Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Fachbereich Informatik,
Kommunikation und Wirtschaft
Studiengang Wirtschaftsmathematik
HTW Berlin

CTC vs. Seq2Seq Learning: Two Approaches for Automatic Music transcription

Deep Learning Seminar Report

by
Phillip Olshausen, Livia Kastrati

submitted to
Prof. Alla Petukhina

Berlin, February 20, 2026

Contents

1 Introduction

1.1 Motivation

Automatic Music Transcription (AMT) aims to convert an audio recording into a symbolic representation such as a MIDI-like sequence of note events (pitch, onset time, and offset/duration). While monophonic transcription reduces to tracking a single pitch contour, *polyphonic* AMT must disentangle multiple simultaneously sounding notes, overlapping harmonics, and rapid temporal changes. The mapping from waveform to discrete note events is therefore highly underdetermined: different pitch combinations may produce similar spectral evidence, and acoustic cues for note boundaries are often blurred by sustain, reverberation, and performer variation.

Despite these challenges, reliable AMT is a relevant part of music information retrieval, education technology, digital musicology, content-based recommendation, and downstream generative modeling. From a machine learning perspective, AMT represents a demanding instance of *sequence transduction*: a high-dimensional continuous input must be mapped to a structured discrete output sequence under temporal uncertainty and strong combinatorial constraints.

Two core modeling challenges arise:

1. **Acoustic modeling:** extracting pitch-relevant information from time–frequency representations under timbral and recording variation.

2. **Temporal alignment and structure:** determining when note events occur and how they relate sequentially.

Modern deep learning systems typically address acoustic modeling with convolutional and recurrent encoders operating on time–frequency features, and they address temporal alignment using either frame-synchronous objectives (e.g., CTC) or autoregressive generation with attention.

1.2 From Frame-Wise Classification to Event-Based Generation

Early neural AMT systems commonly formulated transcription as frame-wise classification, predicting pitch activity at each time frame of a spectrogram representation. This formulation is closely tied to datasets such as MusicNet (Thickstun et al., 2017b), where note annotations are aligned to audio at fine temporal resolution. A prominent direction in piano transcription further emphasized the importance of onset modeling, demonstrating that explicit onset objectives significantly improve timing accuracy (Hawthorne, 2018).

Within this frame-synchronous paradigm, **Connectionist Temporal Classification (CTC)** provides a principled way to train sequence models without requiring exact frame-level alignment (Graves et al., 2006). CTC introduces a blank symbol and marginalizes over monotonic alignments between input frames and output labels. In our CTC formulation, each encoder time step t produces a distribution over MIDI pitch IDs plus a `BLANK` token. The model outputs a fixed-length sequence of frame-wise predictions, and decoding collapses repeated labels and removes blanks. This yields a monotonic pitch sequence aligned to the acoustic frames.

More recently, AMT has increasingly been cast as a *conditional sequence generation* problem. Instead of predicting frame-wise pitch states, the model generates a stream of discrete musical events (e.g., time shifts, note-on, note-off tokens). Encoder–decoder architectures with attention learn soft alignments between acoustic representations and symbolic outputs (Sutskever et al., 2014; Bahdanau et al., 2015). This paradigm naturally accommodates polyphony, explicit note durations, and variable-length outputs. Transformer-based token models further reinforce this framing, demonstrating competitive multi-task transcription via unified event vocabularies (Gardner, 2021).

These two paradigms—frame-synchronous CTC and event-based Seq2Seq embody different inductive biases. CTC enforces monotonic alignment and tends to train stably but does not explicitly encode event grammar or note duration structure. Seq2Seq with attention can model richer symbolic dependencies but is typically more data-intensive and sensitive to decoding design and exposure bias (Bengio et al., 2015).

1.3 Research Question

This project investigates the following question:

> **How do frame-synchronous CTC decoding and attention-based Seq2Seq event generation behave when transcribing polyphonic classical excerpts on MusicNet under constrained data and compute?**

Rather than pursuing a leaderboard-style comparison, our goal is to understand the *behavioral differences, strengths, and failure modes* of both approaches when implemented within a unified pipeline and evaluated under identical preprocessing and encoder architecture.

1.4 Methodological Overview

We construct a unified transcription pipeline with a CNN–BiLSTM encoder that processes time–frequency representations into contextual acoustic embeddings. On top of this encoder, we implement:

- **CTC baseline:** Frame-synchronous pitch prediction with vocabulary {MIDI pitches} $\cup$ {`BLANK`}, fixed-length output of size T, monotonic alignment, and collapse-based decoding.
- **Seq2Seq model:** Autoregressive event-token generation using vocabulary {`TIME_SHIFT`, `NOTE_ON(p)`, `NOTE_OFF(p)`, `EOS`} with additive attention over encoder states. Polyphony is represented explicitly at the token level: multiple `NOTE_ON` events may be emitted consecutively at the same decoder time step before a `TIME_SHIFT` token advances the internal clock. Note durations are modeled via matching `NOTE_OFF` tokens that deactivate previously activated pitches. This event-based grammar therefore encodes simultaneous notes and variable-length durations directly within the symbolic sequence, in contrast to frame-synchronous pitch classification.

Both models are evaluated using:

- **Token-level metrics:** token error rate (TER) and token-F1.
- **Event-level metrics:** onset precision, recall, and F1 under timing tolerance.

Additionally, we include diagnostic analyses such as confusion matrices, attention peak trajectories, and entropy measures to interpret alignment learning and decoding behavior.

1.5 Contributions

The main contributions of this seminar project are:

1. A unified event-token and frame-synchronous transcription pipeline for MusicNet.
2. Implementation of both CTC and additive-attention Seq2Seq models with the same encoder Model.
3. Comparative evaluation under limited-data conditions.
4. Detailed diagnostic analysis of decoding behavior, alignment patterns, and pitch confusion.

1.6 Structure of the Report

The remainder of this report is organized as follows. Section 2 describes the dataset and subset construction. Subsequent sections detail preprocessing and model design for CTC and Seq2Seq. Experimental results are then presented and analyzed, followed by conclusions and discussion of limitations and future work.

2 Data

All experiments use the MusicNet dataset, introduced by Thickstun et al. (2017b) in *Learning Features of Music from Scratch* as a freely licensed benchmark for supervised AMT (Automatic Music Transcription). MusicNet comprises 330 classical music recordings spanning ten composers (Beethoven, Bach, Brahms, Cambini, Haydn, Liszt, Mozart, Schubert, and others) and eleven instrument families. Each recording is accompanied by over 1.29 million aligned note annotations, each specifying start time, end time, instrument ID, MIDI pitch, and metrical position. These labels are precisely time-aligned to human performances through an automated synchronization protocol that warps musical scores onto audio recordings, which are then human-verified for accuracy. Each record in MusicNet is a comma-separated annotation file in which every row encodes a single note event as a tuple (`start_time`, `end_time`, `instrument`, `MIDI_note`, `measure`, `beat`, `note_value`), where `start_time` and `end_time` are given in samples at 44,100 Hz.

2.1 Subset construction and split strategy

Due to computational and storage constraints, this study uses a 50-track subset of original dataset. The selection process for the training pool was motivated by label density; specifically, we selected 40 recordings from the training set with the highest frequency of labeled note events to ensure the models received maximum supervision per audio frame. The 40 training tracks were divided into a training set and a validation set, while the test set remains unchanged. A critical decision in our methodology was the implementation of a work-disjoint validation split. Standard random splits often lead to data leakage, where segments of the same musical piece (e.g., different movements or identical melodic themes) appear in both training and validation sets. To prevent this, we grouped recordings by a "work key" consisting of the composer and the composition title. We then held out 8 recordings (approximately 20% of the training pool) for validation, ensuring that no musical work in the validation set had any presence in the training

Table 1: Key descriptive statistics of the 50-track subset.

Metric	**Value**
Tracks (total)	50
Split (train / val / test)	32 / 8 / 10
Ensemble dominance	Solo Piano: 38/50 (76%)
Track duration (min–max)	52.91–225.06 s
Note events per track (median)	886.5
Pitch coverage (train / test)	27–96 / 28–95 (MIDI)
Polyphony p95 (median, IQR)	4 active notes
Polyphony max (min–max)	1–15 active notes

data. The final allocation resulted in 32 tracks for training, 8 for validation, and the 10 standard test tracks.

2.2 Descriptive statistics and implications

Table 1 summarizes the key descriptive statistics of the MusicNet subset used in this study. Track durations are relatively uniform, so performance differences are unlikely to be explained by large variation in excerpt length. Instead, they are more driven by differences in annotation density and musical texture. The subset provides a consistently high level of supervision. Tracks contain many labeled note events and these events occur frequently over time. This density is advantageous for optimization because it increases the amount of target information per frame, but it also intensifies the CTC alignment problem, since long target sequences must be mapped onto a finite number of acoustic frames under temporal uncertainty. In addition, the subset is strongly imbalanced toward solo piano (76% of tracks). From a music-theoretic perspective, solo piano is among the most demanding settings for automatic music transcription. Classical piano music often has many notes played at the same time (sometimes up to 10), and it can cover almost the full keyboard range. Because piano appears so often in the dataset, the model is exposed much more to piano sounds than to the different sound patterns of wind or string instruments. Pitch statistics show that most annotated notes fall in the middle register rather than covering the full 88-key range (MIDI 21–108). In our subset, the labeled pitch range is narrower (train: 27–96; test: 28–95). For a pitch-only CTC formulation with an 88-key vocabulary, this implies token imbalance. Mid-range pitches contribute most of the training signal, while very low and very high keys are rare or absent, which can reduce recall for these pitches and bias the model's emission probabilities. Since the extreme registers are underrepresented, results should be interpreted as most

reliable for the central register and less conclusive for the lowest and highest keys.

3 Methodology and Model Design

3.1 PreProcessing

3.1.1 CTC: CQT and HCQT

A central design choice in Automatic Music Transcription (AMT) is the acoustic input representation. For the CTC model, two time–frequency features were evaluated: the Constant-Q Transform (CQT) and the Harmonic CQT (HCQT). AMT is fundamentally pitch-centric: pitch and musical structure are organized on an approximately logarithmic frequency scale, where an octave corresponds to a doubling of frequency. In addition, many musical instruments produce harmonic spectra, meaning that energy is not only present at a fundamental frequency but also at integer multiples (overtones). For these reasons, log-frequency filterbanks provide a useful inductive bias for transcription models. They align the representation with musical structure and enable parameter sharing along the frequency axis, an idea that has been exploited successfully in supervised transcription pipelines (Thickstun et al., 2017). More broadly, empirical AMT studies show that the choice of time–frequency representation can materially affect transcription accuracy even when the network architecture is held fixed (Cheuk et al., 2020).

Segmentation and time resolution. The waveform $x[n]$ is resampled to $s = 44{,}100$ Hz (mono), then segmented into excerpts of $L_{\text{seg}} = 10$ s with a segment hop of 5 s (50% overlap). Within each excerpt, the time–frequency transform uses a hop size of $H = 512$ samples, yielding a frame rate of $s/H \approx 86.1$ frames/s (about $T \approx 862$ frames per 10 s segment in our setting). This temporal resolution is intentionally high, since the CTC formulation relies on per-frame posterior estimates and marginalizes over alignments via a forward–backward recursion.

Constant-Q Transform (CQT). The Constant-Q Transform (CQT) represents audio as a time–frequency map whose frequency bins are spaced on a logarithmic scale. This is well matched to music because musical pitch perception is approximately logarithmic: moving up by one octave corresponds to doubling the frequency. In the CQT, the center frequencies therefore increase geometrically. Let b denote the number of bins per octave and $f_{\min}$ the lowest center

frequency. The center frequency of bin $k \in \{0, \ldots, F-1\}$ is

$$f_k = f_{\min} \, 2^{k/b}. \tag{1}$$

A common definition of the (complex-valued) CQT coefficient at frequency bin k and time frame τ is given by a windowed inner product between the signal and a complex sinusoid:

$$C[k, \tau] = \sum_{n=0}^{N_k - 1} x[n + \tau H] \; w_k[n] \; e^{-2\pi i \, n \, f_k / s}, \tag{2}$$

where $x[\cdot]$ is the waveform, s is the sampling rate, H is the hop size between frames, $w_k[n]$ is an analysis window, and N_k is a bin-dependent window length. A key property of constant-Q designs is that N_k increases as f_k decreases: low frequencies are analyzed with longer windows (better frequency resolution), while high frequencies use shorter windows (better temporal resolution). This trade-off aligns well with musical acoustics, where low notes require finer frequency discrimination and high notes benefit from sharper timing.

In this study, the CQT is computed with $b = 36$ bins per octave and $F = 252$ total bins, using $f_{\min} = 32.703$ Hz (C1), which provides roughly seven octaves of coverage. The magnitude CQT is used, followed by logarithmic compression and per-segment standardization:

$$X_{\text{cqt}}[k, \tau] = \log \left(|C[k, \tau]| + \varepsilon \right), \tag{3}$$

$$\widetilde{X}_{\text{cqt}} = \frac{X_{\text{cqt}} - \mu}{\sigma + \delta}, \tag{4}$$

where ε avoids numerical issues when $|C|$ is near zero, and μ and σ are the mean and standard deviation computed over all time–frequency entries of the segment. This zero-mean, unit-variance normalization reduces sensitivity to overall loudness and dynamic range and is commonly used to stabilize learning in deep spectral pipelines (Taenzer et al., 2019).

Harmonic CQT (HCQT). While a single CQT provides a musically meaningful frequency axis, many instruments produce harmonic spectra: energy is distributed not only at the fundamental frequency but also at multiples of it (overtones). In polyphonic music, these harmonic patterns overlap across notes, making pitch inference more challenging. The Harmonic CQT (HCQT) makes harmonic relations easier to access by stacking multiple CQTs computed on frequency-scaled grids, so that evidence related by harmonic factors appears aligned across channels (Weiss et al., 2021). Let $\mathcal{H} = \{h_1, \ldots, h_C\}$

be a set of harmonic multipliers. For each $h_c \in \mathcal{H}$, a CQT is computed with a scaled minimum frequency $f_{\min}^{(c)} = h_c f_{\min}$ (keeping the remaining parameters fixed), and the resulting representation is

$$X_{\text{hcqt}}[c, k, \tau] = \log\left(\left|C^{(c)}[k, \tau]\right| + \varepsilon\right), \tag{5}$$

where $C^{(c)}$ denotes the CQT computed using $f_{\min}^{(c)}$. This yields a tensor $X_{\text{hcqt}} \in \mathbb{R}^{C \times F \times T}$. In our configuration, $\mathcal{H} = (0.5, 1, 2, 3, 4)$, producing $C = 5$ channels. After stacking, a single normalization is applied over all entries of the segment:

$$\widetilde{X}_{\text{hcqt}} = \frac{X_{\text{hcqt}} - \mu_{\text{all}}}{\sigma_{\text{all}} + \delta}. \tag{6}$$

Rationale for CTC-based transcription. The CTC baseline produces frame-synchronous distributions over pitch tokens (plus blank) and learns alignments implicitly by marginalizing over possible frame-to-token assignments during training. In this setting, HCQT can be viewed as an encoder-side inductive bias: by exposing harmonic structure explicitly across channels, early convolutional filters can pool correlated energy at harmonically related locations instead of learning overtone relationships solely within a single time–frequency map. This is particularly relevant in dense polyphonic mixtures, where multiple simultaneous notes share partials and therefore generate overlapping spectral energy.

3.1.2 S2S: Log-Mel Spectrogram Features

Why we use log-mel. For the Seq2Seq Model, we transform each audio excerpt into a *log-mel spectrogram* $X \in \mathbb{R}^{T \times F}$, which provides a compact time–frequency representation that is well-suited for convolutional feature extraction. The mel scale roughly matches human frequency resolution and compresses high-frequency detail, while the logarithm stabilizes dynamic range so that both quiet and loud note components remain learnable.

STFT and mel filterbank. Let $x[n]$ be the discrete-time audio signal sampled at f_s Hz. We compute a short-time Fourier transform (STFT) using a window of length N_{fft} and hop length H (in samples). For frame index t and frequency bin k,

$$X_{\text{stft}}(t, k) = \sum_{n=0}^{N_{\text{fft}}-1} x[n + tH]\; w[n]\; e^{-j2\pi kn/N_{\text{fft}}},$$

where $w[n]$ is the analysis window. We use the power spectrum $|X_{\text{stft}}(t,k)|^2$ and project it onto F mel bands via a mel filterbank matrix $M \in \mathbb{R}^{F\times K}$:

$$S_{\text{mel}}(t,f) = \sum_{k=1}^{K} M_{f,k}\,|X_{\text{stft}}(t,k)|^2, \quad f = 1,\dots,F.$$

Finally, we apply a logarithmic compression (with small $\varepsilon > 0$ for numerical stability):

$$X(t,f) = \log\big(S_{\text{mel}}(t,f) + \varepsilon\big).$$

This yields the input feature matrix $X \in \mathbb{R}^{T\times F}$.

Temporal resolution and alignment to labels. Each row of X corresponds to one time frame of duration

$$\Delta t = \frac{H}{f_s} \text{ seconds.}$$

All symbolic labels (note onsets/offsets) are converted into *frame indices* on this grid. Concretely, if an event occurs at time τ seconds, its frame index is

$$t(\tau) = \left\lfloor \frac{\tau}{\Delta t} \right\rfloor.$$

In our event-token representation, time is advanced by emitting `TIME_SHIFT` tokens that move the current frame index forward; `NOTE_ON(p)` and `NOTE_OFF(p)` then mark changes of pitch activity at the current frame. Therefore, the hop length H directly controls the granularity of possible onset/offset positions in the target token sequence.

How log-mel is used by the encoder. We treat X as a 2D "image" with axes (time $\times$ mel-frequency) and feed it into the CNN–BiLSTM encoder. The CNN layers learn local time–frequency patterns (e.g., harmonic stacks, spectral envelopes, and onset-like transients), producing a sequence of latent feature maps over time. The BiLSTM then aggregates temporal context in both forward and backward directions, outputting an encoder state sequence

$$H = (h_1,\dots,h_{T'}) \in \mathbb{R}^{T'\times d},$$

where T' may be smaller than T if the CNN uses pooling/striding, and d is the encoder feature dimension. These encoder states h_t form the conditioning signal for: (i) the CTC classification head (frame-wise logits), and (ii) the Seq2Seq decoder attention mechanism, which computes a context vector as a weighted sum over $\{h_t\}_{t=1}^{T'}$ at each decoding step.

Practical note (normalization). For stable training, we apply the same feature normalization procedure (e.g., per-example or dataset-level mean/variance normalization) consistently across train/validation/test. This ensures that the CNN sees log-mel inputs on a comparable scale and reduces sensitivity to recording-level gain differences.

3.2 Methodology

3.2.1 Convolutional Neural Network

Mathematical foundations. A Convolutional Neural Network (CNN) is a feed-forward architecture that exploits locality via shared kernels applied through discrete 2-D cross-correlation. CNNs were popularised for visual pattern recognition by LeCun et al. (1998) and subsequently became a standard front-end for structured signals, including spectrogram-like audio representations. Following the convention of Goodfellow et al. (2016), we denote the 2-D cross-correlation (often referred to as "convolution" in deep learning libraries) for an input map $X \in \mathbb{R}^{T\times F}$ and kernel $W \in \mathbb{R}^{k_T\times k_F}$ by

$$(X \star W)[t,f] \;=\; \sum_{i=0}^{k_T-1}\sum_{j=0}^{k_F-1} X[t+i, f+j]\, W[i,j]. \tag{7}$$

With C_{in} input channels and C_{out} kernels, one convolutional layer produces an output tensor $Z \in \mathbb{R}^{C_{\text{out}}\times T\times F}$, with parameter count $C_{\text{in}}C_{\text{out}}k_Tk_F$ (plus biases), independent of the spatial dimensions. This weight sharing is the key computational advantage over dense layers for time–frequency "images".

Non-linearity and normalisation. After each convolution we apply a Rectified Linear Unit (ReLU) $\sigma(z) = \max(0, z)$, which avoids saturating gradients and supports deep optimisation (Nair & Hinton, 2010). We additionally apply Batch Normalisation (BN) to stabilise training by normalising intermediate activations using mini-batch statistics (Ioffe & Szegedy, 2015). For an activation channel z_c in a mini-batch, BN computes

$$\tilde{z}_c = \frac{z_c - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}}, \qquad \hat{z}_c = \gamma_c \tilde{z}_c + \beta_c, \tag{8}$$

where μ_c and σ_c^2 are batch mean and variance, and γ_c, β_c are learnable affine parameters.

Pooling and the asymmetric design choice. To reduce redundancy and improve invariance, we use max-pooling. For pooling window $(p_T \times p_F)$, max-pooling computes

$$\mathrm{Pool}[t, f] = \max_{0 \le i < p_T,\, 0 \le j < p_F} Z[t \cdot p_T + i,\ f \cdot p_F + j]. \tag{9}$$

The key architectural decision in our encoder is *asymmetric* pooling with (1×2), i.e., pooling only along frequency and leaving time intact. This is not an aesthetic choice; it is structurally motivated by alignment-sensitive decoding and by the different semantics of time vs. frequency in music: time encodes event timing (onsets and offsets), whereas adjacent frequency bins often carry redundant harmonic energy. Compressing frequency is therefore acoustically safer than compressing time.

CNN front-end for the CTC model (CQT/HCQT input). We first document the CNN as used in the CTC baseline, since it directly consumes the CQT/HCQT features defined in the preprocessing subsection. The input to the CNN is a time–frequency tensor X with shape (T, F) for CQT or (T, C, F) for HCQT. For convolution, we use channel-first layout: CQT becomes $(1, T, F)$ and HCQT becomes (C, T, F), where $C = |\mathcal{H}|$. This distinction is fundamental: under HCQT, the channel dimension encodes harmonic factors, so the first convolution can learn cross-harmonic combinations at fixed time–frequency locations.

The CNN consists of three stages, each of the form

$$\mathrm{Conv2d}(3 \times 3, \mathrm{pad} = 1) \to \mathrm{BN} \to \mathrm{ReLU} \to \mathrm{MaxPool}(1 \times 2),$$

with channel widths $[32, 64, 128]$. After three (1×2) pools, the frequency dimension is reduced by a factor $2^3 = 8$, while the number of time frames remains T. Let F_0 denote the original number of frequency bins; the output has shape $(128, T, F_0/8)$. We then permute and flatten to obtain a time-major sequence of embeddings:

$$H_{\mathrm{cnn}} \in \mathbb{R}^{T \times d_{\mathrm{cnn}}}, \qquad d_{\mathrm{cnn}} = 128 \cdot (F_0/8). \tag{10}$$

This sequence is then passed to the BiLSTM encoder (described in the next subsection). Importantly, preserving the time dimension is particularly critical in CTC settings because the encoder output length determines the number of alignment positions available to the forward–backward recursion; excessive time downsampling can therefore make long label sequences difficult or impossible to align in practice.

CNN front-end for the Seq2Seq model (log-mel input) and architectural differences. The Seq2Seq model uses the same CNN template (three Conv–BN–ReLU–Pool stages with frequency-only pooling), but the input representation differs: instead of CQT/HCQT, Seq2Seq consumes a log-mel spectrogram (see the log-mel subsection). Consequently, the input is always single-channel, i.e., $(T, M) \mapsto (1, T, M)$, where M is the number of mel bins. This changes the *semantics* of the first-layer channels: in the CTC+HCQT case, channels correspond to harmonic factors; in the Seq2Seq case, all harmonic structure must be inferred within one channel across the frequency axis. The downstream implication is that the CTC encoder can exploit explicit cross-harmonic evidence early in the network, whereas Seq2Seq relies more heavily on deeper layers and the recurrent context to disentangle harmonics from timbre.

Despite this representational difference, the pooling constraint remains aligned with both paradigms. For CTC it is a mathematical and practical alignment constraint (time resolution cannot be overly compressed), while for Seq2Seq it improves attention localisation by maintaining a fine-grained memory over frames. Therefore, the same (1×2) frequency pooling is a Pareto-consistent choice: it satisfies the stricter CTC requirement while preserving maximal temporal detail for attention-based decoding.

3.2.2 BiLSTM

After the CNN front-end has transformed log-mel frames into higher-level local feature representations, our encoder uses a **bidirectional Long Short-Term Memory (BiLSTM)** layer to model temporal structure across frames. Recurrent modeling is important in polyphonic transcription because note events are rarely identifiable from a single frame: onsets may be smeared by the instrument's attack, harmonic energy overlaps across simultaneous notes, and offsets can be masked by reverberation or sustain. A BiLSTM mitigates these ambiguities by conditioning each time step not only on past context but also on future context (Schuster and Paliwal, 1997; Graves, 2012).

Sequence representation entering the BiLSTM. Let $\mathbf{X} \in \mathbb{R}^{T \times M}$ denote the log-mel spectrogram sequence with T time frames and M mel bins. The CNN produces an embedding sequence

$$\mathbf{z}_{1:T} = (\mathbf{z}_1, \ldots, \mathbf{z}_T), \qquad \mathbf{z}_t \in \mathbb{R}^{d_z},$$

where $\mathbf{z}_t$ summarizes a local time–frequency neighborhood around frame t (local invariances such as small spectral shifts or noise are handled at this stage). Convolutional front-ends are a common choice for learning

representations from spectrogram-like inputs, while recurrent layers capture longer-range temporal dependencies (Goodfellow et al., 2016).

LSTM dynamics and gating (why LSTMs instead of vanilla RNNs). An LSTM augments a recurrent hidden state with a memory cell that is updated through gates, enabling stable learning of dependencies across many time steps and mitigating vanishing gradients (Hochreiter and Schmidhuber, 1997). For each time step t (we write the equations for one direction), given input $\mathbf{z}_t$, previous hidden state $\mathbf{h}_{t-1}$, and previous cell state $\mathbf{c}_{t-1}$:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{z}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{11}$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{z}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{12}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{z}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \tag{13}$$
$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{z}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \tag{14}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{15}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{16}$$

where $\sigma(\cdot)$ is the logistic sigmoid, $\odot$ denotes element-wise multiplication, $\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t$ are the input/forget/output gates, and $\mathbf{c}_t$ is the memory cell state.

Bidirectional recurrence (context from both sides). We run two LSTMs: a forward LSTM (left-to-right) and a backward LSTM (right-to-left).

$$\overrightarrow{\mathbf{h}}_t = \mathrm{LSTM}_f(\mathbf{z}_t, \overrightarrow{\mathbf{h}}_{t-1}), \tag{17}$$
$$\overleftarrow{\mathbf{h}}_t = \mathrm{LSTM}_b(\mathbf{z}_t, \overleftarrow{\mathbf{h}}_{t+1}), \tag{18}$$

and concatenate their hidden states to obtain the encoder output at frame t:

$$\mathbf{h}_t = [\, \overrightarrow{\mathbf{h}}_t ; \overleftarrow{\mathbf{h}}_t \,] \in \mathbb{R}^{2d_h}.$$

The full encoded sequence is $\mathbf{H} = (\mathbf{h}_1, \ldots, \mathbf{h}_T) \in \mathbb{R}^{T \times 2d_h}$. Intuitively, $\mathbf{h}_t$ can represent evidence that spans several frames before and after t, which helps localize the *timing* of note events while still capturing global context such as tempo and sustained harmonics (Schuster and Paliwal, 1997; Graves, 2012).

How the BiLSTM output is used by our decoder. Our Seq2Seq decoder does not directly predict frame-wise note states; instead it predicts a sequence of *event tokens* (time-shifts, note-on, note-off). Therefore, the encoder must provide a rich sequence representation that can be *queried*

during decoding. We use additive attention to compute, for each decoder step, a context vector as a weighted sum over $\mathbf{H}$ (Bahdanau et al., 2015). In our implementation, the BiLSTM output dimensionality (the concatenated forward/backward hidden states) is the *attention source* that the decoder aligns to, making $\mathbf{H}$ the central interface between acoustic evidence and symbolic event generation.

Why this choice is appropriate for our setting. Given the limited amount of training material in our subset (tens of tracks), the BiLSTM provides a strong inductive bias for sequential structure without requiring extremely large data to learn long-range dependencies from scratch. It improves robustness to local ambiguity by incorporating future context, and it yields an encoder sequence that is naturally compatible with attention-based decoding (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997; Bahdanau et al., 2015).

4 Connectionist Temporal Classification (CTC)

4.1 Theoretical Foundation

Transcription is formulated as an *alignment problem* between two sequences of different length. The encoder produces a frame-synchronous sequence of outputs (one output per time frame), while the target is a shorter sequence of pitch tokens without explicit frame-level timing. Connectionist Temporal Classification (CTC) addresses this setting by defining a likelihood that *sums over all valid monotonic alignments* between frames and target tokens. This makes it possible to train end-to-end without providing explicit onset/offset labels at the frame level (Graves et al., 2006; Zalkow and Müller, 2021; Weiss et al., 2021).

In the present model, the time axis is preserved throughout the encoder (pooling is applied only along the frequency dimension). As a result, the number of encoder steps T equals the number of input frames in a segment. With a hop length of 512 samples at 44.1 kHz, a 10 s segment yields approximately $T \approx 862$ frames.

4.1.1 CTC vocabulary, pitch-only targets, and the blank symbol

The pitch vocabulary is restricted to the standard 88-key piano range, i.e., MIDI notes $m \in \{21, \ldots, 108\}$. Each MIDI pitch is mapped to a pitch class index

$$\phi(m) = m - 21 \in \{0, \ldots, 87\}. \tag{19}$$

CTC augments the pitch alphabet with a dedicated *blank* symbol $\varnothing$, which represents "no output" at a frame (Graves et al., 2006). In addition, special symbols such as padding and sequence boundary markers are reserved for batching and potential sequence-to-sequence experiments. However, the effective CTC label set consists of the 88 pitch symbols plus blank:

$$\mathcal{V}_{\text{CTC}} = \{\text{P}_0, \ldots, \text{P}_{87}\} \cup \{\varnothing\}. \tag{20}$$

Targets are constructed as *pitch-only onset sequences* within each segment. For a segment interval $[s_{\text{seg}}, e_{\text{seg}})$, all note events whose onset lies in the interval are selected, sorted by onset time (and then by pitch), mapped via $\phi(\cdot)$, and finally compressed by removing *immediate repeats* (consecutive duplicate pitch IDs). Segments with no onset events are excluded to avoid degenerate all-blank training examples.

This tokenization linearizes polyphonic chords into an ordered list of pitches. Consequently, the model learns to predict an *ordered pitch stream* rather than a multi-label chord set per frame.

4.1.2 Framewise predictions and CTC decoding

Given an input segment X, the encoder produces hidden states $h_t \in \mathbb{R}^d$ for $t = 1, \ldots, T$. A linear classifier followed by a softmax yields framewise probabilities over the CTC vocabulary:

$$\ell_t = W h_t + b, \qquad p(v \mid h_t) = \text{softmax}(\ell_t)_v, \quad v \in \mathcal{V}_{\text{CTC}}. \tag{21}$$

A CTC alignment is a length-T path $\pi = (\pi_1, \ldots, \pi_T)$ with $\pi_t \in \mathcal{V}_{\text{CTC}}$. The collapsing map $\mathcal{B}$ transforms an alignment into an output label sequence by (i) merging consecutive repeats and (ii) removing blanks:

$$\mathcal{B}(\pi) = \text{remove\_blanks}\big(\text{merge\_repeats}(\pi)\big). \tag{22}$$

For example, $\mathcal{B}([\varnothing, \varnothing, A, A, \varnothing, B, \varnothing]) = [A, B]$. Our greedy decoder implements this collapse in a single left-to-right pass: blanks are dropped, and repeats are only merged when they are consecutive *without* an intervening blank (blanks reset the repeat state).

For a target sequence $y = (y_1, \ldots, y_M)$, the set of valid alignments is

$$\Phi(y) = \{\pi \in \mathcal{V}_{\text{CTC}}^T : \mathcal{B}(\pi) = y\}. \tag{23}$$

CTC defines the conditional probability of y by marginalising over all valid alignments:

$$p(y \mid X) = \sum_{\pi \in \Phi(y)} \prod_{t=1}^{T} p(\pi_t \mid h_t). \tag{24}$$

This sum is exponentially large in T, but can be computed efficiently using a forward–backward dynamic program (Graves et al., 2006; Zalkow and Müller, 2021).

4.1.3 Training objective

Given a dataset $\mathcal{D} = \{(X^{(n)}, y^{(n)})\}_{n=1}^{N}$, training minimizes the negative log-likelihood:

$$\mathcal{L}_{\mathrm{CTC}}(\theta) = -\sum_{n=1}^{N} \log p\big(y^{(n)} \mid X^{(n)}; \theta\big)\,. \tag{25}$$

In practice, the CTC loss is computed with a standard library implementation (e.g., `torch.nn.CTCLoss`), which performs the same forward and backward marginalization internally.

4.1.4 Decoding and blank-penalty calibration

At inference time, best-path (greedy) decoding is used: the most probable symbol is selected at each frame and then collapsed via $\mathcal{B}(\cdot)$. A common practical issue is *blank dominance*, where blank becomes the most frequent framewise prediction, especially early in training. To reduce overly blank-heavy greedy outputs, a simple *blank penalty* is applied at decoding time by subtracting a constant α from the blank score before the argmax:

$$\log \tilde{p}(v \mid h_t) = \begin{cases} \log p(v \mid h_t) - \alpha, & v = \varnothing, \\ \log p(v \mid h_t), & \text{otherwise.} \end{cases} \tag{26}$$

The penalty is annealed during early training and evaluated on a small grid of candidate values; validation performance is used to select the decoding penalty reported for test results.

4.2 Evaluation Metrics

4.2.1 Token F1 Score

The primary metric is a bag-of-tokens Token F1 score, computed as the harmonic mean of token-level Precision (P) and Recall (R) accumulated over the entire test set:

$$P = \frac{TP}{TP + FP}, \qquad R = \frac{TP}{TP + FN},$$

$$F1 = \frac{2PR}{P + R}.$$

where TP, FP, and FN denote true positives, false positives, and false negatives, respectively, accumulated globally across all test segments before forming the ratios (micro-averaging). This metric is well matched to the unordered pitch-bag formulation because it penalizes both over-prediction (high $FP \rightarrow$ low Precision) and under-prediction (high $FN \rightarrow$ low Recall) symmetrically.

A key limitation is that this F1 score ignores ordering information. A model that correctly identifies all pitches active in a 10-second segment but assigns them in the wrong temporal order will still score $F1 = 1.0$. This is an explicit design choice arising from the weakly supervised CTC target, and it means Token F1 can overestimate transcription quality relative to onset-level note evaluation. Within the present setup, it should therefore be interpreted primarily as a measure of pitch vocabulary coverage rather than temporal accuracy.

4.2.2 Token Error Rate (TER)

To complement the order-invariant F1 score, we report a Token Error Rate derived from edit-distance measures used in sequence recognition. Given a decoded token sequence $\hat{Y}$ and a reference sequence Y, the token error rate is defined as

$$TER = \frac{S + D + I}{M},$$

where S, D, and I are the numbers of substitutions, deletions, and insertions required to transform $\hat{Y}$ into Y, and $M = |Y|$ is the reference length. TER is computed via Levenshtein distance (Levenshtein, 1966), as commonly used in sequence recognition (Jurafsky and Martin, 2009). Unlike token F1, TER is order-sensitive and explicitly penalizes length mismatches: under-generation increases deletion counts, while over-generation increases insertion counts. By construction, $TER = 0$ indicates an exact match.

4.2.3 Length Ratio

Many failure modes in CTC manifest as systematic under- or over-generation, so we include a simple length diagnostic: the ratio of decoded to reference token counts,

$$\frac{|\hat{Y}|}{|Y|}.$$

Values below 1 indicate under-generation, whereas values above 1 indicate over-generation. In the present setting, very small ratios are a strong indicator of blank dominance, *a known failure mode in CTC-style training* (Graves

et al., 2006). For stable runs, the length ratio typically falls moderately below 1, reflecting that some target pitches are missed in a dense polyphonic context.

4.2.4 Per-Pitch Precision, Recall and F1

To diagnose pitch-specific performance, we compute pitch-wise precision, recall, and F1 scores for each MIDI note. For every pitch, true positives, false positives, and false negatives are accumulated across the full test set, and the corresponding per-pitch metrics are computed using the same definitions as above. This analysis isolates whether errors concentrate in rarely observed extremes versus the frequently occurring mid-register pitches.

4.2.5 Blank Diagnostics

CTC models can fail by assigning most probability mass to the blank symbol (often referred to as blank dominance), so we report additional diagnostics alongside the main evaluation metrics. For each epoch, we record (i) the proportion of time frames whose most likely prediction is blank, and (ii) the average predicted probability assigned to blank across frames. Together, these measures indicate whether the model relies on blank as a default outcome rather than emitting informative pitch tokens.

Additionally, we track output diversity and confidence. Diversity is measured as the number of distinct non-blank pitch tokens produced under framewise decoding, indicating whether the model collapses to a small subset of pitches. Confidence is summarized by the normalized entropy of the framewise class probabilities: low entropy reflects overly peaked predictions (often seen in collapse), whereas higher entropy suggests a more distributed and stable output distribution.

Finally, we monitor gradient norms in the major network components (encoder and classifier). These values help distinguish blank collapse from other training issues such as vanishing gradients or unstable updates. Overall, these diagnostics provide insight into training dynamics and support a clear separation between blank-dominated and healthy regimes.

4.3 Experimental setup and results

All CTC experiments share the following training setup: the Adam optimiser with an initial learning rate 1×10^{-3}, batch size 32 (variable-length sequences are padded to the maximum T within each batch), gradient clipping at a global norm of 5.0, weight decay 1×10^{-4}, and early stopping on validation

Table 2: Experiment I Summary Results.

Exp	Feature	Aug	Best Epoch	Best Val F1	Test F1 / TER
1	HCQT	Off	67	0.8227	0.7227 / 0.6141
2	HCQT	On	76	0.7569	0.5857 / 0.6580
3	CQT	On	77	0.6565	0.4783 / 0.7365
4	CQT	Off	72	0.8161	0.7055 / 0.6648

Token F1 with a patience of 20 epochs (maximum 100 epochs). The CTC head consists of a single linear projection from the BiLSTM output dimension to $|V_{\text{CTC}}| = 89$ logits, followed by a softmax layer. Greedy decoding is used throughout. Experiment I evaluates four configurations (2 feature representations $\times$ 2 augmentation settings) under a fixed random seed (42). Experiment II repeats the best-performing configuration (HCQT, no augmentation) across five independent random seeds to assess variance.

Augmentation. In augmented runs, data augmentation comprises: (i) frequency-axis bin rolling (to simulate pitch shifts) and (ii) time and frequency masking following the SpecAugment strategy (Park et al., 2019). As discussed in Section 4.5, augmentation proved counterproductive when not paired with a consistent label transformation.

4.3.1 Experiment I

Table 2 summarizes the results of Experiment I. Across the four runs, the strongest configuration is HCQT without augmentation (Token F1 = 0.7227, TER = 0.6141). By the model's best epoch (67), training diagnostics are consistent with stable learning. Argmax decoding produces 50 distinct pitch tokens (covering a substantial portion of the 88-key vocabulary) and normalized entropy drops from 0.240 at initialization to 0.018, indicating that the model has become highly confident in its per-frame predictions. The loss decreases to approximately 0.09, and the gradient norm distribution shows the CNN contributing the largest fraction (0.0363) while the LSTM and classifier contribute smaller, stable fractions. This indicates that the CNN is the primary learning site in this architecture configuration.

The blank rate of 73.8% needs to be interpreted carefully in the context of how CTC aligns tokens to frames. On average, each segment contains about $M \approx 41$ pitch tokens spread over $T \approx 862$ time frames. If each target token were matched to roughly a single frame and the remaining frames were

blank, the expected blank proportion would be approximately

$$\frac{T-M}{T} \approx \frac{862-41}{862} = 95.2\%.$$

The fact that we observe a much lower blank fraction (73.8%) indicates that the model does not place each token on a single frame. Instead, it tends to hold the same token across multiple consecutive frames. This behavior is consistent with polyphonic music, where pitches often persist over time (sustained notes) and where repeated notes may appear as longer runs of the same pitch within the alignment.

The second-best configuration is CQT without augmentation (Token F1 = 0.7055, TER = 0.6648). Comparing the non-augmented runs shows a consistent but modest advantage for HCQT over CQT. A plausible interpretation is that HCQT makes harmonic structure more explicit by providing several harmonically related channels, so the CNN can detect cross-channel harmonic patterns directly rather than inferring them implicitly from a single 2D map. This advantage is most visible in the edit-distance metric (lower TER), which is sensitive to ordering and length mismatches.

Both augmented runs perform worse, with CQT + augmentation dropping most (Token F1 = 0.4783, TER = 0.7365). A likely explanation is that the augmentation pipeline includes frequency-axis shifts (via bin rolling) and time/frequency masking. If the input is shifted in frequency but the pitch-token targets are not shifted accordingly, the transformation is no longer strictly label-preserving, which can weaken the supervision signal. This is important because augmentation is most beneficial when it preserves the target labels (or transforms them consistently) (Thickstun et al., 2017a). The effect is strongest for CQT because it is a single-channel representation, so masking or frequency displacement removes or relocates key pitch evidence directly. For future work, pitch-shift augmentation should be paired with a matching shift of the target tokens (with careful handling of out-of-range pitches), and masking strengths should be tuned more conservatively for dense polyphonic segments.

Figure 1 shows blank dominance for the best configuration HCQT without augmentation. It shows that training starts in a high-blank regime, which is expected for CTC models before alignment stabilizes. It is easier for the model to explain everything as blank before it learns reliable non-blank emissions. Over training, it transitions to a "healthy" regime after epoch 18 with substantially more emitted pitch diversity. Blank dominance drops

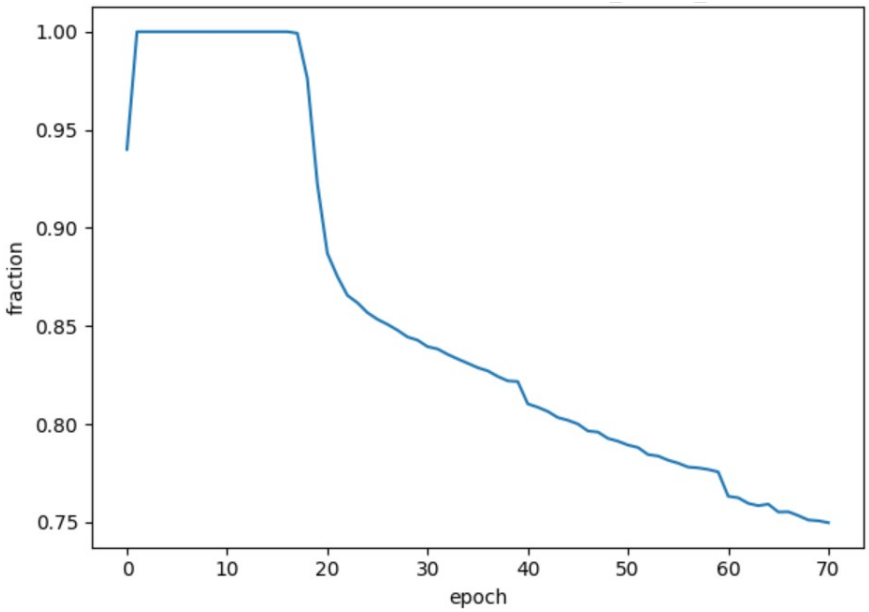

Figure 1: Blank dominance (framewise argmax = BLANK) over training epochs for the best configuration (HCQT, no augmentation).

sharply and then continues decreasing gradually, ending around 0.75. The model still produces many blank frames, which is normal in CTC because blanks occupy frames where no token is emitted.

4.3.2 Experiment II

Table 3 reports the results of Experiment II. After HCQT without augmentation was identified as the strongest setting in Experiment I, the same training and evaluation protocol was repeated across five independent random seeds to assess robustness. Across seeds, test Token F1 ranges from 0.684 to 0.727, and the best validation checkpoint occurs between epoch 60 (seed 44) and epoch 79 (seed 11). The mean test F1 is 0.7145 ± 0.0176, with a 95% confidence interval of [0.6927, 0.7363]. The corresponding mean TER is 0.6858 ± 0.0688, with a 95% confidence interval of [0.6004, 0.7712]. Overall, These results confirm that the point estimate of 0.7227 from Experiment I (seed 42) lies within but near the upper end of the cross-seed confidence band, suggesting seed 42 was modestly favourable but not an outlier.

Table 3: HCQT without augmentation across five random seeds.

Seed	Best Epoch	Best Val F1	Test F1	Test TER
11	79	0.8259	0.7214	0.6642
22	67	0.8361	0.7272	0.6302
33	74	0.8370	0.7150	0.6969
44	60	0.8169	0.6841	0.7997
55	75	0.8256	0.7248	0.6381
Mean	**71.0**	**0.8283**	**0.7145**	**0.6858**
Std	**7.00**	**0.0082**	**0.0176**	**0.0688**

5 Sequence to Sequence

5.1 Theoretical Foundations

Sequence-to-sequence (Seq2Seq) modeling casts transcription as a conditional generation problem: given an acoustic input representation X (here: log-mel frames), the model produces a discrete output sequence of symbolic events $y_{1:T}$ (here: time-shifts, note-on, note-off, end-of-sequence). In its canonical form, an *encoder* maps X to a sequence of latent acoustic states $\mathbf{h}_{1:U}$, while a *decoder* autoregressively predicts the next token from a learned distribution

$$p_\theta(y_t \mid y_{<t}, X),$$

and the full transcription probability factorizes as $p_\theta(y_{1:T} \mid X) = \prod_{t=1}^{T} p_\theta(y_t \mid y_{<t}, X)$ (Sutskever et al., 2014).

A central challenge in AMT is that musical structure couples *fine timing* with *polyphonic pitch content*. Event-based Seq2Seq addresses this by (i) representing time explicitly through dedicated `TIME_SHIFT` tokens and (ii) representing note activity through `NOTE_ON`/`NOTE_OFF` tokens, so that the decoder emits a musically meaningful symbolic stream rather than frame-wise classifications. To link the decoder to the relevant acoustic regions, our model uses *additive attention*, which computes a content-based alignment between the current decoder state and the encoder states, and forms a context vector that is fed back into the decoder at each step (Bahdanau et al., 2015). This "listen-and-attend" principle is widely used in sequence transduction tasks with unknown alignment (e.g., speech recognition), and directly motivates our attention-based decoder design (Chan et al., 2016).

From an optimization perspective, training typically uses *teacher forcing*, where the decoder conditions on the ground-truth prefix $y_{<t}$ to stabilize learning; however, this introduces *exposure bias* because at inference the model must condition on its own past predictions. We therefore incorporate

scheduled sampling as a simple mitigation, gradually replacing ground-truth decoder inputs with model predictions during training (Bengio et al., 2015). Finally, recent token-based AMT work demonstrates that Seq2Seq generation can be competitive even in low-resource regimes by leveraging unified event vocabularies and attention-based sequence modeling—most notably in Transformer-based multi-task transcription (Gardner, 2021). While our implementation uses an RNN decoder (rather than a Transformer), it follows the same fundamental Seq2Seq framing: *transcription as conditional token generation with learned alignment.*

Problem formulation. Let x denote an audio segment (waveform) and $X \in \mathbb{R}^{T \times M}$ its log-mel representation with T time frames and M mel bins. The target is an *event token* sequence $y = (y_1, \ldots, y_L)$, where each token $y_t \in \mathcal{V}$ belongs to a discrete vocabulary $\mathcal{V}$ that encodes musical events. We learn a conditional sequence model

$$p_\theta(y \mid X) = \prod_{t=1}^{L} p_\theta(y_t \mid y_{<t}, X),$$

where $y_{<t} = (y_1, \ldots, y_{t-1})$ and θ are model parameters.

Encoder–decoder with attention. We use an encoder that maps X to contextual acoustic states $H = (h_1, \ldots, h_{T'})$ with $h_i \in \mathbb{R}^{d_h}$. The decoder is an autoregressive RNN producing decoder states $s_t \in \mathbb{R}^{d_s}$ and output logits over $\mathcal{V}$.

Because the alignment between audio frames and symbolic events is unknown and non-monotonic at the event level, we apply additive attention (Bahdanau et al., 2015). At decoding step t, an attention score is computed for each encoder position i:

$$e_{t,i} = v^\top \tanh(W_h s_t + W_s h_i), \qquad \alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{T'} \exp(e_{t,j})},$$

and the context vector is

$$c_t = \sum_{i=1}^{T'} \alpha_{t,i} h_i.$$

The token distribution is obtained from

$$z_t = W_o[s_t; c_t] + b_o, \qquad p_\theta(y_t \mid y_{<t}, X) = \mathrm{softmax}(z_t).$$

Here W_h, W_s, v, W_o, b_o are learned parameters and $[s_t; c_t]$ denotes concatenation.

Training objective (teacher forcing). Training minimizes the negative log-likelihood under teacher forcing:

$$\mathcal{L}(\theta) \;=\; -\sum_{t=1}^{L} \log p_\theta(y_t \mid y_{<t}, X),$$

implemented as a token-level cross-entropy loss. We additionally apply *class weights* over the vocabulary to counter token-frequency imbalance.

Scheduled sampling. Teacher forcing can create exposure bias: at inference the model conditions on its own previous predictions, while during training it conditions on ground-truth history. To reduce this mismatch we use scheduled sampling (Bengio et al., 2015), where at each step the decoder input is chosen as

$$\tilde{y}_{t-1} = \begin{cases} y_{t-1}, & \text{with probability } 1 - p_{\text{ss}}, \\ \hat{y}_{t-1}, & \text{with probability } p_{\text{ss}}, \end{cases}$$

with $\hat{y}_{t-1} = \arg\max p_\theta(\cdot \mid \tilde{y}_{<t-1}, X)$. In our implementation p_{ss} increases linearly up to a capped maximum (Section 5.3).

Model architecture and design decisions. Our Seq2Seq system follows an encoder–attention–decoder design. The encoder transforms the acoustic input $X \in \mathbb{R}^{T \times M}$ (log-mel frames) into a shorter sequence of high-level representations $H = (h_1, \ldots, h_{T'})$, where $h_i \in \mathbb{R}^{d_h}$ summarizes local spectral patterns and longer-range temporal context around encoder frame i. This is achieved by a convolutional front-end followed by a bidirectional LSTM. The CNN layers act as learned time–frequency feature extractors and can reduce temporal resolution via striding/pooling, yielding $T' < T$ and improving efficiency while providing invariances to small local shifts. The subsequent BiLSTM produces contextual states by combining a forward recurrence (past context) and a backward recurrence (future context), so each $h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]$ incorporates information from both directions. This bidirectional context is particularly useful in music, where note evidence can span multiple frames and the acoustic realization of a note onset is influenced by what precedes and follows.

The decoder is an autoregressive LSTM that predicts an event-token sequence $y = (y_1, \ldots, y_L)$ conditioned on both its own history and the encoder states. At step t, the decoder maintains a hidden state s_t updated from the previous token embedding and an attention context vector:

$$s_t = \text{LSTM}\big(s_{t-1},\ [\text{Emb}(\tilde{y}_{t-1}); c_{t-1}]\big),$$

where $\tilde{y}_{t-1}$ denotes the previous input token (ground truth under teacher forcing or a sampled model prediction under scheduled sampling). The key challenge is that event tokens do not correspond to a fixed, monotonic alignment to acoustic frames: a single decoder step can represent a time shift or a note on/off event, and multiple note events may occur between time shifts. We therefore employ additive attention to produce a content-based, step-specific summary of the encoder output. Figure 2 shows the conceptual encoder–decoder transcription framework inspired by Hawthorne et al. (2021). Our implementation replaces the Transformer decoder with an LSTM + additive attention. Note that we do not include a token for velocity in this project.

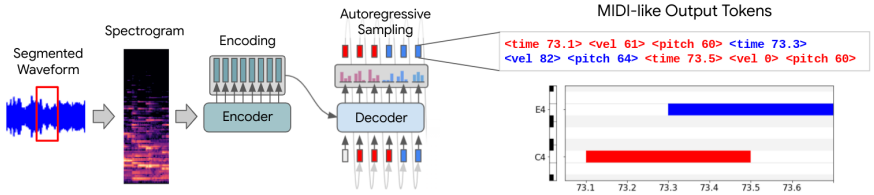

Figure 2: Seq2Seq Model Architecture adapted from Hawthorne et al., 2021

Additive attention (code-matching formulation). We use Bahdanau-style additive attention (Bahdanau et al., 2015). Given the current decoder state s_t and the encoder sequence $\{h_i\}_{i=1}^{T'}$, we compute unnormalized energies

$$e_{t,i} = v^\top \tanh(W_h s_t + W_s h_i),$$

with $W_h \in \mathbb{R}^{d_a \times d_s}$, $W_s \in \mathbb{R}^{d_a \times d_h}$, and $v \in \mathbb{R}^{d_a}$, where d_a is the attention bottleneck dimension. These energies are normalized across encoder time via

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{T'} \exp(e_{t,j})}, \qquad c_t = \sum_{i=1}^{T'} \alpha_{t,i} h_i.$$

In our implementation, W_h and W_s are linear projections of s_t and h_i into a shared attention space, then combined and passed through tanh, followed by a learned vector projection $v^\top$. Importantly, we apply a padding mask before softmax to prevent attending to padded encoder positions. The context vector c_t is then concatenated with the decoder state to produce vocabulary logits:

$$z_t = W_o[s_t; c_t] + b_o, \qquad p_\theta(y_t \mid y_{<t}, X) = \text{softmax}(z_t).$$

This mechanism allows the model to dynamically select which acoustic frames are most relevant to the current token decision. Conceptually, when predicting

a NOTE_ON(p) token, attention should peak near frames containing onset evidence for pitch p; when predicting TIME_SHIFT tokens, attention may distribute more broadly or shift forward to track time progression. Attention also provides interpretability: $\alpha_{t,\cdot}$ can be visualized as a heatmap aligning predicted token positions to encoder frames.

Decoder constraints and inference-time design. Although the neural decoder models $p_\theta(y_t \mid y_{<t}, X)$, unconstrained greedy decoding often produces invalid musical event streams (e.g., repeated NOTE_ON for an already-active pitch, or NOTE_OFF for an inactive pitch), overly dense note bursts, or poor termination behavior. We therefore incorporate a constrained greedy decoder at inference that implements hard validity constraints (masking illegal transitions) and soft biases that regularize sequence structure. Concretely, the decoder maintains a per-sample binary state of active pitches and a counter of note events emitted since the last time shift. Logits for illegal pitch actions are set to $-\infty$ (e.g., if pitch p is active, NOTE_ON(p) is forbidden; if pitch p is inactive, NOTE_OFF(p) is forbidden). We additionally impose polyphony controls such as a maximum number of simultaneously active pitches, and a cap on the number of note events per frame, to reduce false positives. Termination is handled via a minimum-length constraint (masking EOS early) plus an increasing EOS bias schedule after a minimum length, and a hard cutoff that forces EOS beyond a maximum step. These inference-time rules do not change the learned model distribution but act as a structured decoding prior that makes the output sequence better match the grammar implied by the tokenization scheme.

Scheduled sampling. To reduce exposure bias, we implement scheduled sampling (Bengio et al., 2015), where during training the decoder input token at time t is drawn from the model prediction with probability p_{ss} and from the ground-truth token with probability $1 - p_{\text{ss}}$. This forces the decoder to learn to recover from its own mistakes and typically yields more stable autoregressive inference compared to pure teacher forcing.

5.2 Preprocessing

Log-mel front-end. Audio is resampled/segmented and converted into log-mel spectrograms (Section 3.1.2). Each segment yields a matrix $X \in \mathbb{R}^{T \times M}$.

Event tokenization. For each segment, symbolic note annotations are transformed into an event-token sequence. The vocabulary includes: (i) `TIME_SHIFT` tokens that advance time by discrete steps, (ii) `NOTE_ON(p)` and (iii) `NOTE_OFF(p)` for pitch p, plus special tokens `SOS`, `EOS`, and `PAD`. The resulting sequence is padded to a fixed maximum length for batching.

Segmentation. We train on short fixed-length windows sampled from tracks. This increases the number of training examples but also makes long-range musical structure harder to learn; therefore, attention is helpful to relate each predicted event to the relevant portion of the window.

5.3 Experimental Setup

Loss function and class reweighting. We use weighted cross-entropy over tokens:

$$\mathcal{L}_{\text{WCE}} = -\sum_{t=1}^{L} w_{y_t} \log p_\theta(y_t \mid y_{<t}, X),$$

where w_k is a weight for token class k. This addresses the strong class imbalance typical for event-based transcription (many time shifts vs. comparatively fewer note events). In the final setup, time-shift and end-of-sequence tokens receive larger weights than the frequent note events to stabilize decoding length and prevent degenerate solutions (e.g., endless time-shifts or premature termination). Padding and `SOS` are ignored in the loss.

Optimization. Parameters are optimized with AdamW. We use gradient clipping to control exploding gradients and (when required by GPU constraints) gradient accumulation to simulate larger effective batch sizes.

Validation strategy. Full autoregressive decoding on the entire validation set is expensive. During training we therefore compute a *fast validation* each epoch on a small subset of validation batches using a short greedy decode, enabling quick feedback and early stopping. After training, we run a *full decode sweep* (Block 10b in the code) on the complete validation set, tuning decoder constraints and termination biases to maximize validation onset-F1.

Inference: constrained greedy decoding. Naive greedy decoding often yields pathological event streams (too many time shifts, excessive note spam, or poor termination). We therefore use a constrained greedy decoder that: (i) biases time-shift vs. note tokens, (ii) limits polyphony via a maximum

number of simultaneously active notes, (iii) limits the number of note events per frame, (iv) enforces valid note transitions (no `NOTE_OFF` for inactive pitches, no repeated `NOTE_ON` for active pitches), (v) applies an EOS schedule (minimum length, increasing EOS bias, and a hard EOS cutoff). Decoder hyperparameters are tuned on validation and then fixed for test evaluation.

Hyperparameter choices and justification. Our hyperparameters were selected to balance (i) limited data per track segment, (ii) GPU constraints, and (iii) the known instability of autoregressive decoding for dense event vocabularies.

Encoder/decoder dimensions and depth. The decoder hidden size d_s (and embedding dimension) is chosen to be large enough to model event dependencies (e.g., multiple note events between time shifts) but small enough to avoid overfitting and excessive compute. A multi-layer LSTM decoder increases representational capacity; dropout is applied between layers (when > 1 layer) and on the context vector to regularize training. The attention bottleneck d_a ("attn_dim") controls the capacity of the alignment function: too small can underfit alignments, too large increases compute and overfitting risk.

Optimization and stability controls. We optimize with AdamW, which decouples weight decay from gradient updates and is commonly used for sequence models due to stable convergence properties. We apply gradient clipping (global norm) to avoid exploding gradients in recurrent networks. When GPU memory is limited, gradient accumulation is used so that the *effective* batch size is increased without raising peak memory; this tends to reduce gradient noise and can improve convergence.

Teacher forcing vs. scheduled sampling schedule. We start with pure teacher forcing ($p_{\text{ss}} = 0$) to learn a strong conditional next-token model early in training. We then increase p_{ss} linearly up to a cap (e.g., 0.2) to gradually expose the decoder to its own predictions while avoiding destabilizing training too early. The cap prevents training from becoming overly noisy when predictions are still poor.

Loss weighting for token imbalance. Event token distributions are highly imbalanced: `TIME_SHIFT` and sequence-control tokens can dominate, while pitch-specific note events are sparse and distributed across many classes. We therefore use weighted cross-entropy with class weights w_k to prevent degenerate modes. In practice, we assign higher weight to `TIME_SHIFT` tokens to discourage the model from emitting only note events (note spam) without advancing time, and we weight `EOS` to help the model learn reasonable termination behavior. `PAD` and `SOS` are ignored. These weights were treated

as core training hyperparameters because they significantly affect decoded length ratio and onset precision/recall.

Early stopping and validation configuration. Full validation decoding is expensive because it requires autoregressive generation. During training we therefore use a fast validation proxy each epoch: (i) compute teacher-forced loss on a small subset of validation batches, and (ii) compute quick decode-based metrics using a short decode length. This yields a stable early-stopping signal without slowing training by an order of magnitude. After training, we run a full decode sweep on the entire validation set to select inference-time decoding hyperparameters (below).

Inference-time decoding hyperparameters (selected by validation sweep). We treat constrained decoding settings as inference hyperparameters and tune them on the validation set (then freeze for test):

- *Length control:* (`min_len`, `hard_force_eos_at`) and the EOS bias schedule (`eos_bias_base`, `eos_bias_slope`) regulate sequence length and address premature stopping or overly long outputs (diagnosed by the length ratio).
- *Time-shift vs. note balance:* a `time_shift_bias` and optional run penalties discourage pathological runs of time shifts or, conversely, excessive note emission without time advancement.
- *Polyphony constraints:* `max_active_notes` bounds simultaneous note count, while `max_notes_per_frame` limits note bursts between time shifts. A `note_penalty` can further reduce repeated note emissions at the same frame.

These parameters were chosen explicitly to reflect the event grammar implied by the tokenization and to improve onset-F1 by reducing false positives while maintaining adequate recall. The final setting is the one maximizing validation onset-F1 at a fixed tolerance window, and we report that configuration for test evaluation.

5.4 Evaluation Metrics

Token-level metrics. We report:

- **Token error rate (TER):** normalized edit distance between predicted and target token sequences.
- **Micro Precision/Recall/F1 over tokens:** based on token multiset counts (counts of predicted vs. reference tokens).

- **Length ratio:** $\mathbb{E}[|\hat{y}|]/\mathbb{E}[|y|]$, used to diagnose premature EOS or overly long outputs.

Event-level onset metrics. Token metrics do not directly measure musical correctness. We additionally compute **onset-F1** by converting tokens to note events and matching predicted onsets to reference onsets within a tolerance window (e.g. ± 2 or ± 3 frames). We report onset precision, recall, F1, and counts of true/false positives/negatives.

Benchmark comparison and robustness checks. We do not claim a direct state-of-the-art comparison because the dataset subset and segmentation strategy differ from standard full-dataset settings. Instead, we: (i) compare variants of the decoder constraints and EOS scheduling, (ii) perform small hyperparameter sweeps (decode biases and constraints), (iii) inspect attention maps and attention-peak overlays as qualitative robustness checks.

5.5 Results

Training Dynamics and Generalization Behavior. The training and validation loss curves in Figure 3 demonstrate stable optimization behavior. Training loss decreases monotonically, while validation loss decreases during early epochs and begins to plateau after approximately 40 epochs. This divergence indicates moderate overfitting, which is consistent with theoretical expectations for high-capacity sequence-to-sequence models trained on relatively small datasets.

The fast-validation token error rate (TER) in Figure 4 decreases from approximately 0.97 to below 0.90 over the course of training. The steady reduction in TER confirms that the encoder–decoder architecture successfully learns non-trivial sequential structure. However, the absolute level of TER remains relatively high compared to systems trained on the full MusicNet dataset or larger corpora such as MAESTRO (Thickstun et al., 2017b; Hawthorne, 2018). This behavior is expected: additive attention models are data-intensive and require substantial examples to learn precise temporal alignments between acoustic frames and symbolic events.

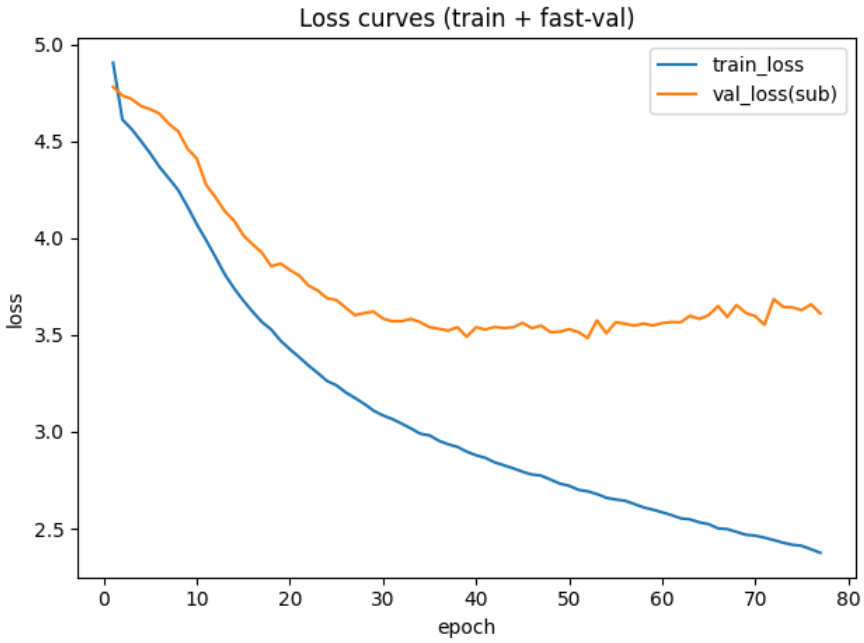

Figure 3: Training and validation loss over epochs.

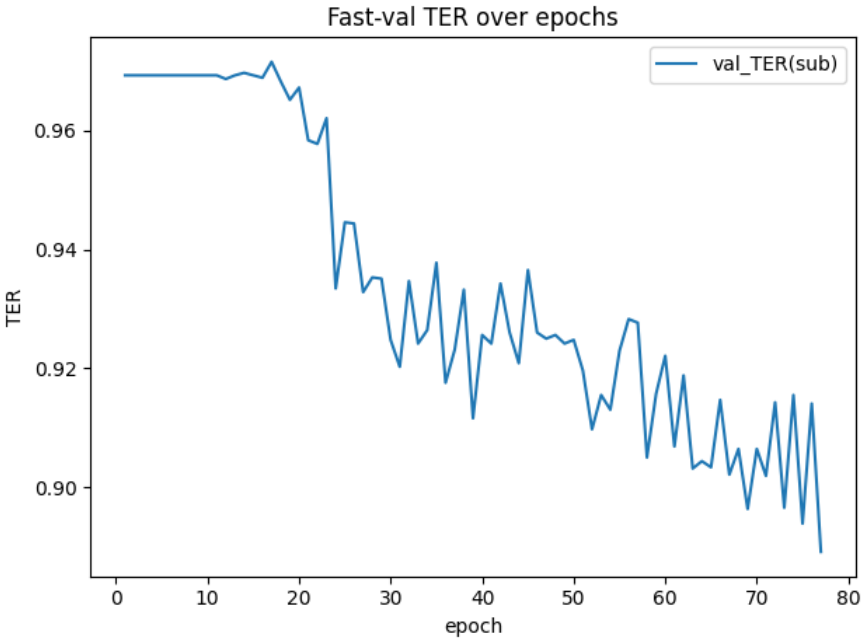

Figure 4: Fast-validation token error rate (TER) over epochs.

Token-Level Versus Event-Level Accuracy. Figure 5 presents token-, onset-, and offset-F1 scores over training. Token-F1 increases steadily to approximately 0.25, indicating that the model learns the general event vocabulary and sequential structure of symbolic music representation.

In contrast, onset- and offset-F1 remain low (around 0.03). This gap between token-level and event-level metrics is theoretically expected. The autoregressive objective optimizes next-token likelihood, which encourages correct symbolic ordering but does not directly optimize precise temporal localization. Prior work has shown that explicit onset objectives or hybrid frame-based supervision are often required to achieve strong onset-F1 performance (Kelz et al., 2016; Hawthorne, 2018).

The observed discrepancy therefore aligns with established findings in the AMT literature and confirms that the model primarily learns symbolic sequencing before mastering precise boundary detection.

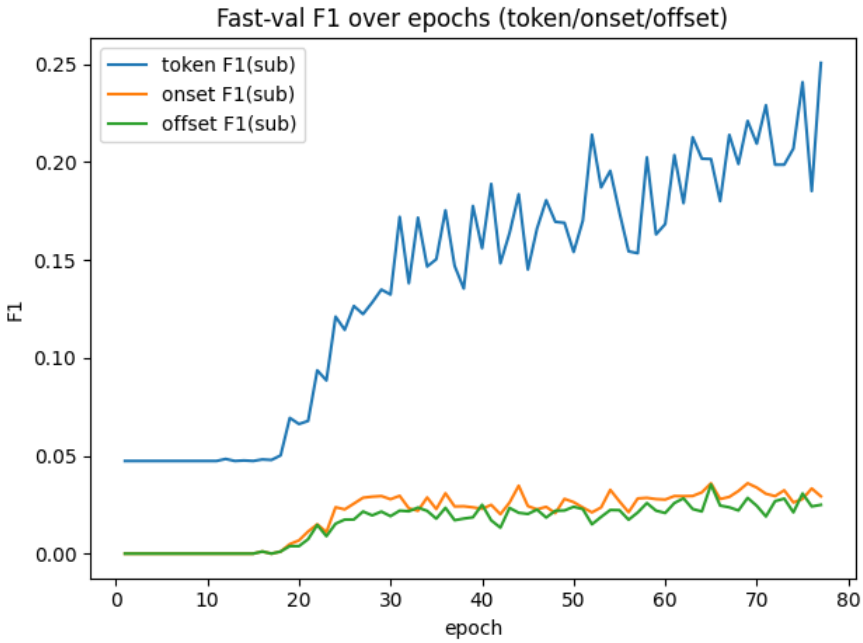

Figure 5: Token, onset, and offset F1 over epochs.

Attention Alignment and Temporal Structure. Under successful training, attention weights are expected to exhibit approximately diagonal structure in the (decoder step $\times$ encoder frame) plane, reflecting monotonic temporal progression. Deviations from this pattern indicate alignment instability or token misprediction. The attention heatmap in Figure 6 exhibits predominantly low-contrast activation, with most weights distributed broadly across encoder

frames rather than concentrated in sharp peaks. This diffuse pattern indicates that the decoder does not strongly localize individual output tokens to specific acoustic time regions.

Such behavior is consistent with the relatively high attention entropy observed in Figure 8. Instead of forming narrow alignment bands, the model maintains uncertain and spread-out attention distributions. In the additive attention formulation of (Bahdanau et al., 2015), sharp alignments typically emerge when sufficient data allows the model to confidently associate encoder representations with output tokens. The absence of clear alignment stripes therefore suggests that the training data is insufficient for learning precise frame-to-event correspondences.

The attention peak progression (Figure 7) further confirms this interpretation. While some temporal drift can be observed, the peak trajectory does not follow a clean monotonic path. Instead, it fluctuates substantially, reflecting unstable alignment and re-attention behavior.

Importantly, this outcome does not indicate architectural failure. Rather, it aligns with theoretical expectations for attention-based models operating in low-data regimes, particularly in polyphonic music transcription where overlapping harmonic content complicates temporal grounding. The combination of diffuse heatmaps and elevated entropy therefore provides evidence that alignment learning has begun but has not yet fully converged.

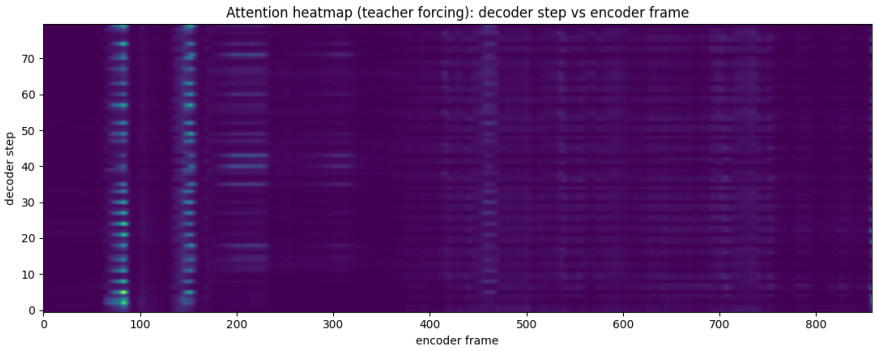

Figure 6: Attention heatmap (decoder step vs encoder frame).

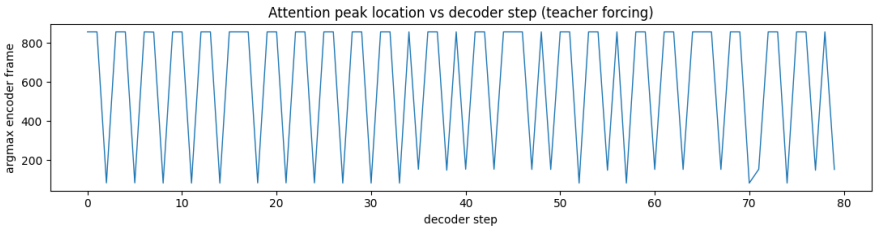

Figure 7: Attention peak location vs decoder step.

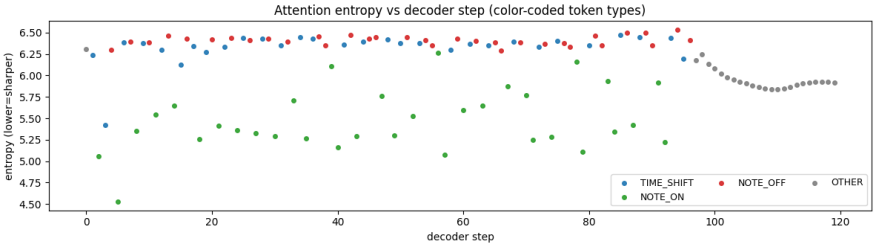

Figure 8: Attention entropy vs decoder step (color-coded by token type).

Pitch Learning and Error Structure. The token-type confusion matrix (Figure 9) shows strong diagonal dominance for TIME_SHIFT and NOTE_OFF, while NOTE_ON exhibits greater confusion. This aligns with prior studies indicating that pitch onset detection constitutes the most challenging component of polyphonic AMT Thickstun et al. (2017b).

More importantly, the pitch-level confusion matrix for the top-20 NOTE_ON classes (Figure 10) reveals a clear but imperfect diagonal structure. Probability mass is concentrated near the diagonal, demonstrating that the model has begun to learn pitch discrimination. However, substantial off-diagonal mass remains, particularly between harmonically related pitches.

This partial diagonal structure is theoretically consistent with the known difficulty of separating overlapping harmonics in polyphonic spectra Kelz et al. (2016). The results therefore indicate that pitch learning has emerged but is not yet fully refined, likely due to limited training data and the absence of explicit spectral disentanglement objectives.

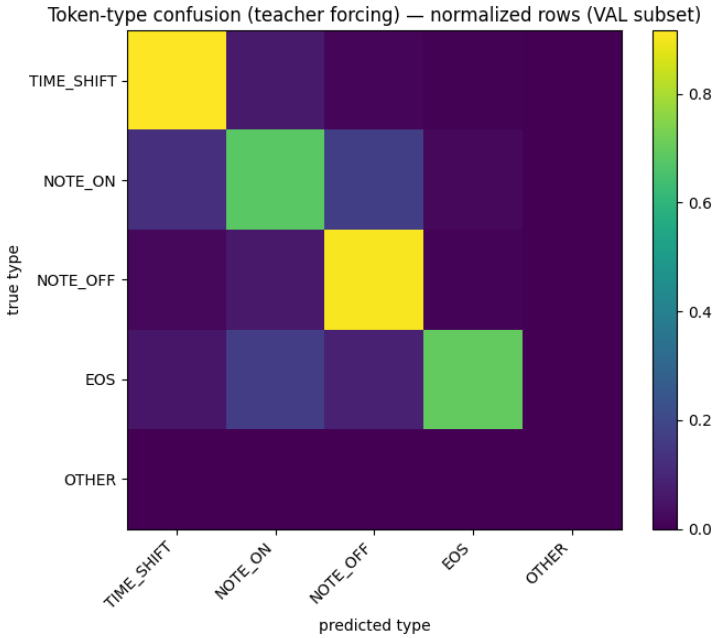

Figure 9: Token-type confusion matrix (normalized rows).

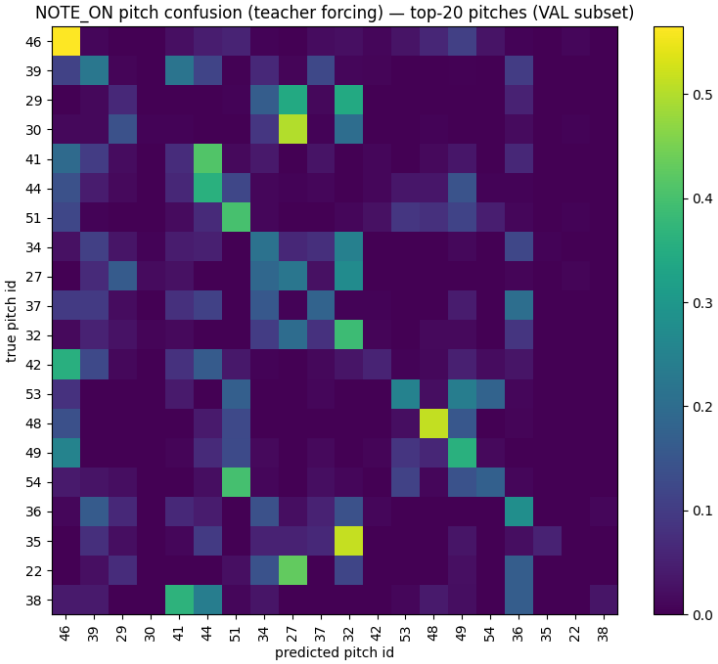

Figure 10: Pitch-level NOTE_ON confusion matrix (top-20 pitches).

Overall Interpretation. In summary, the Seq2Seq model exhibits learning behavior consistent with theoretical expectations for additive-attention architectures. The model successfully acquires symbolic sequence structure and develops emerging pitch discrimination, yet struggles with precise onset localization and sharp temporal alignment.

The qualitative alignment patterns, entropy behavior, and confusion structure collectively support the conclusion that performance is primarily limited by data availability rather than architectural malfunction. The results therefore validate the functional correctness of the implemented additive-attention framework while highlighting the data-intensive nature of attention-based polyphonic transcription.

6 Discussion and Conclusion

This report presented a controlled comparative investigation of two paradigms for polyphonic Automatic Music Transcription (AMT) under constrained data and compute: frame-synchronous CTC decoding and attention-based Seq2Seq event generation. Both approaches used a CNN–BiLSTM encoder, enabling a focused analysis of differences arising purely from output representation, training objective, and decoding strategy.

Empirically, the CTC model with HCQT features and no augmentation achieved the strongest overall performance, reaching a mean Test Token F1 of 0.7145 ± 0.0176 (95% CI: [0.6927, 0.7363]) and mean TER of 0.6858 ± 0.0688 across five random seeds. Convergence was stable, blank dominance transitioned from an expected early high-blank regime to a healthy emission regime, and per-pitch diagnostics confirmed meaningful pitch discrimination across the mid-register. The consistent advantage of HCQT over CQT suggests that explicitly encoding harmonic structure along the channel dimension provides a useful inductive bias in dense polyphonic settings.

The Seq2Seq model, in contrast, learned symbolic sequence structure and demonstrated emerging pitch discrimination but struggled with precise onset localization. Token-level metrics improved steadily during training, indicating that the decoder successfully acquired the event vocabulary and basic structural regularities. However, onset- and offset-F1 remained low, and attention analyses revealed diffuse alignment patterns and elevated entropy. These findings are consistent with the data-intensive nature of additive attention models in highly ambiguous acoustic environments.

Importantly, the Seq2Seq results should not be interpreted as architectural failure. Given the extremely limited 50-track subset and strong solo-piano bias (76% of tracks), the model achieved non-trivial symbolic sequencing

performance under conditions far below typical Transformer-based AMT training regimes (e.g., MAESTRO-scale corpora). Prior work has shown that attention-based token models scale effectively with data (**??**). It is therefore plausible that, under substantially larger and more diverse training sets, the Seq2Seq paradigm would surpass CTC by leveraging its richer event grammar, explicit duration modeling, and global sequence conditioning. In this study, the data regime likely constrained the alignment learning capacity more than the representational capacity of the architecture itself.

Several structural trade-offs between paradigms become clear. CTC enforces monotonic alignment and trains stably with limited supervision, but its pitch-only vocabulary does not encode note duration and collapses polyphony into ordered pitch streams. Moreover, greedy decoding was used throughout; beam search with a pitch-language-model prior could improve sequential coherence without modifying the network. Conversely, the Seq2Seq formulation directly models time shifts, note onsets, and offsets, yielding a musically expressive symbolic representation. However, this flexibility requires reliable alignment learning and exposure-bias mitigation, which appear sensitive to data volume.

The limitations of this study constrain generalisability. The 50-track subset introduces composer and instrumentation bias, and performance on chamber or orchestral recordings may differ substantially. Token F1 for CTC is order-invariant and therefore overestimates musical quality relative to onset-based metrics. The use of greedy decoding restricts both paradigms, and the CTC vocabulary omits duration information.

Overall, under the specific constraint of limited data and moderate compute, frame-synchronous CTC with harmonically informed preprocessing constitutes a more robust and data-efficient solution than autoregressive event generation. However, the Seq2Seq framework demonstrates promising symbolic modeling capacity even in low-resource conditions, and theoretical considerations together with prior large-scale results suggest that its relative advantage would likely grow with increased data, pre-training, and hybrid CTC/attention decoding strategies. The mechanistic analyses presented here — including blank dynamics, gradient flow, per-pitch F1, attention entropy, and confusion structure — provide a diagnostic foundation for understanding not only which paradigm performs better in this regime, but why.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NeurIPS*, 2015.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell. In *ICASSP*, 2016.

Kin Wai Cheuk, Kat Agres, and Dorien Herremans. The impact of audio input representations on neural network based music transcription. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2020. doi: 10.1109/IJCNN48605.2020.9207605. URL `https://doi.org/10.1109/IJCNN48605.2020.9207605`.

Josh et al. Gardner. Mt3: Multi-task multitrack music transcription. *arXiv preprint arXiv:2111.03017*, 2021.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ICML*, pages 369–376, 2006.

Curtis et al. Hawthorne. Onsets and frames: Dual-objective piano transcription. In *ISMIR*, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall, 2nd edition, 2009.

Rainer Kelz, Matthias Dorfer, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. In *ISMIR*, 2016.

Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 1966.

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. Specaugment: A simple data augmentation method for automatic speech recognition. In *INTERSPEECH*, 2019.

Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 1997.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NeurIPS*, 2014.

Michael Taenzer, Andreas Arzt, Gerhard Widmer, Matthias Mauch, and Meinard Müller. Comparing deep models and evaluation strategies for multi-pitch estimation in music recordings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

John Thickstun, Zaid Harchaoui, and Sham M. Kakade. Invariances and data augmentation for supervised music transcription. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017a.

John Thickstun, Zaid Harchaoui, and Sham M. Kakade. Learning features of music from scratch. In *International Conference on Learning Representations (ICLR)*, 2017b.

Christof Weiss, Jordi Pons, and Meinard Müller. Pitch-class multi-pitch estimation for music recordings with deep neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

Frank Zalkow and Meinard Müller. Ctc-based piano transcription with improved note alignment. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

7 Appendix

- Grammerly and ChatGPT have been used to improve grammer and formulation.
- The repository is spereated in three folders: Preprocessing (which includes the code for the subset creation, CTC (which includes the jupyternotebook for the CTC Model), and Seq2Seq (which includes the documented Notebook for the Seq2Seq Model)

A Model Hyperparameters

A.1 Encoder Architecture for Seq2Seq(CNN–BiLSTM)

A.1.1 CNN Front-End

- Convolutional blocks: 3
- Channels: $(32, 64, 128)$
- Kernel size: 3×3, padding=1
- Normalization: BatchNorm2d
- Activation: ReLU
- Pooling: MaxPool(1×2) after each block

After three (1×2) pooling operations, the frequency dimension is reduced by a factor of 8 while the time dimension is preserved.

A.1.2 BiLSTM Encoder

- Hidden size (per direction): 256
- Bidirectional: yes
- Number of layers: 2
- LSTM dropout: 0.2

Because the LSTM is bidirectional, the encoder output dimension is:

$$d_{\text{enc}} = 2 \times 256 = 512.$$

The input dimension to the BiLSTM is:

$$d_{\text{proj}} = 128 \cdot \frac{n_{\text{mels}}}{8}.$$

A.2 CTC Head

- Vocabulary size: 89 (88 pitches + blank)
- Optimizer: Adam
- Learning rate: 1×10^{-3}
- Weight decay: 1×10^{-4}
- Gradient clipping: 5.0
- Blank penalty grid: $\{0, 0.05, 0.10, 0.15\}$

B Seq2Seq Hyperparameters (Final Implementation)

This appendix lists the concrete hyperparameters used in the final Seq2Seq (additive-attention RNN) implementation (Block 10: fast training; Block 10b: full decode sweep).

B.1 Decoder and Attention

- Decoder type: LSTM (autoregressive)
- Decoder hidden size / embedding dimension: $d_s = 192$
- Number of decoder layers: 2
- Dropout: 0.2 (applied as LSTM inter-layer dropout since layers > 1, and as context dropout)
- Attention mechanism: additive (Bahdanau-style)
- Attention projection dimension: $d_a = 256$

B.2 Loss and Class Reweighting

Training uses weighted cross-entropy over the full event vocabulary (PAD and SOS ignored). Let w_k denote the per-class weight. In our final setup:

- $w(\texttt{TIME\_SHIFT}) = 4.0$
- $w(\texttt{NOTE\_ON}) = 1.3$
- $w(\texttt{NOTE\_OFF}) = 1.2$

- $w(\texttt{EOS}) = 3.0$
- $w(\texttt{PAD}) = 0.0$, $w(\texttt{SOS}) = 0.0$
- `CTC_BLANK` (if present in shared vocabulary): $w(\texttt{BLANK}) = 0.0$

B.3 Optimization and Training Procedure

- Optimizer: AdamW
- Learning rate: 3×10^{-4}
- Weight decay: 1×10^{-4}
- Gradient clipping (global norm): 1.0
- Gradient accumulation steps: 3 (effective batch size multiplied by 3)
- Max epochs: 80
- Early stopping patience: 20 epochs (based on fast validation loss)

B.4 Scheduled Sampling Schedule

We use scheduled sampling to mitigate exposure bias. The probability of feeding the model's own prediction as the next input token increases linearly with epoch:

$$p_{\text{ss}}(\text{epoch}) = \min\big(0.20,\ 0.02 \cdot (\text{epoch} - 1)\big).$$

Thus p_{ss} starts at 0 in epoch 1 and reaches its cap of 0.20 by epoch 11.

B.5 Fast Validation Decode Settings (Training-Time Diagnostics)

During training we compute a fast validation proxy using short constrained greedy decoding:

- Maximum decode length: 120 tokens
- Minimum decode length (EOS masked before): 20 tokens
- Validation subset size: first 3 batches per epoch (fast proxy)

Full (slow) validation decoding and constraint tuning are performed separately in Block 10b.

C Event Vocabulary

The event vocabulary consists of:

- TIME_SHIFT(Δt)
- NOTE_ON(p) for MIDI pitches $p \in [21, 108]$
- NOTE_OFF(p)
- SOS, EOS, PAD

Polyphony is represented by emitting multiple NOTE_ON tokens at the same decoder time index prior to advancing time via TIME_SHIFT.

D Reproducibility

Random seeds were fixed for each reported run. Code blocks referenced in the report correspond to the implementation in the submitted Jupyter notebook.

Declaration of Authorship

"I hereby confirm that I have authored this report independently and without use of others than the indicated sources. All passages which are literally or in general matter taken out of publications or other sources are marked as such. All verbatim or in-sentence copies and quotations, as well as all sections that were designed, written and/or edited with the help of AI-based tools, have been identified and verified. Information on the use of AI-based tools: In the appendix of my work, I have listed all AI-based resources with product names and prompts used. I assure that I have not used any AI-based tools whose use has been explicitly excluded in writing by the examiner. I understand that AI- generated content does not guarantee correctness. I remain fully responsible for the content of this work."

Berlin, February 20, 2026

Phillip Olshausen

Livia Kastrati