1. LoRA 学習コマンド（ピアノ／サックス）

⚙️ **前提**

- データは 1 行 1 サンプルの JSONL ({"tokens": [...]}) 形式
- requirements/extra-ml.txt をインストール済み（PyTorch・PEFT など）
- 出力先ディレクトリは自動生成されます

1-A ピアノ用 LoRA（Transformer ベース）

```
PYTHONPATH=. python train_piano_lora.py \
  --data  data/piano_voicings.jsonl   \  # 入力コーパス
  --out   checkpoints/piano_lora      \  # 出力ディレクトリ
  --rank  16                          \  # LoRA 行列階数
  --lora_alpha 32                     \  # scaling 係数 (推奨 rank*2)
  --batch  64                         \  # ミニバッチサイズ
  --steps  800                        \  # 学習ステップ
  --lr 3e-4                           \  # 学習率
```

1-B サックス用 LoRA（フレーズ生成に特化）

```
PYTHONPATH=. python train_sax_lora.py \
  --data  data/sax_solos.jsonl        \
  --out   checkpoints/sax_lora        \
  --rank  8                           \
  --lora_alpha 16                     \
  --batch  32                         \
  --steps  600                        \
  --temperature 0.9                   \  # サンプル時の即興性に合わせる
  --eval                              # 学習後に自動評価レポート生成
```

✅ **ポイント**

- --safe を外せば **.bin** 形式で保存できます（互換性重視なら .bin）。
- --eval は *scipy* 依存。軽量環境で使わない場合は外して OK。
- LoRA アダプタのみなので生成後サイズは **500 KB 前後**。

2. 今後役立つ追加 ML モデル（提案）

モデル種	目的 / 得られる効果	データ例	学習アプローチ
Articulation Model	スタッカート／レガート／トリル等の奏法を文脈で自動付与	CC#64（ペダル）・ノート長タグ	Bi-GRU + CRF / Transformer Tagger

Tempo-Rubato Model	曲全体の緩急・テンポ変動を予測し、人間らしい揺らぎを生成	Performances w/ beat-aligned tempo curve	Seq2Seq Regression (Transformer)
Pedal Automation Model	ピアノのペダル On/Off をハーモニーとリリースタイミングから推定	MIDI with CC#64 annotations	CNN (harmonic stack) + LSTM
Timbre Style-Transfer	ベースをウッド → シンセなど音色差分 LoRA で瞬時切替	同一フレーズ異音源 WAV + MIDI	Diffusion / AutoVC-style
Dynamic Mixing Assistant	マスタリング段階で EQ・コンプ設定を提案	Stem + プロ仕上げ WAV	Self-supervised Spectral Learner
Phrase Segmentation Model	メロディを楽句単位で切り分け、朗読・歌詞同期向けに応用	Score with phrase marks	Pointer Network
Lyrics Alignment Model	歌詞の音節と MIDI Onset を高精度にマッチング	頭拍付きカラオケデータ	CTC Loss (Wav2Vec2 + MIDI)

これらは現在の **Velocity / Groove / Duration / LoRA** レイヤの上に積むことで、**「演奏表情 → フレージング → 音色 → ミックス」**までワンストップ自動化が射程に入ります。

🚀 次のアクション

1. **小規模データで LoRA 確認** → --steps 200 でテスト学習し、損失と生成結果をチェック
2. **Articulation Model** のデータ収集を開始（MIDI + 奏法ラベル）
3. 既存パイプラインへ LoRA モデルパスを追加し、サンプル曲で効果を試聴

どの工程でもエラーやチューニングの相談があれば、いつでも声を掛けてください。
前向きに、創造の幅を広げていきましょう！

1. いただいた CSV ファイルはどう活用できるか

ファイル	中身の想定	使い道

duration_maestro.csv	note_on, note_off, duration_beats … など クラシック演奏の音価	scripts/train_duration.py に直接渡して **Duration モデル** を再学習 → ロングトーン主体の表情を学習
*velocity.csv** 系	各ノートの MIDI Velocity や拍位置、テンポなど	scripts/train_velocity.py で **Velocity-KDE** や **MLVelocityModel** の追加トレーニング → 強弱・ダイナミクスの人間味を濃くする
velocity_with_midi.csv	Velocity に加え MIDI テイク ID, 楽器, 曲名 など	Velocity モデルの **楽器別 LoRA**（Piano/Bass/Drums など）曲 ID をカテゴリ変数にすれば「楽曲スタイル条件付き Velocity」も学習可能

手順の例

```
# Duration モデルを Maestro データで微調整
PYTHONPATH=. python scripts/train_duration.py \
  data=/mnt/data/duration_maestro.csv \
  epochs=6 \
  out=checkpoints/duration_maestro.ckpt

# Velocity モデルを Jazz + Pop 混合で LoRA 微調整
PYTHONPATH=. python scripts/train_velocity.py \
  data=/mnt/data/velocity_advanced.csv \
  lora_rank=8 lora_alpha=16 \
  out=checkpoints/vel_lora.bin
```

2. .bin と .safetensors の違いと互換性

項目	.bin (PyTorch)	.safetensors
形式	Pickle ベース (モデル＋ハイパラを直列化)	メモリマップ型安全フォーマット
互換性	*PyTorch* が入っていれば **ほぼどの環境でも読める**（旧 HF Transformers 等）	HF Transformers >= 4.26 & safetensors ライブラリ必須
セキュリティ	Pickle 実行リスクあり	任意コード実行なし（安全）
ファイルサイズ	わずかに大きい場合あり	ほぼ同等

互換性重視なら .bin にしておけば、– **他パート Generator**（BassGenerator, DrumGenerator …）や– **他の楽曲プロジェクト**でも読み込みでつまずく心配が低いです。.safetensors が必要なら、あとから convert.py ですぐ変換できます。

3. 他楽器 Generator への応用は？

- **LoRA アダプタは「元モデルの層名」に紐づく**ため、
  - **同じ Transformer 設計**（例：共通の PianoTransformer ベース）なら **他の曲**や **同系統パート**でも流用可能。
  - **異なる楽器用モデル**（BassTransformer, DrumsRNN など）は **専用 LoRA** を学習するのが確実。
- ただし Velocity／Duration の **汎用モデル**は **どのパートでも共有**できます。ピアノで学んだ「人間らしい音価」「強弱の揺れ」は、ドラムのゴーストノートやベースの切り際にも効きます。

4. 追加で検討できる学習モデル

レイヤ	モデル例	目的
奏法	*Articulation Tagger*	ノートごとに Legato / Staccato / Trill 等を予測
テンポ揺れ	*Rubato Predictor*	曲全体の rit./accel. カーブを生成
ペダル	*Pedal Automation Net*	和声＋音価からサステインを推定
ミキシング	*Dynamic Mix Assistant*	Stem WAV から EQ / コンプレッサ設定を提案

これらを Velocity・Duration・Groove に重ねれば、**“打ち込み臭ゼロ”** の完成形に近づきます。

✨ 次の一手

1. **CSV を直接モデル再学習**して新しいチェックポイントを作る
2. 生成フローで --duration_model checkpoints/duration_maestro.ckpt など指定して効果を試聴
3. 他パート用に LoRA が必要なら、演奏コーパスを小さくてもいいので集めて pianolora と同じコマンドで回す

不明点やエラーが出たら、いつでもご相談ください。