Universiteit Leiden
The Netherlands

Opleiding Informatica

NoodleGP:

Automatic Guitar Tablature Generation Using Conformers

Peter Branger, s3692965

Supervisors:
1st Supervisor: Dr. Erwin Bakker & 2nd Supervisor: Dr. Michael Lew

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

28/05/2025

Abstract

This paper introduces NoodleGP, a hybrid CNN–Transformer model designed for automating guitar tablature transcription from audio-input. Recognizing the limitations of previous purely convolutional approaches, such as limited global context awareness and inherent string-fret ambiguities, NoodleGP integrates a pretrained ResNet CNN backbone with Transformer-based self-attention layers, aiming to leverage both local spectral-temporal patterns and global sequence dependencies.

Through experimentation on the GuitarSet dataset, the model's capability to resolve ambiguous mappings of pitch to one of multiple potential fret positions is evaluated. The study systematically explores architectural variations, including different depths of ResNet encoders, the number and depth of Transformer layers, and evaluating critical hyperparameters like learning rate, dropout, and batch size. Evaluation metrics include frame-level and note-level F-scores, and tablature disambiguation rates.

Preliminary results indicate that overall transcription accuracy remains modest compared to established CNN-only and attention-augmented baselines, though showing potential, especially scoring higher than the baseline for tablature-generation-specialized metrics. These findings point to interesting directions for improvement, including richer domain adaptation, explicit onset detection modules, and integrated convolution–attention blocks to enhance both local feature extraction and global context modeling.

Contents

1 Introduction

Over the past decade, the advances in of machine learning and Music Information Retrieval has dramatically enhanced our capacity to automatically transcribe acoustic music into symbolic notation, underpinning applications in music education, interactive learning tools, and large-scale musicological analysis. Early methods include the A* search–based fingering arrangement algorithm[BF13] to generate guitar tablature. The method tried to improve fingering ambiguity, although the implementation and approach left much room for improvement. Even when limited to simple note-level transcription, guitar tablature must resolve polyphonic textures across six strings and contend with inherent ambiguities, most notably that identical pitches may be produced by multiple string-and-fret combinations, rendering the task a challenging problem due to source ambiguities.

Given the laborious and time-consuming process of producing accurate guitar tablature by hand, automated transcription techniques promise to greatly broaden access to high-quality tabs. Moreover, these methods can be employed to refine existing tablature, especially for obscure repertoire that has typically been transcribed only once and may contain errors. Finally, as new music is released at an ever-increasing rate, automated systems offer a scalable solution to clear the substantial backlog of non-transcribed works.

However, existing CNN-only transcription models evaluated on GuitarSet[XBP+18], such as TabCNN[WK19], Tab-Estimator[KHT22], and FretNet[CHK23], remain hampered in their performance by their inherently local inductive bias: convolutional filters operate over limited receptive fields and must either rely on very deep or dilated architectures to capture long-range temporal dependencies, which in turn increases computational cost and risks overfitting on the modestly sized and inherently unbalanced GuitarSet[XBP+18]. Consequently, these approaches often struggle to resolve the string-and-fret ambiguities inherent in polyphonic passages, since local feature aggregation lacks the global context needed to disambiguate simultaneous note events and detect the correct fingering across musical phrases. The absence of an explicit mechanism for modeling interactions spanning entire spectrogram sequences further limits their ability to produce consistent and musically plausible tablature, motivating the exploration of transformer-and-self-attention–based architectures that have been shown to be capable of capturing global dependencies[VSP+23].

In this work it has been studied how Transformer-based architectures[VSP+23], leveraging self-attention to model global context, can be applied to simple guitar tablature transcription and evaluated the proposed novel architecture against baseline methods. The main research question of this paper is: "How well can a Hybrid-Transformer model resolve string-fret mappings?". A tailored Transformer model that integrates spectrogram-based encoders with sequence-to-sequence decoding strategies is proposed, benchmarking performance is evaluated using transcription and pitch accuracy, and mean opinion score of experienced guitarists. The conducted experiments explore the impact of using pretrained ResNet[HZRS15] encoders as spectrogram feature extractors, and Transformer models with different numbers of self-attention layers and hidden dimensions. A grid-search for the tuning hyperparameters, such as learning rate, batch size, and dropout rate, has been performed in order to determine the optimal configuration for the proposed architecture.
The rest of this paper is organized as follows: Section 2 discusses related work, focusing on foundational papers, convolutional models, domain adaptation techniques, the Conformer architecture, and available annotated guitar datasets. Section 3 provides essential background information on guitar tablature, evaluation metrics, and the Constant-Q Transform spectrograms[SK10]. Section 4 conceptually discusses the baseline models that function as the most direct comparison to the proposed model. Section 5 introduces the proposed model, NoodleGP, including its research motivation, architectural details, and full implementation. Section 6 describes the GuitarSet[XBP+18] dataset, the applied data preprocessing, augmentation strategies, validation pipeline, dataset wrapper implementa-

tion, and dataset splitting methodology. Section 7 describes the experimental setup including the training loop, the performed grid-search, and results of the hyperparameter tuning. In section 8 the experimental results are described, discussed and compared against the baseline, the user evaluation results are also shown and discussed. Finally, Section 9 describes the conclusions and contributions, discusses limitations, and proposes directions for further research.

2 Related Work

A wide range of deep-learning methods have been developed for automatic tablature transcription on the GuitarSet dataset, ranging from frame-level CNN models, to attention-based note-level systems, continuous-pitch estimators, domain-adaptation pipelines, and semi-supervised frameworks leveraging unlabeled audio. Before the advent of widely available neural networks, other now classical techniques, such as SVMs[PE06], HMMs[BTSB12][BKTB12], etc, were used to lay the foundation of a complex and interesting field of audio processing.

2.1 Seminal Papers

Several influential works have shaped automatic guitar transcription through various innovative methods. Below we will discuss a few of the papers that, while by today's standards are outdated, shaped the field substantially.

Poliner and Ellis[PE06] introduced supervised transcription using frame-wise Support Vector Machines (SVMs) combined with a Hidden Markov Model (HMM) for temporal continuity. This approach pioneered multi-label classification for music transcription, influencing guitar-focused methodologies.

Barbancho et al.[BTSB12] leveraged guitar-specific inharmonicity analysis to determine precise fret and string combinations, directly generating accurate tablature. This method explicitly tackled fret assignment using guitar acoustics.

Barbancho et al.[BKTB12] extended their work to chord sequences using an HMM framework, incorporating multipitch analysis, chord transition grammar, and playability constraints to optimize realistic finger movements.

Yazawa et al.[YSN+13] enhanced transcription realism by incorporating explicit playability constraints using dynamic programming as the core technique. This approach ensured physically feasible fingerings, significantly improving transcription accuracy.

2.2 Convolutional Approaches to Automatic Tablature Transcription

Recent advances in automatic guitar tablature transcription have predominantly utilized convolutional neural networks (CNNs) to process audio inputs and predict corresponding string-fret combinations. Notable models in this domain include TabCNN[WK19], Tab-Estimator[KHT22], and FretNet[CHK23], each introducing distinct architectural innovations while operating within the convolutional paradigm.

TabCNN[WK19] employs a series of 2D convolutional layers to analyze CQT-spectrograms of guitar audio, directly estimating the fret positions for each string at every time frame. Building upon this foundation, *Tab-Estimator[KHT22]* integrates Conformer-style self-attention mechanisms and a beat-informed quantization layer, facilitating both frame-level and note-level tablature predictions through a multi-task learning approach. *FretNet[CHK23]* diverges by framing tablature transcription as a continuous pitch estimation problem, utilizing a dual-head output to predict discrete activations and relative pitch deviations, complemented by an onset-detection module for event grouping.

Despite their contributions, these models share inherent limitations characteristic of convolutional architectures. CNNs primarily capture local patterns due to their constrained receptive fields, necessitating deeper networks or dilated convolutions to model long-range dependencies—approaches that can increase computational complexity and the risk of overfitting. In contrast, Transformer-based architectures, with their self-attention mechanisms, offer a more efficient means of capturing global context and long-term temporal relationships across entire spectrogram sequences, presenting a promising alternative for future developments in guitar tablature

transcription.

2.3 Groove modeling

A particularly relevant study is "Automatic Composition of Guitar Tabs by Transformers and Groove Modeling"[CHHY20], which employs a purely Transformer-XL architecture—12 layers deep with 8 attention heads and roughly 41 million parameters—to generate fingerstyle guitar tablature. By extending the usual MIDI event vocabulary to include STRING, FRET, TECHNIQUE, and a learned GROOVE token (clustered onset-density patterns), the model is explicitly encouraged to capture rhythmic "groove" alongside accurate pitch and fingering assignments. In objective tests, groove-aware variants achieved nearly 80% accuracy on bar-level onset patterns, and in listening studies their continuations earned mean-opinion scores almost indistinguishable from human-composed excerpts, demonstrating that sufficiently deep Transformer architectures can excel at both precise transcription tasks and nuanced rhythmic modeling.

Domain Adaptation[RED24]

An alternative strategy for guitar transcription involves adapting high-resolution piano transcription models to guitar audio. This process entails aligning the activations of a pre-trained piano model with guitar scores using time-warping techniques, thereby generating pseudo-labeled guitar data for fine-tuning. This approach leverages the temporal precision of piano models, particularly their accurate onset detection capabilities. However, it's important to note that the evaluation metrics used in this context differ from those commonly employed in guitar transcription studies. Specifically, onset-only F1 scores assess whether predicted note onsets occur within a specified time window of the ground truth, without considering pitch, string, or fret accuracy. In contrast, tablature F1 scores evaluate both the timing and the correctness of string-fret assignments. Consequently, direct comparisons between onset-only and tablature F1 scores are not meaningful. Nonetheless, the strong temporal detection performance of piano-trained models when applied to guitar audio suggests that integrating their onset detection capabilities with guitar-specific pitch and fingering estimation could enhance the overall accuracy of guitar tablature transcription systems.

Conformer

In the Conformer[GQC+20] architecture was proposed, a hybrid of convolutional neural networks with self-attention layers for end-to-end speech recognition. On the LibriSpeech benchmark, an established benchmark for automatic speech recognition (ASR) models, Conformer based ASR-models established new state-of-the-art word error rates, demonstrating that combining CNNs and attention yields significant gains over pure Transformer or CNN models. The guitar-tablature transcription pipeline proposed in this work similarly aims to leverage the complementary strengths of CNNs and transformer architectures.

2.4 Annotated Guitar Datasets

DadaGP[SKC+21] is a large symbolic guitar corpus comprising 26 181 song scores in GuitarPro formats (gp3–gp5) spanning 739 musical genres. It provides an encoder/decoder that converts these files into an event-based token sequence—encoding note on/off events, string/fret assignments, tempo changes, and expressive techniques—enabling sequence-model training for generative and transcription tasks.
The *IDMT-SMT-Guitar*[KME23] dataset contains recordings from seven electric and acoustic guitars under varying pickup settings, including isolated notes, chords, short licks, and polyphonic excerpts. Each audio file is paired with XML annotations of pitch, string and fret positions, plucking style (finger, pick, mute), and expressive effects such as bends and vibrato, supporting fine-grained supervised transcription research.
GAPS[RGED24] offers 14 hours of real classical guitar performances by over 200 players, each audio file precisely aligned to high-resolution MIDI "score" data and accompanied by performance video. This diverse corpus—freely available under a permissive license—provides note-level timestamps for supervised and zero-shot transcription benchmarking.
EGDB[CHH+22] contains 240 tablature excerpts played on an electric guitar and captured via hexa-

phonic pickup, then rendered through six amplifier presets to total 118 minutes of audio. Each recording is matched with detailed tablature annotations, enabling evaluation of transcription models under varied timbral conditions.

SynthTab[ZZCD24] synthesizes over 6700 hours of guitar audio across 15 211 tracks and 23 timbres by rendering DadaGP's tokenized tablature with commercial guitar plugins. It preserves original fingerings and technique markings, producing vast pseudo-labeled data.

AnimeTAB[ZJX22] is a MusicXML fingerstyle guitar dataset focusing on anime and video-game soundtracks, comprising 412 full tracks and 547 structural clips (intro, verse, chorus, bridge). It includes an analysis toolkit for melody/bass extraction, key detection, and chord labeling.

GuitarSet[XBP+18] was selected as the benchmark dataset in this our own experiments, because it is both widely used in atuomatic guitar tablature-generation research, and publicly available under an open license. The dataset is recorded with a hexaphonic pickup that yields individual string signals, and richly annotated with time-aligned string/fret positions, pitch contours, chord labels, beats/downbeats, and playing style metadata. Its widespread adoption in the guitar transcription research community provides a well-established baseline for comparison and reproducibility, ensuring that our results can be meaningfully benchmarked against prior research.

3 Fundamentals

In this section we introduce the fundamental concepts for defining the problem of the guitar transcription task. Furthermore, we introduce the fundamental building blocks of the proposed solution. This includes integral concepts, the evaluation metrics used to compare the performance of the proposed method are defined, and methods used in this work.

Guitar Tablature At the root of the problem lies the subject of the transcription itself. As opposed to the musical notation most are familiar with, guitar tablature is a form of musical notation indicating instrument finger placement, rather than the played pitch. Each line represents one of the guitar's six strings, and numbers on the lines denote where on the neck of the guitar to press. The combination of string and pressure point is a fret, small metal bars that bind the string to a specific pitch. An example of the notation is depicted in Figure 1. The reason this form of notation is used as the standard for most guitarists, is that a guitar has several ways of playing the same pitch. For instance, the 'E' note can be played on five different locations on the guitar, making it potentially difficult to intuitively tell which one would need to be played when given classical musical notation. Another way of phrasing this, is to say that the problem of deciding where to place your finger on the guitar is inherently ambiguous.

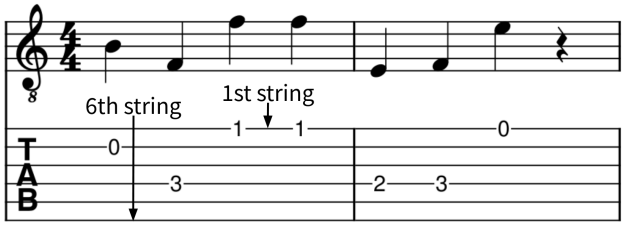

Figure 1: Example comparison between sheet music and guitar tablature (Tab), where each horizontal line represents a string.

Perfomance measures. To measure the performance of the model through experimentation, as well as to be able to compare to the baseline methods, performance measures for tablature transcriptions, precision, recall, F1 and tablature disambiguation rate (TDR) as defined in TabCNN[WK19] are used.

The guitar used in GuitarSet[XBP+18] is capable of producing 120 different string-fret combinations, calculated by multiplying the 6 strings by the fret-count (20). These string-fret combinations account for a total of 44 different pitches. Each training instance consists of two items. First a so-called "frame", which is a small excerpt of larger annotated audio segments which contains the notes played in that time-window, as well as the information of when which notes are played where on the guitar. Second it contains note-level data, a singled out note of varying duration with the pitch and string-fret

combination. The training instance initially contains only the audio segment of the instance (in the form of a constant-Q Transform spectrogram of the frame and note), after prediction the model compares it to the ground truth. The test instances contain only the audio segment, and are not shown the corresponding annotation. e is a vector of all ones, Z denotes a matrix of $N \times 120$, where the 120 comes from the total amount of string-fret combinations since there are 6 strings and 20 frets per string, Z_{gt} is the matrix containing the ground truth, Z_{pred} is the matrix containing the predictions, N is the amount of testing frames for this specific instance, $\odot$ denotes element-wise multiplication, and finally T denotes the transpose operator.

Tablature precision, denoted as p_{tab}, is defined in Eq 1. It is the ratio between the number of correctly identified pitches (numerator) and the total number of pitches (denominator), measuring how often identified pitches are actually correct.

$$p_{tab} = \frac{e^T(Z_{gt} \odot Z_{pred})e}{e^T Z_{pred} e} \qquad \text{(Eq 1)}$$

Tablature recall, denoted as r_{tab}, is defined in Eq 2. It is the ratio between the number of correctly identified string-fret combinations and the the total number string-fret combinations in the ground truth, measuring how often the model manages to predict string-fret combinations actually present in the signal.

$$r_{tab} = \frac{e^T(Z_{gt} \odot Z_{pred})e}{e^T Z_{gt} e} \qquad \text{(Eq 2)}$$

Tablature F-measure, denoted as f_{tab}, is defined in Eq 3. It is the harmonic mean of tablature precision and recall, measuring the overall performance of the model when it comes to predicting tablature.

$$f_{tab} = \frac{2 p_{tab} r_{tab}}{p_{tab} + r_{tab}} \qquad \text{(Eq 3)}$$

Tablature Disambiguation Rate, denoted as TDR, is defined in Eq 4. It is the ratio between the total number of correctly predicted string-fret combinations and the total number of correctly predicted pitches, measuring how often correctly predicted pitches are assigned the correct tablature, in other words, how often the pitches are assigned the correct string-fret combination option. Y denotes a matrix of $N \times 44$, where the 44 represents the total unique pitches a guitar can produce, again gt and $pred$ refer to matrices containing the ground-truth and predicted values.

$$TDR = \frac{e^T(Z_{gt} \odot Z_{pred})e}{e^T(Y_{gt} \odot Y_{pred})e} \qquad \text{(Eq 4)}$$

Constant-Q Transform (CQT) Spectrogram[SK10], is a spectral analysis technique that, unlike the Short-Time Fourier Transform's fixed linear frequency bins, uses geometrically spaced frequency bins whose bandwidths grow with frequency, yielding a constant ratio of center frequency to bandwidth (the "Q" factor) across all bins. This design aligns more naturally with musical pitch perception, each octave is subdivided into the same number of bins so that for instance, the distance between A4 and A5 occupies the same number of bins as between C5 and C6.

In practice, a CQT-spectrogram of a guitar recording will display time on the horizontal axis, logarithmically spaced pitch bins on the vertical axis, and the magnitude of each bin's energy as intensity, giving a clear, musically meaningful depiction of harmonic content over time.

The CQT-spectrogram has proven to be highly effective in previous research on the subject[KHT22][WK19][CHK23], therefore we adopt it in our audio-preprocessing stage.

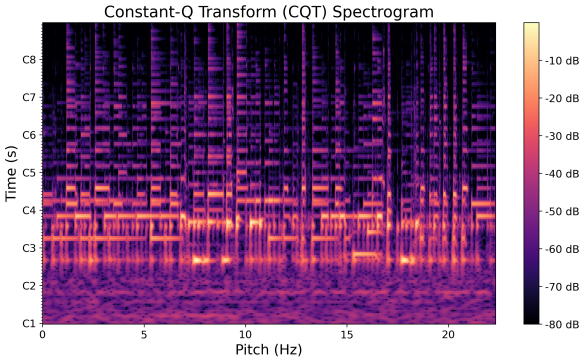

Figure 2: Example CQT spectrogram of one of the GuitarSet audio files (00_BN1-129-Eb_comp_mic.wav) and its harmonics, x-axis represents the note in frequency, the y-axis denotes time, the brightness represents the intensity of the note-frequency and its harmonics in decibels.

4 Baseline methods

In this section we discuss the baseline methods used in our experiments, and their conceptual differences to NoodleGP. We discuss the methods in chronological order, starting with TabCNN[WK19], then Tab-Estimator[KHT22], and finally FretNet[CHK23]. In our experiments we compare the performance of our proposed network NoodleGP to these baselines.

TabCNN[WK19] employs a lightweight convolutional neural network to estimate, for each fixed-length Constant-Q Transform window, independent string-level fret classes, capturing local spectral patterns effectively but lacking mechanisms to model long-range temporal dependencies across frames. Because convolutional kernels have a limited receptive field, deep or dilated layers are required to incorporate broader context, which increases computational cost and heightens overfitting risk on datasets of modest size such as GuitarSet (360 excerpts, 30 s each). This local inductive bias also offers no explicit mechanism for enforcing fingering consistency across nonadjacent notes, leading to occasional physically implausible string-fret assignments when context spans multiple musical phrases.

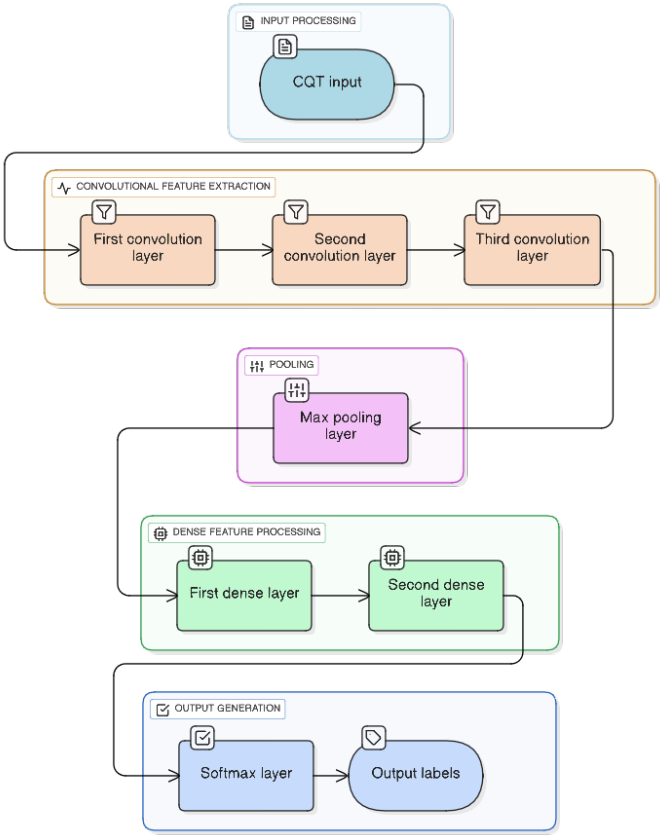

Figure 3: TabCNN[WK19] global model architecture. Figure adapted from the TabCNN paper.

Tab-Estimator[KHT22] integrates self-attention (in a Conformer-inspired block) with beat-informed quantization in a multi-task framework that jointly predicts frame-level and note-level outputs, thereby capturing both local and global dependencies and aligning predictions with musical meter. Beat-informed quantization ties note predictions to a metric grid, improving timing accuracy in strict-tempo performances but risking misalignment on rubato passages or recordings with unreliable beat annotations. The explicit modeling of global interactions enhances consistency of string-fret assignments across phrases, but the quadratic complexity of self-attention and dependency on beat tracking can increase runtime and limit robustness in real-time or noisy scenarios.

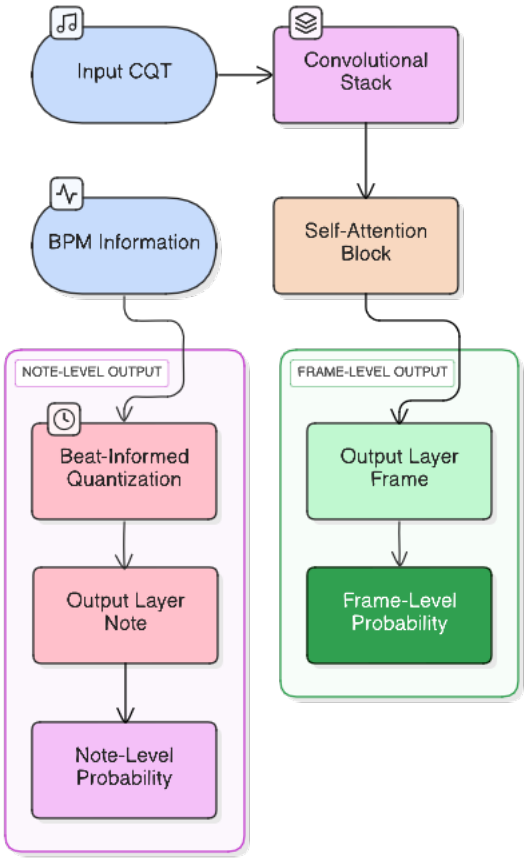

Figure 4: Tab-estimator[KHT22] pipeline. Figure adapted from the Tab-Estimator paper.

The convolutional stack of tab-estimator consists of several convolutional blocks, followed by max-pooling, dropout layers and a linear layer.

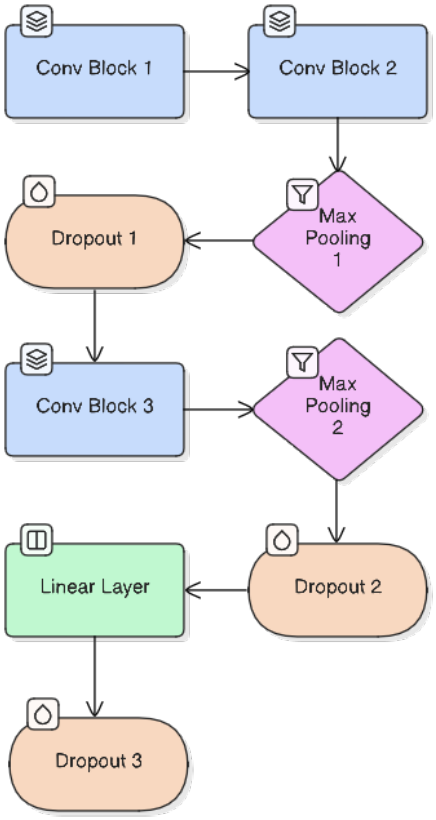

Figure 5: Tab-estimator[KHT22] convolutional stack. Figure adapted from the Tab-Estimator paper.

The convolutional blocks within the convolutional stack consist of a 2 dimensional convolutional layer, followed by batch-normalization, and finally a non-linear layer, that being a ReLu activation function.

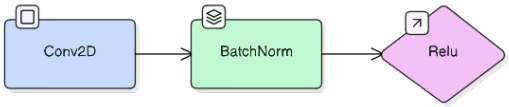

Figure 6: Tab-estimator[KHT22] convolutional block. Figure adapted from the Tab-Estimator paper.

FretNet[CHK23] advances beyond discrete fret classification by predicting continuous-valued pitch contours anchored to string-fret combinations, enabling representation of expressive techniques such as bends and vibrato, and grouping these streams into note events via onset detection and clustering. Continuous-valued outputs offer finer pitch trajectories but may propagate ground-truth annotation noise unless robust clustering and smoothing are

applied. The dual-head architecture also yields competitive tablature-estimation metrics yet introduces additional inference complexity to enforce playability constraints and manage clustering artifacts.

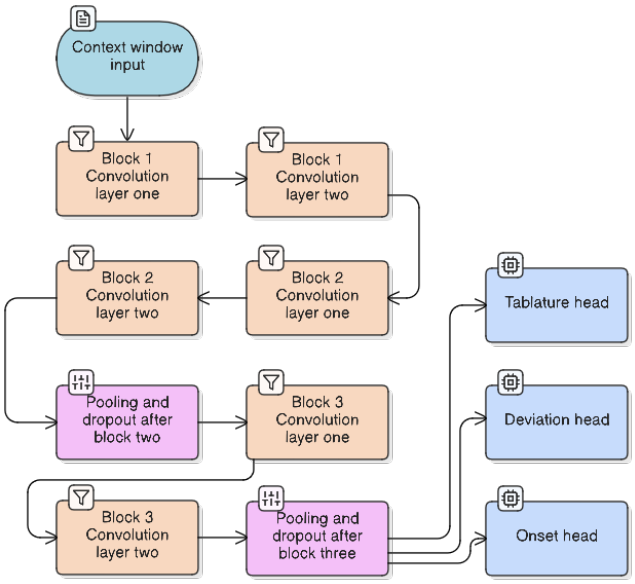

Figure 7: Fretnet[CHK23] model architecture. Figure adapted from the FretNet paper.

The proposed *Transformer–CNN hybrid architecture* leverages a pretrained ResNet backbone to extract local spectral–temporal features and employs sinusoidal positional encodings with multi-head self-attention to integrate global context, striking a balance between efficient local pattern extraction and comprehensive sequence modeling. While this hybrid approach benefits from parameter efficiency relative to pure Transformers and improved contextualization compared to CNN-only models, it may still face challenges in handling out-of-distribution audio variations and extreme expressive techniques without further domain adaptation or data augmentation. Continuous efforts in domain adaptation—such as aligning piano-trained models to guitar via score-activation matching—offer promising directions to enhance generalizability of these transcription systems.

5 NoodleGP

This chapter details our novel proposed method for automatic tab transcription NoodleGP, consisting of a ResNet[HZRS15] CNN, functioning as an encoder, and a Transformer leveraging self-attention[VSP+23], as depicted in Figure 10.

5.1 NoodleGP Detailed Architecture

NoodleGP: CNN Module. For the CNN module, the model utilizes a pre-trained multi-layer CNN, namely ResNet[HZRS15] is used. Resnet's modular design includes variants such as ResNet-18 (where the 18 refers to the amount of learnable layers), all the way up to ResNet-152, allowing systematic exploration of model capacity, depth, and performance in alignment with computational constraints. The bottleneck residual blocks in deeper ResNet variants utilize 1×1 convolutions to compress and then expand feature dimensions, reducing parameter count and computational cost without sacrificing representational power. With ResNet being trained on datasets such as ImageNet-1k, which consists of over 1 million training images, the already trained weights have proven to be widely applicable to different tasks. Finally, its hierarchical, multi-scale feature maps capture both fine-grained details and broader spectral-temporal patterns, providing effective embeddings for the subsequent Transformer modules.

Due to memory constraints and practical experimental considerations, ResNet-18, the smallest ResNet variant was chosen. ResNet-18 begins with a 7×7 convolutional layer comprising 64 filters, stride 2, and padding 3, which reduces the input resolution from 224×224×3 to 112×112×64. A following 3×3 max-pooling layer (stride 2, padding 1) then yields 56×56×64 feature maps.
The main body of ResNet-18, depicted in Figure 8 consists of four stages (conv2_x through conv5_x), each containing two "basic blocks." Every basic block comprises two 3×3 convolutional layers, each followed by batch normalization and ReLU activation, preserving the same number of channels within the stage. In stages where spatial downsampling occurs (e.g., transitioning from 56×56 to 28×28), the first convolution within the block applies stride 2, and the identity shortcut employs a 1×1 projection convolution to match the doubled channel dimension (e.g., from 64 to 128). Subsequent stages follow this pattern, doubling filters to 256 and 512

in conv4_x and conv5_x respectively. The last layer (layer conv4_x) is unfrozen from the start, to allow for the model to be fine-tuned to better extract features from the spectrograms.

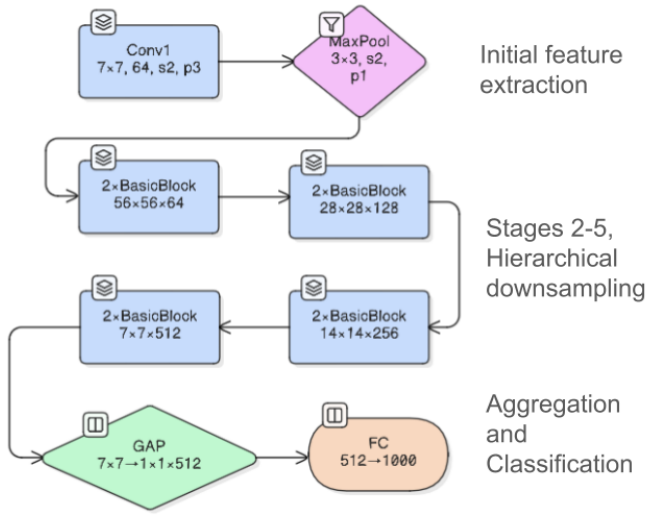

Figure 8: ResNet-18[HZRS15] pipeline.

ResNet's modular architecture scales from ResNet-18 up to ResNet-152, allowing systematic exploration of model capacity and depth under computational constraints. In deeper variants, bottleneck residual blocks introduce additional 1×1 convolutions to first compress and then expand feature dimensions, further reducing parameter count and computational cost without sacrificing representational power. ResNet also benefits from extensive community support and availability of pretrained weights in frameworks such as PyTorch and TensorFlow/Keras, facilitating rapid prototyping and reproducibility. Trained on the ImageNet-1k dataset, which contains approximately 1.28 million training images, these pretrained weights have demonstrated broad applicability across diverse vision tasks. Finally, ResNet's hierarchical, multi-scale feature maps capture both fine-grained details and broader spectral-temporal patterns.

NoodleGP: Transformer Module. For the Transformer module, see Figure 9, the encoder–decoder Transformer framework as described in "Attention Is All You Need"[VSP+23] was selected. This choice was motivated by its multi-head self-attention mechanism, which excels at modeling long-range dependencies across entire sequences, something which reasonably seemed to be essential for capturing the intricate yet repeating patterns found in music theory. Its easily parallelizable architecture[VSP+23](abstract) that facilitates full use of modern GPU hardware to accelerate training made experimentation efficient and fluid. The Transformer's configurable depth, number of attention heads, and feed-forward dimensions allowed me to systematically vary model capacity and study its impact on transcription performance. The well-documented Pytorch implementation facilitated the integration with the CNN-encoder.

In our instantiation, seen in Figure 9 the TransformerModule uses an embedding dimension of 1024 It applies 8 parallel attention heads. We employ 2 encoder layers and 6 decoder layers, as the ResNet is the main encoder of the architecture. Each layer's position-wise feed-forward network has a size of 4096. The values for the embedding dimension (ED) and feed-forward network (FFN) were chosen so that the FFN is 4 times as large as the ED, as this is recommended by the authors of "Attention Is All You Need"[VSP+23]. The ED value was chosen to maximize the depth within the memory and time constraints. The 6 decoder layers and the head-count were chosen as this is also the discussed size in "Attention Is All You Need", the 2 encoder layers were chosen as the ResNet-18 encoder functions as the encoding, as well as to speed up inference.

NoodleGP: Positional Encoding Module. For the Positional Encoding module, see Figure 9 the fixed sinusoidal scheme from "Attention Is All You Need"[VSP+23] was adopted, which injects token order into the Transformer without introducing any extra learnable parameters by relying solely on deterministic sine and cosine functions. This approach keeps the model compact and reduces the risk of overfitting, while its continuous formulation allows seamless transition to sequence lengths beyond those seen during training. The sinusoidal basis also embeds a built-in relative-position bias, enabling the attention mechanism to infer distances between tokens, enabling the model to capture the repeating patterns in musical sequences. Because the transitions between notes follow a pattern (some notes sounding harmonious and some not), the chances of a certain

note being next in a series of notes isn't equal for all. By registering the full positional matrix as a non-learnable buffer, the encoding adds negligible computational and memory overhead. Integration with PyTorch's native API is also straightforward, a simple slice-and-broadcast in the forward pass aligns perfectly with inputs of shape (S, B, D), and because the encodings are fixed at initialization, every experiment remains fully reproducible without reliance on random seeds for positional weights. These characteristics make sinusoidal positional encoding an ideal complement to the ResNet-derived feature embeddings and the standard encoder–decoder Transformer described above.

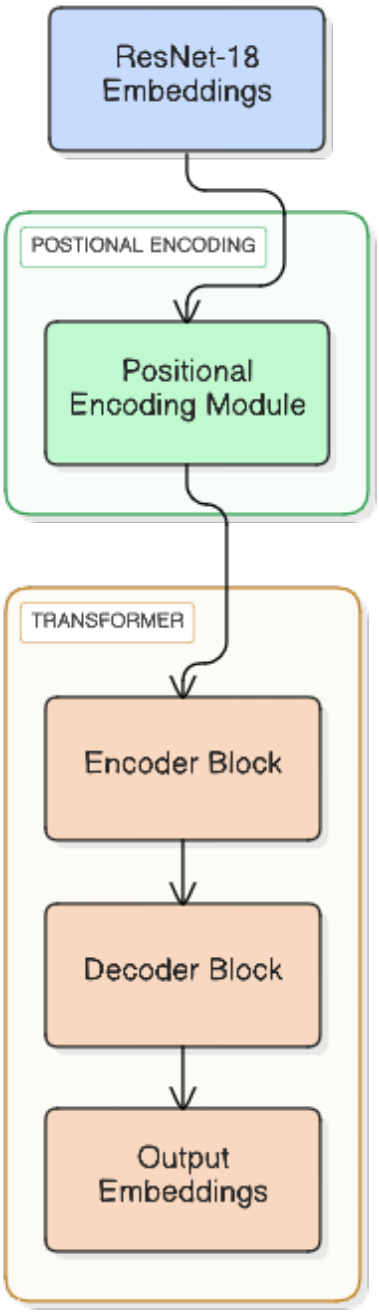

Figure 9: Pipeline of the ResNet-18 encoding through the Positional Encoding module, to the transformer.

5.2 NoodleGP Implementation

In this subsection, an overview of the end-to-end structure of the ResNet–Transformer hybrid model is given, illustrating how the individual modules introduced earlier are connected to form the complete

transcription pipeline.

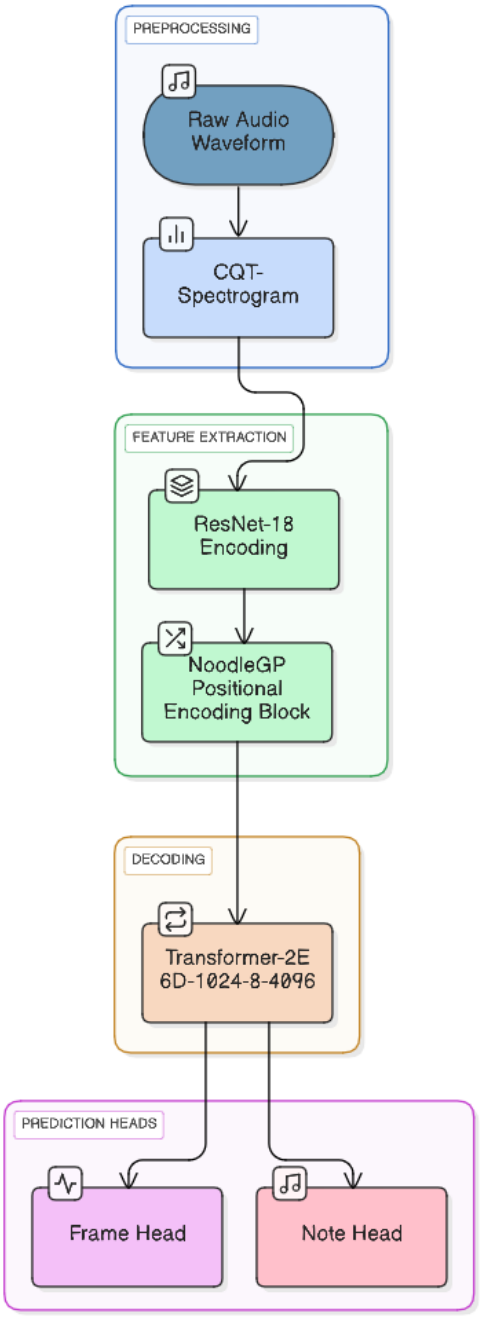

Figure 10: End-to-end transcription pipeline. Note: The transformer uses an embedding dimension of 1024, it applies 8 parallel attention heads, employs 2 encoder layers and 6 decoder layers, each layer's position-wise feed-forward network has a hidden size of 4096.

1. **Backbone Extraction:** Input spectrogram tensors of shape (B, C, H, W) are passed through a ResNet-18 encoder (all layers initially frozen), yielding feature maps of shape $(B, 512, H', W')$.

2. **Finetuning:** Along with the transformer, the script unfreezes `backbone.layer4` and adds its parameters to the optimizer at full learning rate, enabling higher-level CNN features to be fine-tuned.

3. **Sequence Projection:** The feature maps are mean-pooled over the height dimension to produce a sequence of length $S = W'$, then each 1024-dimension vector is linearly projected to the Transformer model dimension d_{model} and augmented with sinusoidal positional encodings.

4. **Transformer Encoder–Decoder:** A custom TransformerModule processes the source sequence (with padding mask) through configurable encoder and decoder layers; the decoder re-uses encoder outputs directly, with no separate token embedding.

5. **Dual Classification Heads:**

   - *Frame Head:* Multi-layer-perceptron (MLP) (Dropout $\rightarrow$ 128 $\rightarrow$ ReLU $\rightarrow$ Dropout $\rightarrow$ 6×21) applied per encoder time step, yielding per-frame softmax logits over 21 frets for each of the 6 strings.
   - *Note Head:* Encoder outputs are decimated via linear interpolation to a fixed n_{notes} length, then the same MLP structure predicts per-note string–fret softmax distributions.

In short, NoodleGP-18 combines a lightweight, partially unfrozen pre-trained ResNet-18 encoder with a compact, sinusoidally-aligned Transformer to turn spectrograms into temporally coherent token sequences, then uses dual MLP heads to output frame and note-level string-fret predictions.

6 Data and Preprocessing

This section details our end-to-end preparation of the GuitarSet dataset for transcription modeling.

We begin by converting hexaphonic recordings and JAMS annotations into unified NPZ archives containing time–frequency features and quantized tablature. Next, we describe the data augmentation used. We then describe the wrapper for the NPZ files, a custom QuantizedNpzDataset and DataLoader that handles batching, padding, and tokenization. Finally, a reproducible 80/10/10 train/validation/test split ensures consistent evaluation.

6.1 GuitarSet

GuitarSet is an annotated dataset for guitar transcription, which leverages hexaphonic pickup recordings to automate note-level annotation and support detailed analysis of polyphonic guitar performance. It comprises 360 audio excerpts, each approximately 30 seconds long, captured from six professional guitarists performing identical sets of 30 lead sheets in two modes—comping and soloing—resulting in a diverse collection of playing styles and textures. The 30 lead sheets themselves span five musical styles (Rock, Singer–Songwriter, Bossa Nova, Jazz, Funk), three chord progressions (12-bar Blues, Autumn Leaves, Pachelbel's Canon), and two tempi (slow, fast), with each excerpt's key sampled uniformly at random to maximize tonal variety. Audio was captured simultaneously via a hexaphonic pickup—providing separate signals for each string, yielding three versions per excerpt: the raw six-channel "hex" recording, an interference-reduced "hex_cln" version, and a mono "mic" mix for reference. See table 6.1 for a breakdown of the attributes of three files "00_BN1-129-Eb_comp_mic.wav" (1), "02_Jazz3-150-C_comp_mic.wav" (2), and "00_Rock2-85-F_solo_mic.wav" (3).

File	Style	Tempo (BPM)	Key	Mode
1	Bossa Nova 1	129	Eb	Comp
2	150	C	Comping	
3	Rock 2	85	F	Solo

Table 1: Track specifications for three randomly selected GuitarSet excerpts.

Each excerpt is accompanied by a JAMS file encoding 16 annotation tiers, including six per-string pitch contours, six MIDI note tracks, chord labels (both instructed and performed), beat and downbeat positions, tempo, and key information, all time-aligned to the audio stream. By combining automated hexaphonic signal separation with JAMS' JSON-based schema, GuitarSet provides a scalable annotation pipeline that dramatically reduces manual effort while ensuring high alignment accuracy between audio and symbolic data. Because of the multimodality, and open-source nature of this resource, GuitarSet has become a benchmark for models of guitar transcription and analysis, underpinning studies in note-level transcription, chord recognition, and performance modeling. The flattened note distribution over the 21 quantized fret-classes is highly imbalanced (see counts above). The "no-note" class (Class 20) overwhelmingly dominates, accounting for 71.71% of all note-slots, reflecting the fact that most time-bins contain no active note. Among the remaining 20 fret classes, Class 6 (4.07%), Class 4 (3.32%), and Class 3 (3.19%) are the most frequent, while mid-range pitches (e.g. Classes 0–2 and 5–9) each occupy roughly 1.2–2.8%. The tail classes (Classes 11–19) are extremely rare (together <1%), with some (Classes 16–19) below 0.01%.

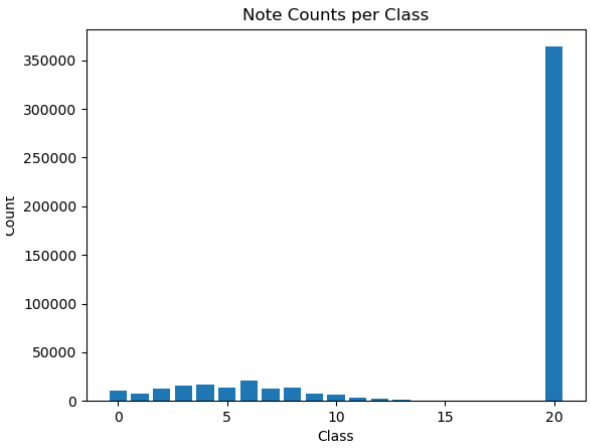

Figure 11: Class distribution histogram of GuitarSet

6.2 Data preparation

The JAMS files are first converted into MIDI format using the tab-estimator conversion toolkit[KHT22], and the resulting MIDI sequences are then aligned with their corresponding WAV files to produce unified NPZ archives. Each NPZ file contains

time–frequency representations (e.g., CQT, Mel spectrograms) alongside quantized tablature and onset data. During this process, MIDI tracks are automatically quantized to the nearest semitone grid to ensure consistent note timing and improve transcription accuracy. Finally, each NPZ is subdivided into bar-level segments, yielding a larger, bar-aligned dataset that facilitates training on manageable sequence lengths.

6.3 Datawrapper

To facilitate architectural tweaks and training, we developed a custom `QuantizedNpzDataset` wrapper and accompanying DataLoader. The methods of the datawrapper were adapted from Tab-Estimator to work with a Transformer[VSP+23] architecture. The dataset wrapper loads the instances, which come in the form of NPZ files, collections of NumPy arrays that can be accessed as a dictionary, stacks selected feature channels into tensors, and tokenizes quantized tablature into integer sequences. A tailored collate function pads input feature maps and target sequences to the same length within each batch, producing masks for teacher-forcing and attention. This modular design cleanly separates data preparation from model logic and supports optional transforms for further augmentation or normalization. The datawrapper allows for seeding, and like Tab-Estimator[KHT22], it uses `random.shuffle()`, meaning the splits are easily reproducible and comparable.

6.4 Datasplits

Using a custom dataset splitter the full `QuantizedNpzDataset` is partitioned into training, validation, and test subsets using a reproducible random split. First, the configuration file is loaded to obtain the NPZ directory and DataLoader parameters. The full dataset is instantiated and its length N determined. Given fractional splits $(\alpha_{\text{train}} = 0.8, \alpha_{\text{val}} = 0.1, \alpha_{\text{test}} = 0.1)$ summing to 1, the integer subset sizes are computed. A fixed random seed ensures the splits are deterministic across runs. Next, PyTorch's `random_split` utility allocates dataset indices into the three subsets. Each resulting subset is wrapped in a `DataLoader` that applies the same batch size, worker count, and padding collate function used in training. The training loader is configured with shuffling enabled to introduce stochasticity in mini-batch selection, while the validation and test loaders have shuffling disabled to ensure consistent evaluation. Pinned memory is enabled for faster host-to-device transfers. This modular splitting and loader construction supports easy experimentation with different split ratios, batch settings, and ensures that model selection and final reporting are based on held-out data.

7 Experiments

To assess NoodleGP's transcription performance, we conduct a series of controlled experiments. First, we establish our baseline by evaluating leading guitar-tab transcription models on GuitarSet. Next, we detail the experimental training setup, including optimizer and scheduling choices. Finally, we describe our grid-search procedure for hyperparameter tuning and summarize its outcomes.

7.1 Baseline

To contextualize the experimental results, the leading automatic tablature transcription models evaluated on GuitarSet are chosen as the baseline. This set comprises the CNN-only approaches TabCNN [WK19] and FretNet [CHK23], together with the hybrid, Conformer-inspired Tab-Estimator [KHT22]. Including all three models ensures coverage of purely convolutional architectures as well as those integrating self-attention mechanisms.

Particular emphasis is placed on Tab-Estimator due to its fully public, well-documented codebase and straightforward execution pipeline, which enable local verification of baseline results. To guarantee a fair comparison, the original six-fold cross-validation protocol from GuitarSet is adopted: in each fold, 80% of excerpts serve as training data, 10% as validation for hyperparameter tuning, and the remaining 10% as test data.

Applying this identical six-fold scheme to both the baseline models and the proposed Transformer–CNN hybrid ensures that evaluation metrics—such as frame-level F1, note-level F1, and

playability rate—are directly comparable. Any observed performance differences can thus be attributed solely to architectural and training strategy innovations rather than to variations in dataset splits or evaluation procedures.

7.2 Experimental Training Setup

The training loop's parameter are mainly controlled from a dedicated configuration files, but command-line arguments are parsed using Python's `argparse` module to allow run-time overrides of hyperparameters (e.g., epochs, learning rate, batch size, etc.). Model parameters requiring updates are optimized with the `AdamW` algorithm, which decouples weight decay from gradient updates to improve the convergence stability. To increase stability further, a composite learning-rate schedule was implemented. An initial linear warm-up phase using `LinearLR` gradually increases the learning rate over a fixed number of iterations, followed by `CosineAnnealingWarmRestarts` to periodically anneal and restart the rate, encapsulated within SequentialLR for smooth transitions. During each training batch, the model computes token logits that are compared to ground-truth sequences via `CrossEntropyLoss` (with padding indices ignored) to produce a scalar loss. Gradients are back-propagated and then clipped using `clip_grad_norm_` to a maximum norm of 1.0, preventing exploding gradients and ensuring stable updates. After each optimizer step, the learning-rate scheduler is advanced to adjust the rate dynamically according to the defined schedule. At the end of every epoch, a validation pass is run with gradients disabled (via `torch.no_grad()`) to compute the validation loss on unseen data, facilitating efficient memory use and faster evaluation.

7.3 Grid Search for Hyperparameter Optimization

To identify the best combination of training settings for the model, an exhaustive 3-fold validated grid search is performed over five key hyperparameters, with a fixed epoch count and early stopping for efficiency purposes. The splits for training/validation/test were 0.8, 0.1 and 0.1 respectively. All permutations of the variables are tested on the same three folds, meaning each hyperparameter combination sees the exact same training and test-sets. To achieve this, three seeds (42, 43, 44), chosen at random, are set to be the folds' seeds. All values in the grid were selected based on preliminary manual testing, after which only remotely viable options were retained. The epoch count was chosen based on earlier simpler grid searches, in which no run experienced any benefits beyond this range on the testing (read shallow) depth of the model. The class-weighting flag refers to an integer flag that selects the weighting for the cross entropy loss, 0 denoting no weights at all, 1 meaning inverse weighting based on class instance count giving higher weights to classes that are less represented in the training set, 2 meaning inverse weighting with clamped values. For the clamped values, manual testing was done over a longer period of development. Due to time constraints, these values are not included in the grid search, and are fixed to (0.1, 600). Due to the imbalance of the dataset, the weights seem to provide some speedup in generalization, though end up converging to around the same F1 values.

- **Number of epochs**: {250}
- **Learning rate**: {1e-5, 1e-4, 1e-3}
- **Batch size**: {16, 32, 64}
- **Weight decay**: {0, 1e-3, 1e-2}
- **Dropout rate**: {0, 0.1}
- **Class-weighting flag**: {0, 1, 2}

This yields a total of 162 unique configurations. Each run is executed in its own directory (`run_0`, ..., `run_107`) and the exact command line is logged to a file named according to its hyperparameter values (e.g. `e250_lr0.001_bs32_wd0.01_do0.1_cew1.txt`).

To allow for efficient search, a shallower architecture compared to the full-scale model is employed during the grid-search, as well as the specific ResNet-module[HZRS15] being kept at the shallowest available model, in order to accommodate hardware constraints. This combined reduced depth lowers

memory footprint and speeds up each epoch, allowing the script to evaluate more configurations within a reasonable time frame. After training, each model is evaluated on the validation set using the precision, recall, F-measure and TDR. Models are ranked primarily by the summed F1 scores of note and frame and TDR to balance overall performance and discovery reliability. The hyperparameter set that maximizes these metrics while respecting the computational constraints is selected for final evaluation and further experimentation.

7.4 Grid Search Results

To evaluate the results of the 3-fold cross-validated grid search, we summed and averaged the summed F_1 scores across folds, then selected the model with the highest mean summed F_1. The best configuration corresponded to Run 59:

- `Learning Rate` = 1e−4
- `Batch Size` = 16
- `Weight Decay` = 0
- `Dropout` = 0.1
- `Class-weighting flag` = 1 (Class-count based inverted weights)

This configuration was then used for the training of the final model.

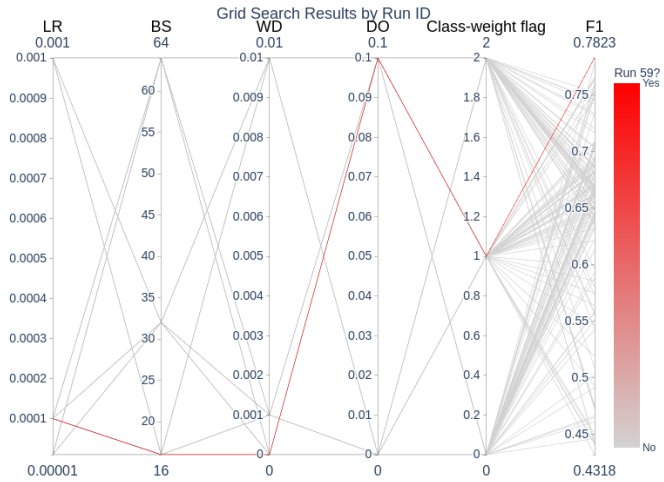

Figure 12: Parallel-coordinates plot where each line denotes a configuration. Each intersection with a vertical axis denotes the specific value of that variable. The line in red is Run 59.

8 Baseline Comparison

The metrics used for the baseline comparison were chosen, as they have been widely used in related research and the baseline methods, the metrics for the evaluation are the ones as defined in section 3, Fundamentals, as well as performing a 6-fold cross validation on the entire dataset. For comparability the same type of 6-fold cross validation was performed as was done in both TabCNN[WK19] and Tab-Estimator[KHT22], namely holding out 1 of the 6 guitarists featured on GuitarSet for testing, using the other 5 for training, with a training/validation ratio of 0.9. For the number of epochs, 250 was still used, as a compromise between allowing time for generalization, and efficiency. The final model is trained on a transformer-depth of 1024, a feed-forward depth of 4096, 2 encoder layers, and 6 decoder layers. The depth values were chosen as this was the maximum transformer-depth that was possible to be used within the memory-constraints, the feed-forward depth was chosen to be 4 times as large as the transformer-depth, as this is recommended by "Attention Is All You Need"[VSP+23], the encoder and decoder layers were chosen as values lower tended to under-perform, yet values higher caused difficulties with the memory-constraints, and consid-

ering that the ResNet-CNN[HZRS15] functions as the encoder, the model architecture is built around a small encoder dimension.

8.1 Experimental Results

The models generated by the cross-validation were then ran over the entirety of the dataset, and the following table shows the results. The two tables are split up based on Frame and Note. The P, R, F_1 and TDR refer to the Precision, Recall, F_1 and Tablature Disambiguation rate as defined in section 3. The note-table contains only Tab-Estimator, as TabCNN[WK19] and FretNet[CHK23] only reported the frame-based scores. The numbers prefixed by ± represent the standard deviation of the scores, calculated over the 6 folds. NoodleGP refers to the model described in the implementation, whereas NoodleGP(s) refers to a smaller variant of the model, with the depth of the model being reduced to 256 from 1024, the dimension of the FFN being reduced to 1024 from 4096, the number of encoder layers being reduced to 1 from 2, and the number of decoder layers being reduced to 3 from 6. The s indicates the small variant, whereas the 34 indicates that ResNet34[HZRS15] was used instead of ResNet18.

	P	R	F_1	TDR
Tab-Est-note	**0.781** ±0.031	**0.777** ±0.039	**0.775** ±0.029	**0.919** ±0.021
NoodleGP-note	0.242 ±0.012	0.580 ±0.022	0.341 ±0.012	0.908 ±0.009
NoodleGP(s)-note	0.239 ±0.006	0.585 ±0.021	0.337 ±0.009	0.909 ±0.006
NoodleGP(34)-note	0.236 ±0.015	0.573 ±0.031	0.334 ±0.020	0.906 ±0.009
NoodleGP(34s)-note	0.227 ±0.011	0.573 ±0.027	0.325 ±0.014	0.906 ±0.014

Table 2: Note-based Precision, Recall, F_1 and TDR of TabCNN[WK19] and the proposed method NoodleGP.

	P	R	F_1	TDR
TabCNN-frame	**0.809** ±0.029	0.696 ±0.061	0.748 ±0.047	0.899 ±0.033
Tab-Est-frame	0.789 ±0.027	0.780 ±0.040	**0.781** ±0.029	0.918 ±0.020
FretNet-frame	0.801	0.669	0.727	Missing
NoodleGP-Frame	0.355 ±0.035	0.797 ±0.029	0.490 ±0.035	**0.937** ±0.007
NoodleGP(s)-frame	0.352 ±0.022	**0.813** ±0.031	0.491 ±0.020	0.936 ±0.008
NoodleGP(34)-frame	0.348 ±0.033	0.791 ±0.037	0.483 ±0.037	0.933 ±0.009
NoodleGP(34s)-frame	0.324 ±0.029	0.782 ±0.045	0.458 ±0.036	0.935 ±0.011

Table 3: Frame-based Precision, Recall, F_1 and TDR of TabCNN[WK19], Tab-Estimator[KHT22], FretNet[CHK23], and the proposed method NoodleGP.

The model's precision is notably lower than that of the baselines, even though its recall exceeds theirs. In practical terms, this means the model is predicting more liberally, detecting more of the true fret positions, but also generating many incorrect predictions. As a result, its overall F_1 score falls below the baselines', driven down primarily by the drop in precision despite the gain in recall. It can be seen that the TDR metric lies substantially higher than the precision, recall and F_1. This is because TDR measures the amount of correct string-fret combinations, only for correctly identified pitches. When the model does make a correct prediction, it captures the exact string–fret combination most the time. In other words, although correct predictions are relatively rare, most of those predictions pinpoint the true string and fret simultaneously. Another reason is that string-fret combinations are heavily skewed to certain frets (see Figure 11), making the task easier for the models.

It can also be seen that the model's performance is higher for frame-level than note-level. For note-level the model sees and attempts to identify a single note-event, whereas with frame level it get shown a small snippet of a larger piece. This implies the model has learnt the relations between the notes more than the individual pitches played. Tab-Estimator's note-level scores align with their frame-level scores, having minimal differences on the metrics. This aligns with the initial theory of

the transformer model being better at the temporal relations between the notes, than at identifying the individual notes, at which the CNNs excel.

The smaller model's recall lies higher than the larger model's, and the precision and TDR are lower by only a very small amount, implying the smaller size better aligns with the size of the dataset, reducing overfitting. The ResNet34 encoder also doesn't seem to improve performance on the dataset, providing diminishing returns. Larger ResNet-encoders couldn't be tested due to time and memory-constraints, and as such are still a possible study.

In Figure 13 a heatmap is depicted showing the predictions compared to the ground-truth, where the y-axis represents a string-fret combination of the ground-truth, and the x-axis represents the prediction the model made for this specific pair. The diagonal represents correct predictions, and anything off of the diagonal represents an incorrect prediction. In Table 4 the most common mistakes are shown, with the number of times the model made this specific mistake shown in the count column. The data shows that most of the mistakes made are by a bias where the model predicts the open fret of the correct string shifted up by 1. This consistent miss prediction could be a result of bias in the dataset, as a result of the small size.

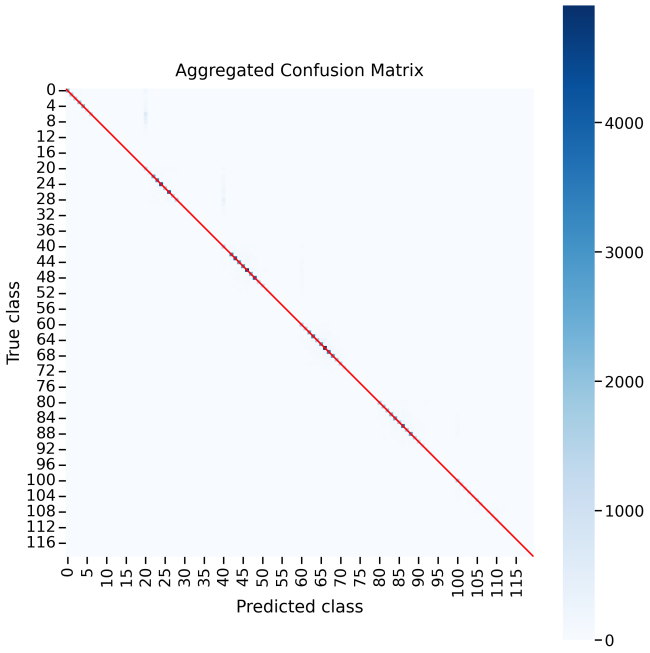

Figure 13: Heatmap of paired mistakes

Rank	$(string, fret)_{gt}$	$(string, fret)_{pred}$	**Count**
1	string 0, fret 6	string 1, fret 0	639
2	string 0, fret 7	string 1, fret 0	429
3	string 1, fret 8	string 2, fret 0	390
4	string 0, fret 8	string 1, fret 0	357
5	string 3, fret 4	string 3, fret 5	300
6	string 3, fret 4	string 3, fret 6	288
7	string 4, fret 5	string 4, fret 6	259
8	string 1, fret 6	string 2, fret 0	248
9	string 0, fret 5	string 1, fret 0	218
10	string 1, fret 9	string 2, fret 0	216
11	string 1, fret 5	string 2, fret 0	209
12	string 3, fret 4	string 3, fret 3	203
13	string 0, fret 0	string 1, fret 0	178
14	string 3, fret 6	string 3, fret 7	178
15	string 3, fret 8	string 3, fret 7	173

Table 4: Paired ground-truth - predicted, with counts of the number of times the mistake was made.

To visualize the way differences in string and fret distances between the ground-truth and the predictions, two histograms are featured below. Figure 14 shows the distance of strings between ground-truth and predicted, whereas Figure 15 shows the distance between ground-truth fret location and predicted. Most of the mistakes made were in the selection of the fret, as can be seen in Figure 14, as

the most common distance is 0, and 1 being the only other distance. The errors made in fret selection are more diverse, with most mistakes being off by little, larger distances trailing off to a tail. Fret use is very varied, being split over 21 different classes, and though imbalanced towards open predictions, the inverse weights cause this to be optimized to predict all classes more equally. The string class has less variance, as well as no direct weighting, and is as such more sensitive to imbalance.

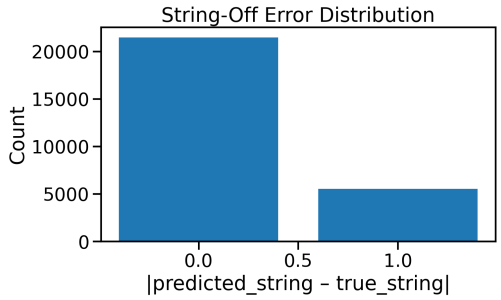

Figure 14: Histogram of string prediction distances.

The distance between ground-truth and predicted for frets is more diverse

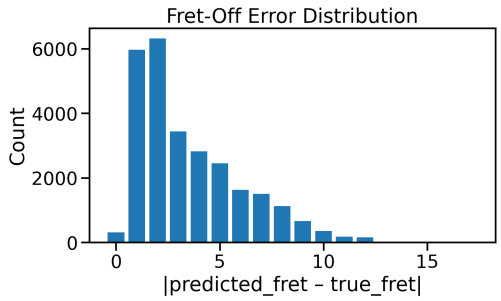

Figure 15: Histogram of fret prediction distances.

9 Conclusions and Further Research

In this work, we proposed NoodleGP, a hybrid ResNet–Transformer architecture for automatic guitar tablature transcription. By combining a pretrained ResNet-18 backbone with multi-head self-attention layers, NoodleGP leverages both local spectral–temporal features and global sequence dependencies. On the GuitarSet benchmark, our proposed model achieved relatively low frame-level and note-level F_1 scores (0.341 and 0.490, respectively) and demonstrated a Tablature Disambiguation Rate of nearly 94% at the frame level, indicating that when pitches are correctly identified, the model still resolves string-fret assignments almost all the time. These results, though substantially below state-of-the-art CNN-only and self-attention-augmented baselines, show that CNN-Transformers have the ability to disambiguate tablature effectively.

9.1 Contributions

The results confirm that the hybrid approach can learn meaningful fingering patterns under limited memory conditions. Though only beating the baseline in TDR and recall, some contributions were made.

- Open-Source Hybrid Architecture: A complete pipeline combining a ResNet-18 Encoder, with Positional Encoding and a dual-headed Transformer.
- Targeted Tuning: We performed a focused 162-run grid search to pinpoint optimal hyperparameters (LR = 1e-4, BS = 16, Dropout = 0.1, class-weighting), maximizing note-level F and TDR under compute constraints.
- Comparative evaluation: A conceptual and result-based comparison between the hybrid architecture and related researches.
- Error-Driven Insights: Identified systematic biases—open-string overprediction and one-string shifts—via heatmaps and distance histograms, guiding precise next-step improvements.

9.2 Future work

While NoodleGP demonstrates the promise of a ResNet–Transformer hybrid for guitar tablature transcription, several avenues could be explored to further enhance both accuracy and generality:

- **Explore deeper CNN backbones.** Consider evaluating larger ResNet variants (ResNet-50, or bottleneck-based ResNets up to ResNet-152) to capture richer spectral–temporal patterns and improve residual feature learning.

- **Explore Other CNN backbones.** Consider evaluating the performance of other more modern pre-trained models such as ConvNeXt[LMW+22].
- **Adopt longer-dependency Transformer variants.** It may be beneficial to experiment with Transformer-XL[DYY+19] like Groove Modeling[CHHY20] does, or similar architectures that support segment-level recurrence, allowing attention over extended musical phrases without prohibitive memory growth.
- **Leverage larger or more diverse data sources.** One could pretrain on large pseudo-labeled corpora (e.g., SynthTab[ZZCD24] or DadaGP[SKC+21]) and fine-tune on high-quality human-annotated sets, or integrate additional guitar datasets (GAPS[RGED24], EGDB[CHH+22], AnimeTAB[ZJX22]) to improve generalization for deeper models.
- **Develop novel augmentation strategies.** Beyond pitch-shifts and time-stretches, applying SpecAugment, mixup of different guitar tracks, or physically informed audio modifications (e.g., simulating string damping or alternate pickup EQ profiles) could enrich training diversity.

References

[BF13] Gregory Burlet and Ichiro Fujinaga. Robotaba Guitar Tablature Transcription Framework. In *Proceedings of the International Society for Music Information Retrieval Conference*. International Society for Music Information Retrieval, Nov 2013.

[BKTB12] Ana Barbancho, Anssi Klapuri, Lorenzo Tardon, and Isabel Barbancho. Automatic transcription of guitar chords and fingering from audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 20:915–921, Mar 2012.

[BTSB12] Isabel Barbancho, Lorenzo Tardon, Simone Sammartino, and Ana Barbancho. Inharmonicity-based method for the automatic generation of guitar tablature. *IEEE Transactions on Audio, Speech, and Language Processing,*, 20:1857–1868, Aug 2012.

[CHH+22] Yu-Hua Chen, Wen-Yi Hsiao, Tsu-Kuang Hsieh, Jyh-Shing Roger Jang, and Yi-Hsuan Yang. towards automatic transcription of polyphonic electric guitar music:a new dataset and a multi-loss transformer model, Feb 2022.

[CHHY20] Yu-Hua Chen, Yu-Hsiang Huang, Wen-Yi Hsiao, and Yi-Hsuan Yang. Automatic composition of guitar tabs by transformers and groove modeling, Aug 2020.

[CHK23] Frank Cwitkowitz, Toni Hirvonen, and Anssi Klapuri. Fretnet: Continuous-valued pitch contour streaming for polyphonic guitar tablature transcription. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, page 1–5. IEEE, June 2023.

[DYY+19] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, Jan 2019.

[GQC+20] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition, May 2020.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, Dec 2015.

[KHT22] Sehun Kim, Tomoki Hayashi, and Tomoki Toda. Note-level automatic guitar transcription using attention mechanism. In *2022 30th European Signal Processing Conference (EUSIPCO)*, pages 229–233, Aug 2022.

[KME23] Christian Kehling, Andreas Männchen, and Andreas Eppler. Idmt-smt-guitar dataset, jan 2023.

[LMW+22] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s, Jan 2022.

[PE06] Graham E. Poliner and Daniel P. W. Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, 2007(1):048317, Dec 2006.

[RED24] Xavier Riley, Drew Edwards, and Simon Dixon. High resolution guitar transcription via domain adaptation, Feb 2024.

[RGED24] Xavier Riley, Zixun Guo, Drew Edwards, and Simon Dixon. Gaps: A large and diverse classical guitar dataset and benchmark transcription model, Aug 2024.

[SK10] Christian Schörkhuber and Anssi Klapuri. Constant-q transform toolbox for music processing. *Proc. 7th Sound and Music Computing Conf.*, 01 2010.

[SKC+21] Pedro Sarmento, Adarsh Kumar, CJ Carr, Zack Zukowski, Mathieu Barthet, and Yi-Hsuan Yang. Dadagp: A dataset of tokenized guitarpro songs for sequence models. *CoRR*, abs/2107.14653, Jul 2021.

[VSP+23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, Jun 2023.

[WK19] Andrew Wiggins and Youngmoo E. Kim. Guitar tablature estimation with a convolutional neural network. In *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, pages 284–291, Nov 2019.

[XBP+18] Qingyang Xi, Rachel M. Bittner, Johan Pauwels, Xuzhou Ye, and Juan P. Bello. Guitarset: A dataset for guitar transcription. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 453–460, September 2018.

[YSN+13] Kazuki Yazawa, Daichi Sakaue, Kohei Nagira, Katsutoshi Itoyama, and Hiroshi Okuno. Audio-based guitar tablature transcription using multipitch analysis and playability constraints. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 196–200, 10 2013.

[ZJX22] Yuecheng Zhou, Yaolong Ju, and Lingyun Xie. Animetab: A new guitar tablature dataset of anime and game music, Oct 2022.

[ZZCD24] Yongyi Zang, Yi Zhong, Frank Cwitkowitz, and Zhiyao Duan. Synthtab: Leveraging synthesized data for guitar tablature transcription. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, page 1286–1290, apr 2024.