Genre-to-Jazz Music Conversion

Coline Berthe, Claudia Brambilla, Mattea Busato

June 2, 2025

1 Abstract

This project investigates the use of deep generative models to synthesize jazz music from genre-diverse symbolic data in the form of MIDI files. Our primary goal is to model the harmonic, rhythmic, and stylistic features of jazz to enable genre-to-jazz conversion and the autonomous creation of novel jazz compositions.

We explore three complementary model families: Recurrent Neural Networks (RNNs) for local event-based prediction, Variational Autoencoders (VAEs) for learning latent representations of jazz phrasing, and Transformer architectures for capturing long-range musical dependencies. Each architecture is trained and evaluated on a curated dataset of jazz and non-jazz MIDI sequences, preprocessed to extract structured representations of notes and chords.

Our methodology includes statistical exploration of event distributions, autocorrelations, and repetition patterns to inform model design choices. We assess model performance both quantitatively (using pitch entropy, note density, and repetition rate) and qualitatively through listening tests.

Results show that while RNNs are effective at modeling short-term structure, Transformers seem to yield more coherent global compositions. VAEs excel in generating stylistically rich and interpolatable musical phrases, suggesting that hybrid approaches may offer the most promising direction for future research.

2 Introduction

2.1 Background

Deep learning has significantly influenced the field of music generation, where the goal is to create original musical content through models trained on symbolic or audio data. Unlike images or plain text, music poses unique challenges: it is highly structured across time, often polyphonic (multiple notes at once), and involves dependencies both within and across instruments. These properties require models that can handle sequential patterns, hierarchical structure, and multi-output events.

Initially dominated by Recurrent Neural Networks (RNNs), the field has more recently embraced architectures such as Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and Transformers, inspired by advances in natural language processing. Beyond entertainment, music generation holds promise in broader applications such as music therapy, education , and even content generation for games and film scoring. These developments position automatic music generation not just as a technical curiosity, but as a tool for creativity and enabling new human-AI collaborations.

2.2 Research Question

This work investigates whether deep generative models can learn to transform music from diverse genres into stylistically coherent jazz compositions. Jazz is chosen as the target domain due to its improvisational nature, rhythmic complexity, and harmonic richness, which make it a challenging and expressive style to model. Focusing on jazz allows us to explore the ability of generative systems to internalize and reproduce high-level musical structure.

3 Dataset

The dataset used in this project is PiJAMA (Piano Jazz with Automatic MIDI Annotations), available at the following link: https://zenodo.org/records/8354955. It contains over 2,700 solo jazz piano performances transcribed into MIDI format from audio recordings using automatic transcription methods. The collection spans more than 200 hours of music by 120 pianists and includes both studio and live performances across 244 albums. The dataset's scale, stylistic diversity, and symbolic representation make it well-suited for generative modeling tasks.

3.1 Processing of the MIDI format

The Musical Instrument Digital Interface (MIDI) is a symbolic representation of music that encodes information about notes, timing, velocity, instruments, and other performance parameters. Unlike audio waveforms, which directly encode sound, MIDI files consist of structured messages that describe how music should be performed.

To enable efficient access and manipulation of the data, we parsed all MIDI files into structured Python objects using the `music21` and `pretty_midi` libraries. From these, we extracted relevant musical features and stored them in tabular format (CSV). During parsing, we applied consistency checks to remove corrupted files, flattened polyphonic structures into event sequences, and converted symbolic durations (e.g., fractions) into floating-point values for compatibility with downstream models. Special care was taken to ensure temporal alignment between musical attributes such as note onsets, durations, and chord groupings.

The following table summarizes the parameters extracted from each MIDI file and stored in our dataset:

Field	Description	Type	Purpose
`file_name`	Name or path of the MIDI file.	`str`	Used to identify or retrieve the file.
`instrument`	Instrument(s) used in the track, such as Piano, Violin.	`List[str]`	Provides information about timbre and orchestration.
`notes`	MIDI pitch values (0–127), where 60 = Middle C.	`List[int]`	Core melodic and harmonic content of the piece.
`chords`	Detected chords, such as Cmaj, Am7.	`List[str]`	Summarizes harmonic content; derived from groups of notes.
`velocities`	Velocity (0–127) of each note, representing how hard the note was played.	`List[int]`	Expressiveness and dynamics of the performance.
`durations`	Duration of each note, in seconds or beats.	`List[float]`	Determines how long each note is held.
`offsets`	Start time of each note from the end of the previous one.	`List[float]`	Determines note onset timing and temporal structure.
`tempos`	Tempo in BPM (beats per minute), possibly varying over time.	`List[float]` or dicts	Allows mapping between time and beats.
`time_signatures`	Time signature(s) of the piece (e.g., 4/4, 3/4).	`List[str]` or dicts	Indicates rhythmic structure and measure groupings.
`key_signatures`	Musical key (e.g., C major, A minor).	`List[str]` or dicts	Represents the tonality or mode of the piece.

Table 1: Description of MIDI file fields used in the project dataset.

4 Exploratory Data Analysis and Cleaning

We performed a series of analyses to understand the structure and statistical properties of the dataset prior to model training. First, we examined the distribution of musical features such as pitch, velocity, duration, and note density. The pitch distribution revealed a concentration of notes around specific registers (e.g., C4, F4, G4), highlighting a strong tonal center and informing the construction of a vocabulary for event tokenization.

We also investigated the repetition and transition patterns across musical events. Bigram analysis showed that immediate repetitions of the same note are extremely frequent, suggesting that local dependencies are a prominent feature of jazz piano performances. To explore temporal structure more formally, we computed the autocorrelation function (ACF) of note sequences, which revealed

significant correlation up to 10–15 time steps. This finding directly motivated our use of Recurrent Neural Networks (RNNs), which are well-suited to model short-term sequential dependencies.

At the same time, we observed recurring harmonic structures over longer time spans, as well as motif repetitions and returns to the tonic key. These long-range dependencies suggested that self-attention mechanisms, such as those in Transformer models, could be valuable for capturing high-level musical structure.

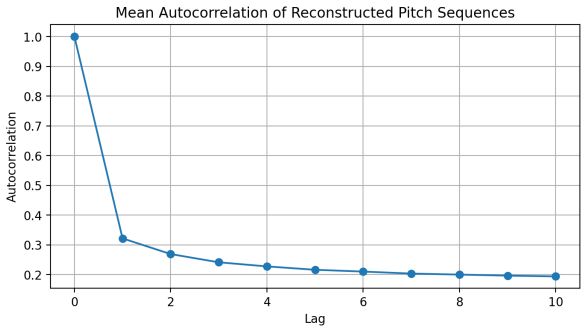

Figure 1: Autocorrelation of event sequences.

5 Models and Training

5.1 RNNs

5.1.1 Justification for using RNNs

Recurrent Neural Networks (RNNs) provide a natural and well-established approach for modeling temporal sequences, making them a suitable baseline for symbolic music generation.

Our exploratory analysis confirmed that jazz MIDI sequences exhibit strong short-term dependencies, including frequent repetition of the same note and consistent local rhythmic motifs. Autocorrelation analysis of event sequences further revealed a temporal memory effect over several time steps, which RNNs are specifically designed to capture through their internal state transitions.

5.1.2 Architecture Overview

We frame music generation as a next-event prediction task: given a sequence of past musical events (notes or chords), the model predicts the next likely event. To prepare the data, each MIDI file is tokenized into a sequence of events using a vocabulary built from all unique notes and chords. We then generate training examples by sliding a window of fixed length over each sequence, pairing input sequences with their next event as targets.

The model consists of an embedding layer, a single-layer recurrent unit (RNN or LSTM), and a fully connected output layer followed by a softmax. The output is a probability distribution over the event vocabulary. This setup allows the model to capture local sequential patterns and produce coherent event continuations, in line with the short-term dependencies observed in our exploratory analysis.

5.1.3 Training Objective

The RNN model is trained to minimize the cross-entropy loss between the predicted probability distribution and the true next event token at each time step. Since music generation is framed as a multiclass classification problem—predicting one token from a vocabulary of possible events—cross-entropy provides a natural and effective objective.

This formulation aligns with the autoregressive nature of the task and enables probabilistic generation: at inference time, new music can be generated by sampling from the model's output distribution conditioned on a given input sequence.

5.1.4 Hyperparameter Selection and Justification

To understand how the model's performance depends on sequence length and hidden size, we conducted a grid search over several configurations, using 5-fold cross-validation to ensure robustness. We evaluated sequence lengths of 8, 16, and 32, reasoning that musical structure might be captured at different temporal scales. This also allowed us to assess the model's ability to learn both very short and moderately long dependencies.

For the hidden size, we tested values in {64, 128, 256} and retained only one recurrent layer to reduce overfitting and ensure interpretability, given the limited complexity of the input representation. The validation results helped select the best-performing configuration for final training and generation.

5.1.5 Results

To evaluate the RNN's generative capabilities, we sampled 100 new sequences using random inputs from the test set. The generated outputs were syntactically valid and locally coherent, but musically simplistic and often repetitive. This is consistent with the model's tendency to overfit short-term dependencies while lacking a sense of larger structure or phrasing.

We also tested the model on tokenized representations of well-known melodies, including *Poker Face*, *My Heart Will Go On* (Titanic), and *Happy Birthday*. In these cases, the generated continuations occasionally preserved fragments of the original melody, but overall failed to maintain harmonic progression or stylistic intent. This highlights the limitations of a purely event-based RNN trained on a single-instrument setup. The fixed-length representation of events and absence of expressive timing further restrict the model's ability to emulate the richness of real jazz performance.

5.2 VAEs

5.2.1 Justification for Using VAEs

While RNNs are effective for modeling local temporal dependencies in music, they lack an explicit mechanism for global control or stylistic interpolation. Variational Autoencoders (VAEs), in contrast, learn a smooth and structured latent space that facilitates creative manipulation of musical ideas. This makes them particularly suitable for jazz, a genre defined by variation, improvisation, and expressive deviation from strict regularity.

By encoding musical phrases into a latent distribution, the VAE framework allows stylistic variation by sampling from the latent space which introduces meaningful diversity in generated outputs. It also enables latent interpolation, the smooth transitions between musical motifs should be achieved by interpolating between latent vectors. Finally, the multi-modal encoding using separate encoders for musical events and durations allows the model to better capture rhythmic expressivity and harmonic structure.

5.2.2 Architecture Overview

The model receives two inputs: a sequence of musical events (multi-hot encoded notes or chords) and their corresponding durations. These are processed through two separate encoder branches:

- **Event Encoder:** The input event sequence is flattened and passed through two fully connected layers with ReLU activations. This captures harmonic and structural information.
- **Duration Encoder:** The input duration sequence is fed into a smaller MLP to encode rhythmic features.

The outputs of the two encoders are concatenated and mapped into a latent Gaussian distribution via two linear layers producing the mean vector μ and log-variance vector $\log \sigma^2$. A latent code z is sampled using the reparameterization trick:

$$z = \mu + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

The decoder has two heads:

- **Event Decoder:** Maps z to a multi-hot vector representing the predicted next event, using sigmoid activation.
- **Duration Decoder:** Maps z to a scalar value representing the predicted duration of the next event.

This modular design enables the model to jointly learn pitch and rhythm while preserving their distinct characteristics.

5.2.3 Training Objective

The model is trained to minimize a total loss composed of three terms:

$$\mathcal{L} = \text{BCE}(\hat{x}_{\text{event}}, x_{\text{event}}) + \text{MSE}(\hat{x}_{\text{dur}}, x_{\text{dur}}) + \text{KL}(q(z|x) \parallel \mathcal{N}(0, I))$$

- **Binary Cross-Entropy (BCE)** is used for reconstructing the next event vector, reflecting the multi-label nature of chords or simultaneous notes.
- **Mean Squared Error (MSE)** measures the error in duration prediction.
- **Kullback-Leibler (KL) Divergence** regularizes the posterior distribution over z toward a standard Gaussian prior, promoting a smooth and continuous latent space.

To improve training dynamics, we employ **KL annealing**, gradually increasing the weight of the KL term β from 0 to 1 over the first 10 epochs. This allows the model to focus initially on accurate reconstruction and later shift toward learning a structured latent space.

5.2.4 Hyperparameter Selection and Justification

We explored several configurations to balance reconstruction fidelity, generalization, and training efficiency. The final choices are based on empirical evaluation and Optuna-based tuning. A latent dimension of 16 captured sufficient structural detail without overfitting. Larger dimensions increased reconstruction accuracy but degraded generation stability. The optimal sequence length 16 proved sufficient to capture local jazz motifs and rhythmic cells while avoiding excessive memory usage. The hidden size selected was 256 as a middle ground between expressive capacity and computational cost. The optimal learning rate identified by Optuna was 0.0073. It is higher than usual but yielded fast, stable convergence when combined with KL annealing. These hyperparameters reflect a trade-off between model complexity and the improvisational characteristics we aim to model in jazz sequences.

5.2.5 Results

The trained VAE demonstrates strong generative capacity for jazz-style sequences regarding their durations but seemed to have a hard time generating proper sequences of events. As witnessed when comparing our results with the original files of the popular songs, the events had notes in common but did not sound in any way harmonic. This is probably due to the lack of parameters we imposed to the model. Music is way more than a sequence of notes or chords and its corresponding duration. Other than these qualitative remarks, we retrieved some quantitative evaluation measures for both VAE models, without using KL annealing and with it. The KL annealing model was trained on a smaller dataset but overall the results are quite comparable. The binary cross entropy loss is the main loss component of the model. This indicates that the model really struggled reconstructing the correct multi-hot event vector. A value around 6 is not abnormal given the sparsity and multi-label nature of the output (chords or overlapping notes). Lower values would have however indicated a more accurate pitch reconstruction. The low KL value indicates that the model behaves almost like a deterministic autoencoder. To encourage better latent space usage, future models could increase the KL weight more aggressively during training.

Regarding duration, the prediction is quite good with a low mean squared error.

Overall, the model reconstructs durations better than it does the events. The KL divergence annealing does not seem to particularly improve the model.

5.3 Transformers

5.3.1 Justification for Using Transformers

As a final step of this project, we briefly explore the use of Transformer models. This choice is motivated by the Transformer's ability to model long-range dependencies through self-attention mechanisms. This feature is particularly relevant in music, where structural patterns often extend across multiple sections. In fact, transformer models are able to access any part of the input sequence directly. This architectural property makes them well-suited for capturing complex, non-local musical dependencies and hierarchical structures typical of jazz.

5.3.2 Tokenization, Architecture overview and Training

First, we represented each MIDI track in the dataset as a sequence of timed musical events: single notes (integers) or chords (lists of integers), paired with their corresponding durations. To feed these data into the model, we developed a tokenization scheme in which each event is converted into two tokens.

- A `NOTE_x` or `CHORD_x_y_z` token for pitch-related information.
- A `DUR_t` token for the associated duration, quantized to three decimal places.

We constructed a vocabulary containing all unique tokens in the training set, and transformed each tokenized song into a flat sequence of alternating note/chord and duration tokens. This encoding ensures that both pitch and rhythmic content are represented in a format compatible with standard Transformer inputs.

The model is trained to predict the next musical event, as a tuple of note/chord and duration, given a fixed-length context. Each input consists of $2 \times$ `context_length` tokens, and the target outputs are the subsequent note and duration tokens. We used a dual-head output structure, with separate softmax layers for predicting pitch and duration. The total loss is computed as the sum of cross-entropy terms for both heads.

Training the model on the full dataset proved to be computationally demanding. To manage this constraint, we opted for a sampling-based approach: instead of processing entire tracks, we extracted short, fixed-length subsequences of musical events from all songs in the dataset. This strategy preserves stylistic and structural diversity while significantly reducing memory and time requirements, making it well-suited for a baseline experiment.

The resulting model serves as an initial benchmark to evaluate the Transformer's capacity to learn from symbolic jazz data. Given the limited scope of this experiment, we selected a lightweight configuration that balances simplicity and expressiveness, avoiding hyperparameter tuning. The goal of this setup is not to achieve optimal performance, but to assess whether the architecture can begin to capture meaningful musical structure under constrained conditions.

5.3.3 Results and Observations

Although our exploration of Transformer models was limited, we observed meaningful learning behavior even in this constrained setting, suggesting that the model was able to learn some predictive relationships between tokens.

To qualitatively assess the model, we conducted music generation by iteratively sampling predictions from the model, one event at a time. Starting from an initial seed sequence, we repeatedly asked the model to predict the next note or chord along with its duration, and appended each prediction to the sequence before feeding it back as input. This autoregressive procedure was used in two settings: first, we used the initial segments of ten training tracks as seeds, then, we experimented with tokenized versions of well-known external songs such as *Final Countdown* and *Back in Black*. Note that, since the vocabulary was constructed from the training set, we could only generate from seeds whose notes, chords, and durations were present in the token vocabulary.

In both settings, the generated sequences initially reflected some characteristics of the seed. However, as generation proceeded, the outputs, while often maintaining a jazz-like tone, gradually began to sound more rhythmically unstable.

Despite these limitations, the fact that the generated outputs were not entirely random and often musically plausible over short spans indicates the promise of Transformer models in this domain. Future work should explore deeper models, longer training schedules, and conditioning techniques to fully realize their potential in symbolic jazz generation.

6 Findings and Limitations

Our project explored the application of RNNs, VAEs, and Transformers for genre-to-jazz music generation and revealed distinct strengths and weaknesses across model families. Recurrent Neural Networks (RNNs) demonstrated a strong ability to capture short-term dependencies in music, producing locally coherent sequences that were structurally valid. However, their outputs often lacked higher-level musical structure, such as harmonic progression or stylistic continuity, limiting their usefulness in generating jazz with authentic phrasing.

Variational Autoencoders (VAEs), by contrast, showed promise in learning smooth latent representations that enabled variation and interpolation. The use of separate encoders for event sequences and durations improved rhythm modeling, resulting in low reconstruction error for note durations. Nonetheless, the model struggled to reconstruct polyphonic textures and harmonically plausible event sequences, likely due to the sparse and multi-label nature of the output representation. Despite the introduction of KL annealing, the low KL divergence indicated minimal latent space usage, which restricted the model's generative flexibility.

Transformer models, although only briefly explored, emerged as a compelling direction due to their ability to model long-range dependencies via self-attention. Even with limited training and a simplified architecture, Transformers were able to maintain stylistic features of seed sequences over short spans and produced musically plausible continuations. However, the outputs often became rhythmically unstable over time, and the lack of conditioning or deeper architectures limited overall quality.

Several limitations affected the scope of our findings. First, the symbolic representation we used—consisting solely of note/chord events and durations—excludes important musical features such as articulation, dynamics, and expressive timing, which are critical to capturing the nuance of jazz performance. Second, the dataset, while large and stylistically diverse, is derived from automatic transcriptions, which may introduce noise or inconsistencies. Additionally, the lack of aligned pairs for genre-to-jazz transformation constrained our ability to supervise stylistic conversion directly. Finally, due to computational constraints, our models were trained on smaller subsets of the data and with limited hyperparameter tuning. These choices, while necessary, likely restricted the models' capacity to learn deeper musical abstractions. Moreover, the evaluation process, which relied heavily on subjective listening, lacks formal rigor, and future work should incorporate structured human evaluation or musicological analysis to better assess quality.

Despite these limitations, our results suggest that a hybrid approach—combining the local sensitivity of RNNs, the stylistic flexibility of VAEs, and the structural awareness of Transformers—may offer the most promise for future jazz generation systems.

Link to the GitHub Repository: https://github.com/matteabusato/Jazzifier