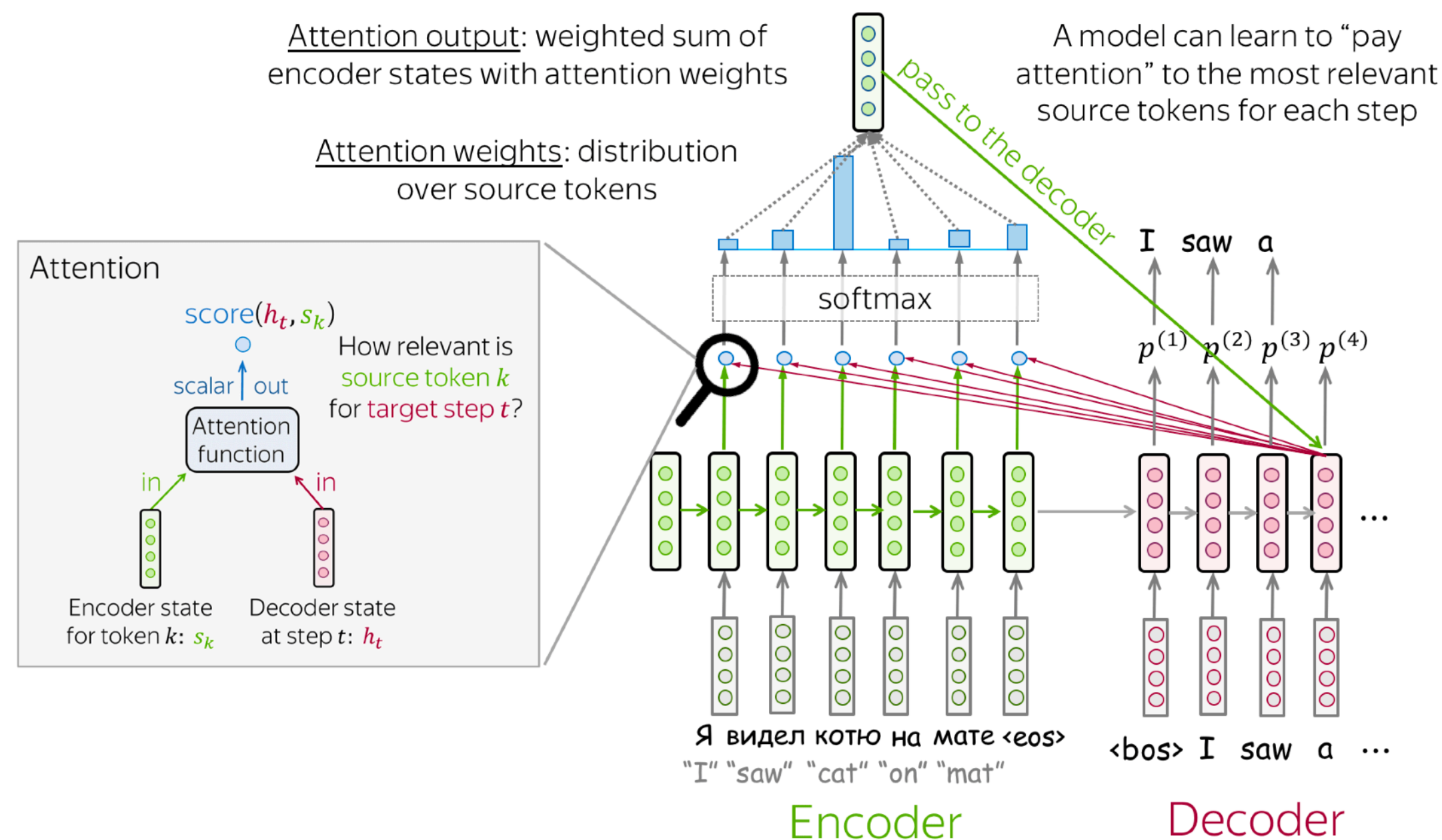
補足

- 入出力が1対1対応しない系列の変換でのモデル
- エンコーダー・デコーダーモデル (一般化)
- 実装上では2つの部分に分ける
  - エンコーダー -> 入力からコンテキストを得る
    - many-to-one
  - デコーダー -> コンテキストから出力を得る (自己回帰形式)
    - one-to-many



[https://pytorch.org/tutorials/intermediate/seq2seq\\_translation\\_tutorial.html](https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html)

- Seq2Seqでは入力を全て固定長のベクトルにしていた
- → 入力の長さが異なろうが全て同じ情報量を保つ
- → Attention（注意機構）を用いて，入力のどこに着目するかを決定



[https://lena-voita.github.io/nlp\\_course/seq2seq\\_and\\_attention.html](https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html)

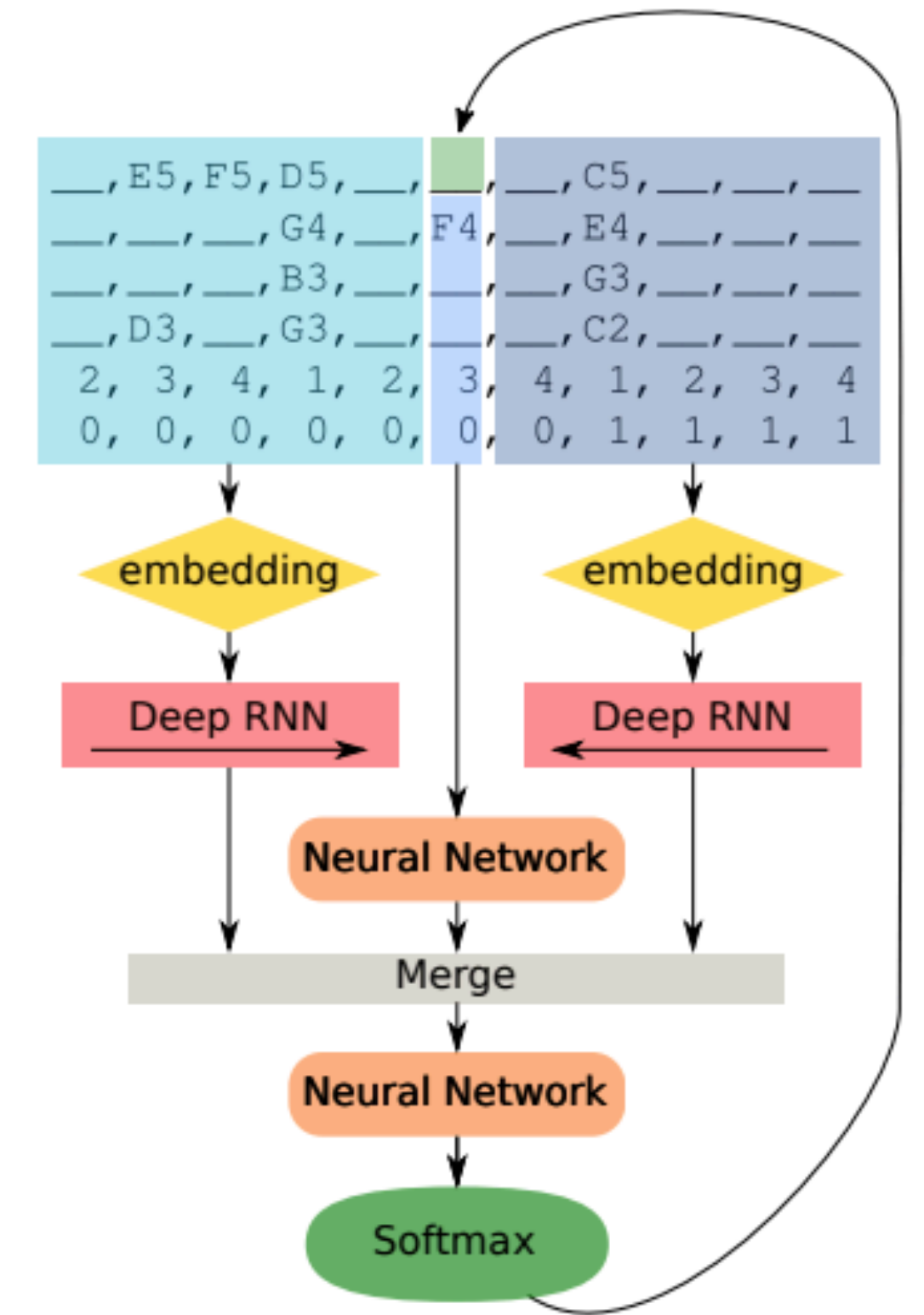


## バッハの賛美歌を自動生成

- Sony CSL発, 音楽のプロ1600人の半分以上が本物と間違える



- 入出力：前までの音の並びから次の音を予測（逆側からも）
- ラベル：予測時点の構成音



モデル