Generating Monophonic Melodies with an End-to-End LSTM Pipeline (MATLAB)

This project focuses on generating **monophonic melodies** using a complete **end-to-end LSTM-based pipeline** implemented in MATLAB.
Starting from raw MIDI files, the system learns melodic and rhythmic patterns and generates new musical sequences using next-event prediction.

The goal of the project was not to build a "black-box" music generator, but to **understand and model melody and rhythm together** in a controlled and reproducible way.

Project Overview

Music generation is formulated as a **sequence modeling problem**, where the model predicts the *next musical event* based on a window of previous events.

The pipeline consists of four main stages:

1. **Data Preparation**
2. **Model Training**
3. **Melody Generation**
4. **Evaluation & Analysis**

All stages are implemented and executed within MATLAB.

Dataset

- **Dataset:** Nottingham Monophonic MIDI Dataset
- **Why monophonic?**
    - Avoids harmonic/polyphonic complexity
    - Allows the model to focus purely on **melodic contour and rhythm**
- **Input format:** MIDI

Preprocessing

- Each MIDI file is converted into:
    - A **pitch sequence**
    - A **duration sequence**
- Melodies with fewer than 16 events are discarded to ensure sufficient context for training.

Unified Pitch–Duration Representation

A key design decision in this project was to **model pitch and rhythm jointly**.

Problem

Models that predict only pitch fail to learn rhythmic structure.

Solution

- Pitch and quantized duration are combined into a **single unified token**
- Each musical event is represented as:

-token = pitch_index + (duration_class - 1) × total_pitch_count

This turns music generation into a **single multi-class classification problem**, rather than predicting multiple features separately.

Duration Quantization

- Raw MIDI durations are continuous and highly skewed
- Durations are quantized into **K = 32 discrete classes**
- Quantile-based binning is used

  This reveals a **class imbalance problem**:

- Short notes appear much more frequently than long notes
- This bias affects model predictions and diversity

Dataset Construction

- Training samples are created using a **sliding window** approach:
  - Input (X): `W` previous tokens
  - Target (Y): the next token

Two Evaluation Strategies

1. **Random Window Split**

   - Windows are randomly split
   - Can cause data leakage
   - Optimistic performance

2. **Song-Level Split (80/10/10)**

   - Entire songs are separated into train/val/test
   - Much stricter and more realistic
   - Used for final evaluation

Model Architecture

The model follows a standard **language-model-style LSTM architecture**:

- Sequence Input
- Embedding Layer
- LSTM Layer
- Fully Connected Layer
- Softmax
- Classification Output

The implementation uses **MATLAB Deep Learning Toolbox**.

Training Details

- **Optimizer:** Adam
- **Learning Rate:** 1e-3 (constant)
- **Batch Size:** 128
- **Epochs:** 20
- **Hardware:** Single GPU
- **Training Time:** ~39 minutes

Performance

- Validation accuracy (window-level split): **~59%**
- Validation accuracy (song-level split): **~38–39%**

> The lower song-level accuracy better reflects real-world generalization.

Ablation Study: Context Window Length

Different window sizes were tested: `W = 8, 16, 32, 64`

Observation:

- Accuracy remains in a narrow range
- Increasing context alone does not significantly improve performance
- Suggests limitations due to:
  - Model capacity
  - Class imbalance
  - Representation bottlenecks

Final choice: **W = 25**

Melody Generation

- Generation is performed **autoregressively**
- The model:
  1. Takes a seed sequence
  2. Predicts the next token
  3. Appends it and repeats

- Generated tokens are decoded back into:
  - Pitch
  - Duration
- Output is exported as a **MIDI file**
- Audio rendering is done using MATLAB’s Audio Toolbox

Evaluation

Qualitative

- Generated melodies show:
  - Plausible pitch transitions
  - Coherent rhythmic groupings
- However:
  - Lack long-term musical structure
  - Tend to repeat local patterns

Quantitative

- Pitch distributions of generated melodies are compared with training data
- The model learns the correct pitch range
- **Mode collapse** is observed:
  - Reduced diversity
  - Overuse of high-probability notes

Pipeline Robustness Test

An external MIDI file (not from the Nottingham dataset) was processed using the same pipeline.

Result:

- Pitch and duration distributions align well with learned representations
- Confirms **pipeline compatibility**, not generalization performance

---

Limitations

- Weak long-term musical structure
- Strong class imbalance in duration modeling
- Reduced output diversity
- Performance plateaus with single-layer LSTM

Future Work

- Deeper LSTM architectures or Transformer-based models
- Class-weighted loss functions
- More expressive event representations

- Advanced sampling strategies (top-k, nucleus sampling)
- Human listening tests and musicality metrics

Reproducibility

The entire pipeline can be executed sequentially:

-prepare_nottingham_sequences -createDataset -trainLSTMModel -generateMelody -evaluateModel