Application for Transcribing Grand Piano Recordings to Sheet Music in PDF

Subject: Computer Engineering Project

Supervisor: Prof. Mariagrazia Fugini

Author: Michał Jagoda

June 3, 2024

Contents

1 Motivation of the Project

The motivation behind this project includes:

- **Problem-Solving:** Addressing the challenge of accurately transcribing complex piano music.
- **Innovation:** Leveraging AI to create new tools for musicians and educators.
- **Accessibility:** Making high-quality music transcription more accessible and convenient.
- **Personal Interest:** A passion for music and technology, and the desire to combine these fields in a meaningful way.

2 Topic Analysis

2.1 Overview

This project involves creating a desktop application designed to transcribe grand piano recordings into sheet music and export the results as a PDF. This involves the use of audio processing, music transcription, and PDF generation technologies, all performed locally without the need for an internet connection.

2.2 Objectives

- To accurately transcribe piano recordings into readable sheet music.
- To provide a user-friendly interface for uploading audio files and exporting sheet music.
- To offer an offline and local application for accessibility and privacy reasons.

2.3 Key Components

- **Audio Processing:** Capturing and processing the audio signals from the grand piano recordings using the `audio_processor.py`.
- **AI Model:** Generating an initial transcription output from the uploaded MP3 file using the `transcription_worker.py`.
- **Postprocessing Algorithm:** Creating a MIDI file from the AI model's output using the `Sheet_music_generation.py`.
- **PDF Generation:** Using the MuseScore API to generate a PDF from the generated MIDI file with `generate_pdf.py`.
- **User Interface:** Allowing users to interact with the application, upload audio files, view PDF previews, listen to MIDI and MP3 files, and download the resulting sheet music and MIDI files via `window.py`.

3 Technologies Used

- **Python:** The primary programming language used for developing the application.

- **Librosa:** A Python library for audio and music analysis, used for audio processing and feature extraction in `audio_processor.py`.
- **Piano Transcription Inference:** A library for transcribing piano music, used in `transcription_worker.py`.
- **Music21:** A toolkit for computer-aided musicology, used in `Sheet_music_generation.py` to create and manipulate music streams.
- **MuseScore:** A software for creating, playing, and printing sheet music, used in `generate_pdf.py` to convert MIDI files to PDF.
- **PySide6:** A set of Python bindings for Qt libraries, used for creating the graphical user interface in `window.py`.
- **VLC:** A media player used for playing audio and MIDI files within the application.
- **Pdf2image:** A library for converting PDF files to images, used in the GUI to display PDF previews.
- **Tempfile:** A Python module used for creating temporary files, ensuring that the application handles file operations efficiently and securely.
- **WSL2 (Windows Subsystem for Linux):** Used to run some Python libraries that were not available on Windows. As WSL2 lacks support for audio and GUI components, server bridges were implemented to facilitate communication between the audio processing and GUI components running in different environments.

4 Functional Requirements

4.1 Core Functionalities

- **Audio File Upload:**
  - Users can upload MP3 audio files.
- **AI Model:**
  - The system processes the uploaded audio using an AI model to generate an initial transcription.
- **Postprocessing Algorithm:**
  - The application converts the AI model's output into a MIDI file.
- **PDF Generation:**
  - The MuseScore API is used to convert the generated MIDI file into a PDF document.
- **PDF Preview and Download:**
  - Users can preview the PDF of the sheet music.
  - Users can download the PDF and the MIDI file.
- **Playback Feature:**

- Users can listen to the transcription by playing the MIDI file.
- Users can also listen to the original MP3 file.

5 Non-Functional Requirements

5.1 Performance Requirements

- **Speed:** The transcription process should be completed within a reasonable time frame (under 15 minutes for a 5-minute recording).
- **Accuracy:** The transcription should accurately reflect the notes, rhythms, and dynamics of the original recording with a high degree of precision (over 90% accuracy).

5.2 Usability Requirements

- **User Interface:** The interface should be intuitive and easy to navigate, even for users with limited technical skills.

5.3 Compatibility Requirements

- **File Format Support:** The application should support common audio file formats and produce PDFs that are compatible with standard PDF readers.

5.4 Maintainability Requirements

- **Modularity:** The application code should be modular to facilitate easy updates and maintenance.
- **Documentation:** Comprehensive documentation should be provided for developers to understand the codebase and contribute to its development.

5.5 Scalability Requirements

- **Future Expansion:** The architecture should allow for easy addition of new features and functionalities.

6 Use Case Diagram

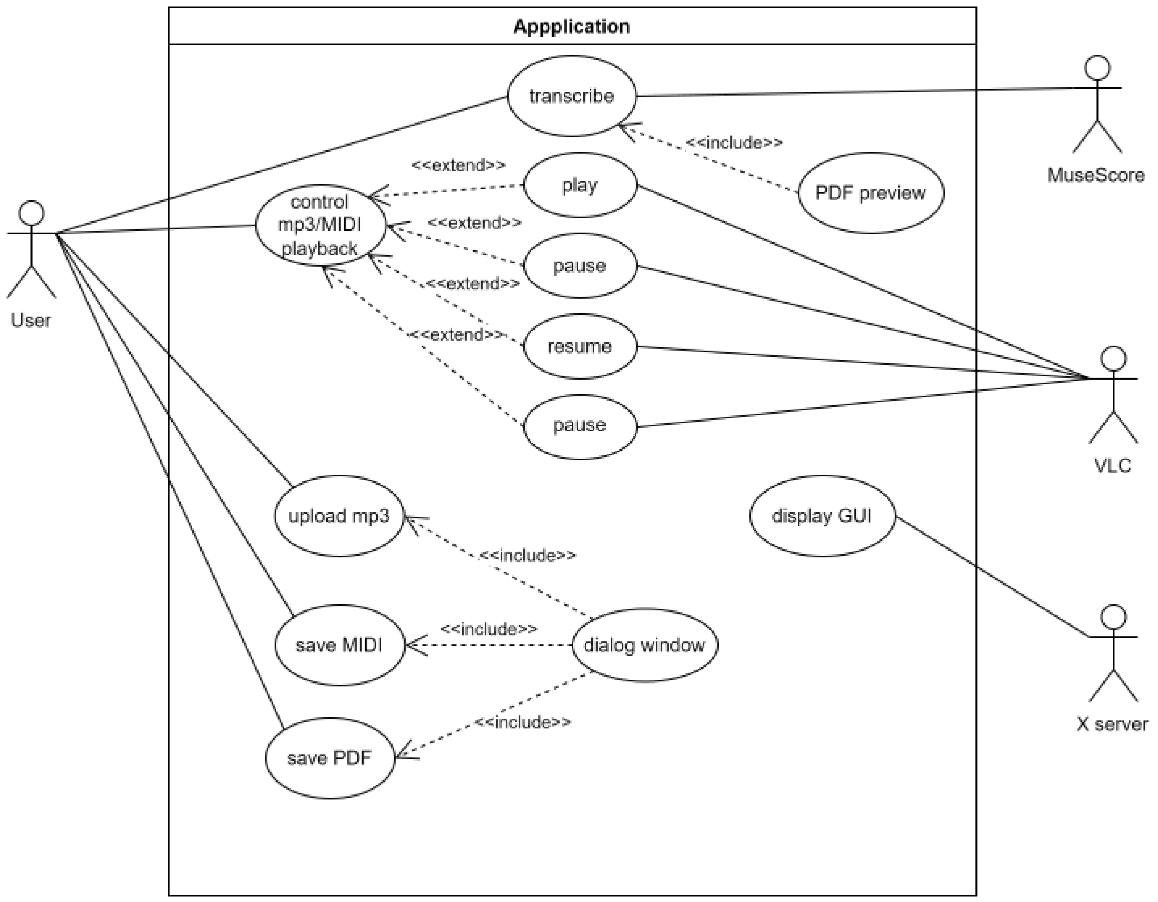


Figure 1: Use Case Diagram

7 System Architecture Flow Chart

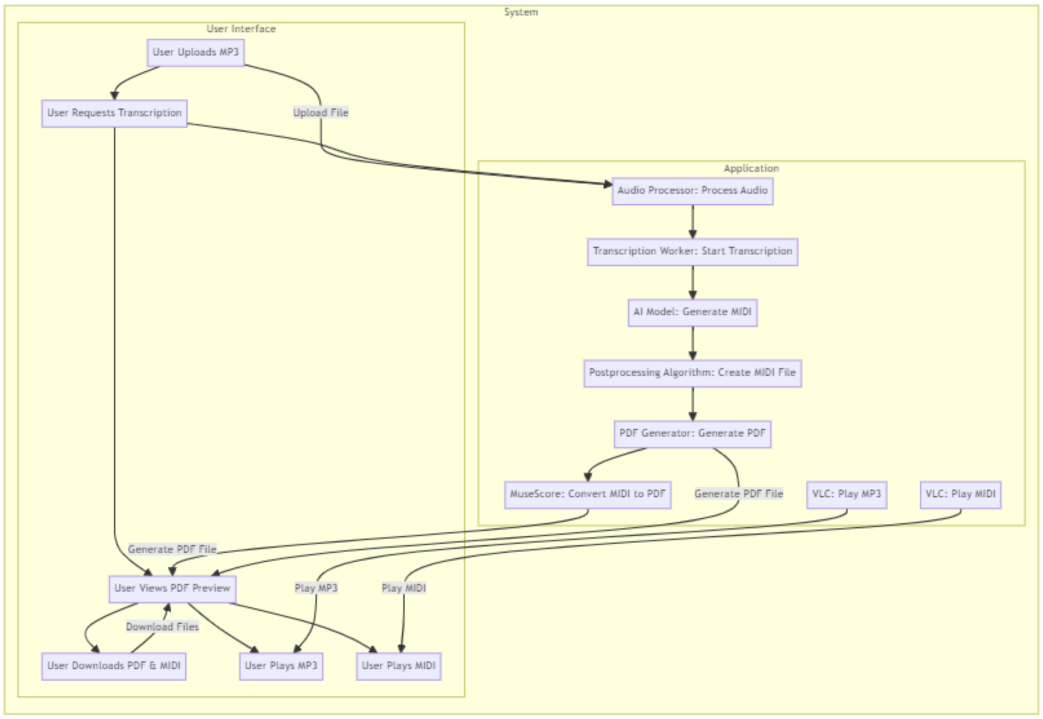


Figure 2: System Architecture Flow Chart

8 Class Diagram

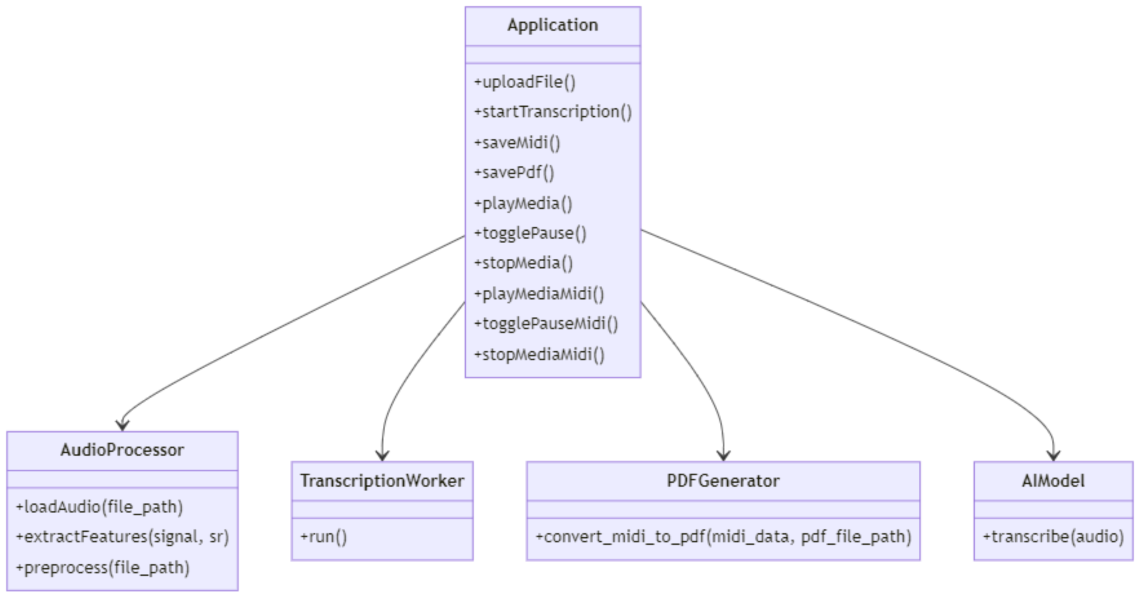


Figure 3: Class Diagram

9 Sequence Diagrams

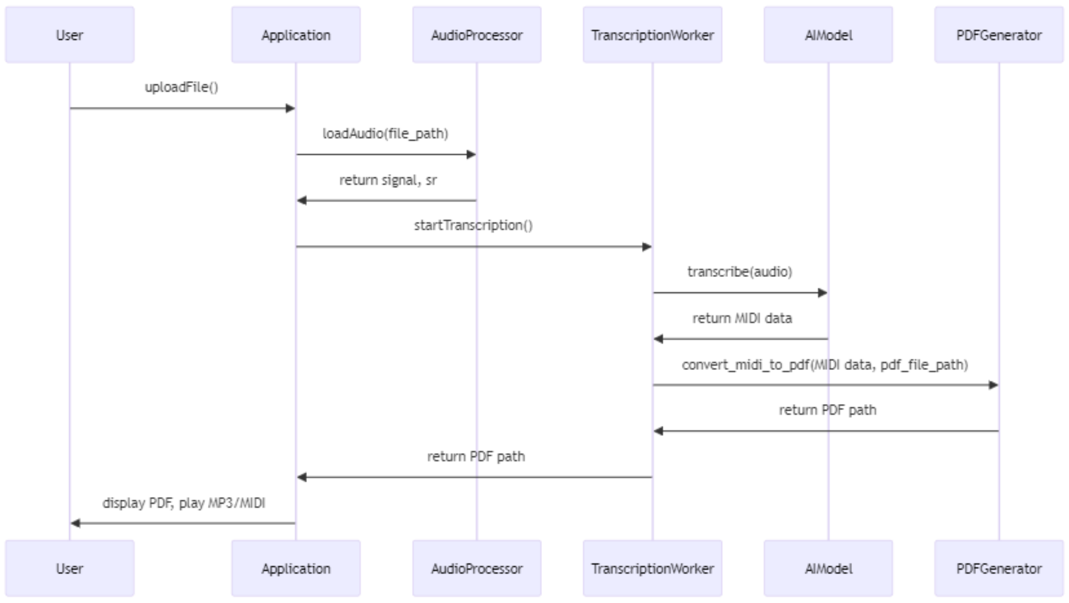


Figure 4: Transcription process sequence diagram

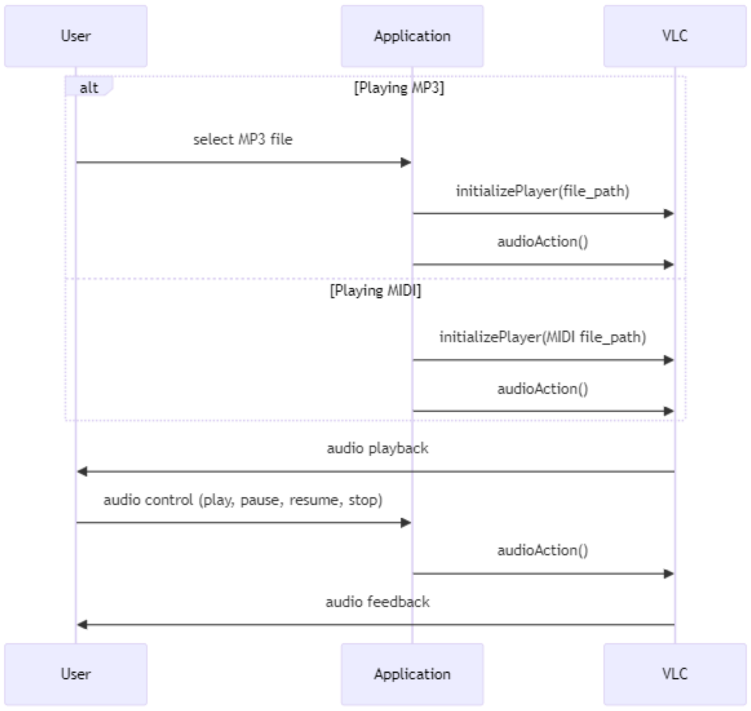


Figure 5: Audio sequence diagram

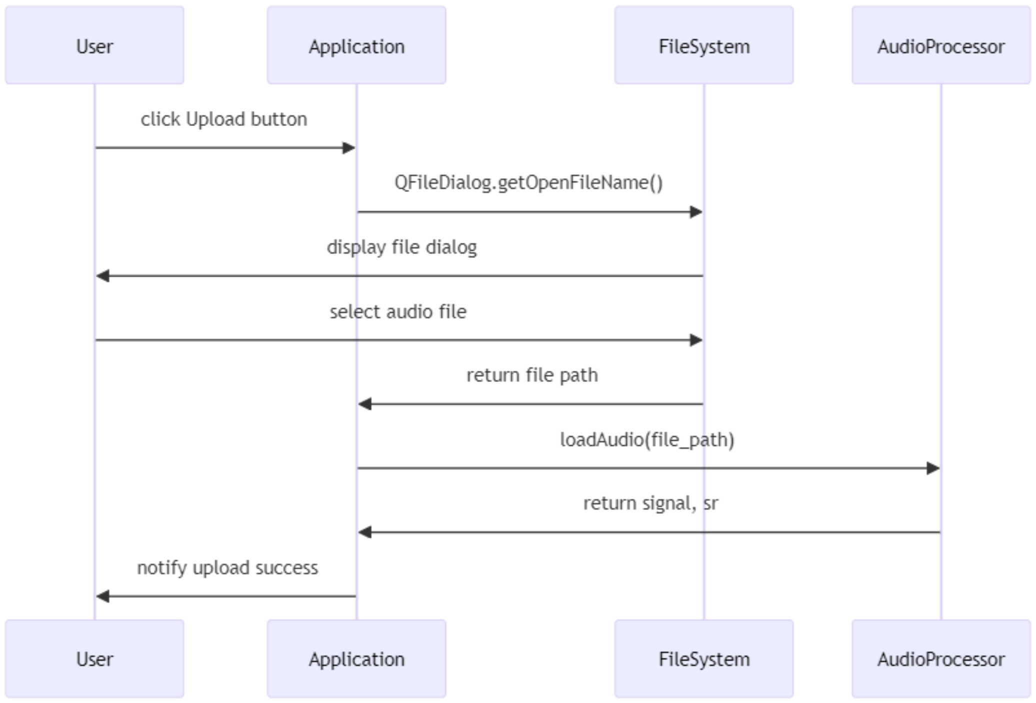


Figure 6: Load file sequence diagram

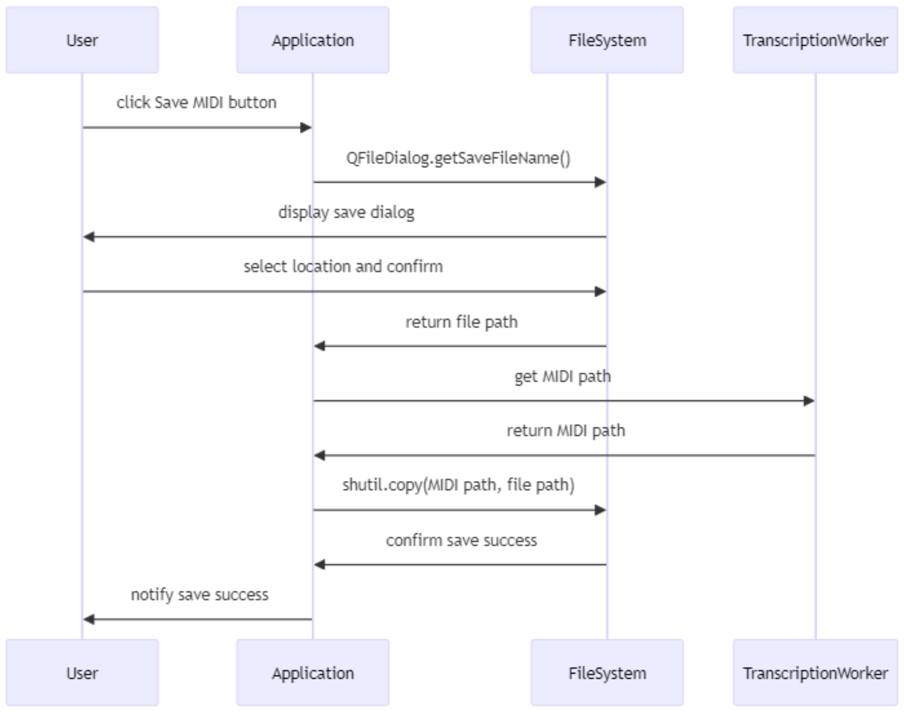


Figure 7: Save MIDI sequence diagram

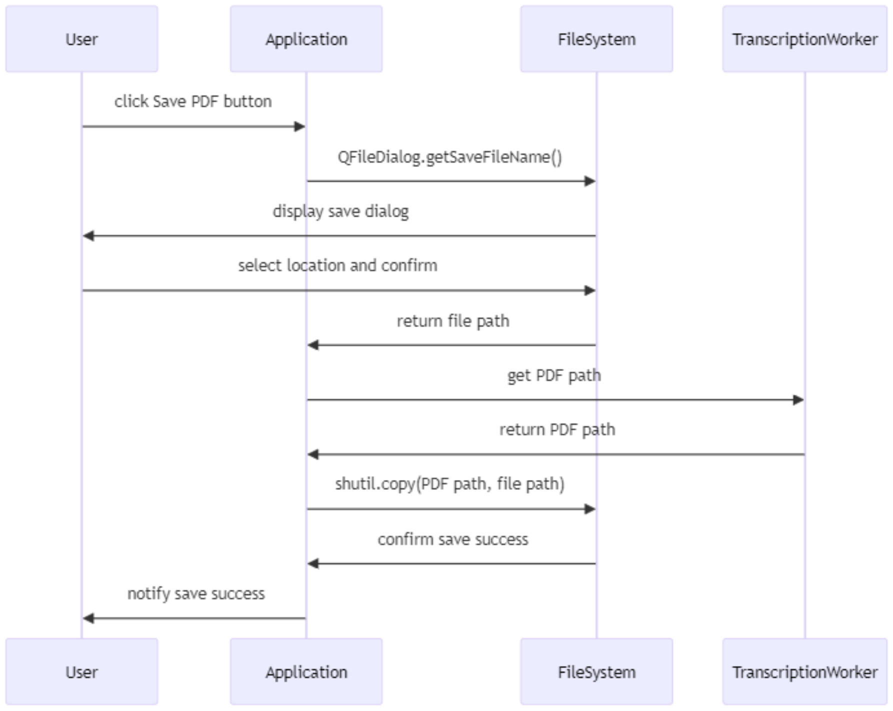


Figure 8: Save PDF sequence diagram

10 AI Model

The AI model implemented for this project is based on the architecture proposed by Kong et al. in "High-resolution Piano Transcription with Pedals by Regressing Precise Onsets and Offsets Times." This model is designed to achieve high accuracy in transcribing both notes and pedal actions from piano recordings.

10.1 Main Points of the AI Architecture

- **Input Features:** The input to the model is a log mel spectrogram of the audio recording, which provides a time-frequency representation of the sound.
- **Convolutional Layers:** These layers extract high-level features from the log mel spectrogram, capturing the essential information required for accurate transcription.
- **Bidirectional Gated Recurrent Units (biGRUs):** These layers model the temporal dependencies in the music, which are crucial for identifying the precise timing of notes and pedals.
- **Regression and Classification Outputs:** The model predicts continuous onset and offset times, velocities for notes, and the presence of notes and pedals in each frame.

10.2 AI Model Architecture

See Figure 9 for the architecture of the AI model.

10.3 Training Model

The training process for the AI model involves several key steps to ensure high accuracy and robustness:

- **Dataset:** The model was trained using the MAESTRO dataset, which contains high-quality piano recordings and corresponding MIDI files. This dataset is ideal due to its precision in time alignment.
- **Preprocessing:** Audio recordings were converted to mono and resampled to 16 kHz. Log mel spectrograms were then extracted with a Hanning window and mel filter banks.
- **Model Architecture:** The architecture includes convolutional blocks followed by biGRU layers. The convolutional layers help in extracting spatial features, while the biGRUs capture the temporal aspects of the music.
- **Loss Functions:** Multiple loss functions were used to train the model, including binary cross-entropy for frame-wise classification and regression losses for onset and offset predictions.
- **Training Procedure:** The model was trained using the Adam optimizer with a learning rate schedule. Dropout was used to prevent overfitting, and the model was trained for a large number of iterations to ensure convergence.

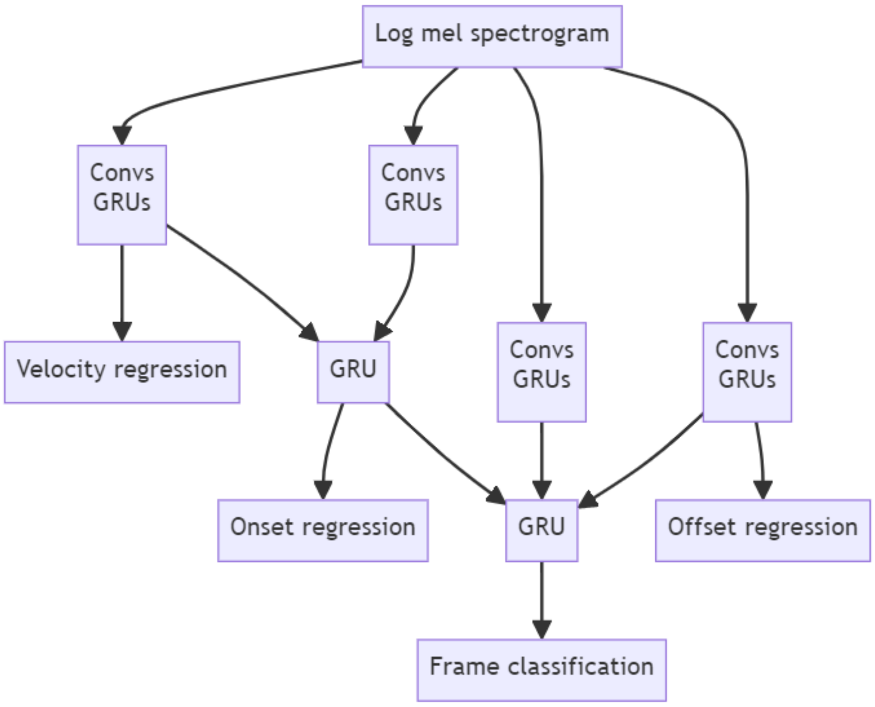


Figure 9: AI Model Architecture

- **Hardware:** Training was performed on Google Cloud Platform using NVIDIA Tesla V100 GPUs to accelerate the process, and to make it possible to train the model in a reasonable time frame.

10.4 Accuracy of AI

The model achieved high accuracy in transcribing piano recordings:

- **Onset F1 Score:** 96.72% on the MAESTRO dataset, outperforming previous state-of-the-art models.
- **Pedal Onset F1 Score:** 91.86%, marking the first benchmark result for pedal transcription on this dataset.
- **Robustness:** The model is robust to misalignments in the onset and offset labels, ensuring reliable performance even with noisy data.

11 Implementation and Usage

For the implementation and usage of the application, I encourage you to visit the GitHub repository: link to the repository. The repository contains:

- **README File:** Detailed instructions on how to set up and run the application.
- **Source Code:** All the necessary code files for the application, including:
  - **audio_processor.py:** Handles the preprocessing of audio recordings.
  - **transcription_worker.py:** Uses the AI model to generate initial transcriptions.
  - **Sheet_music_generation.py:** Post-processes the AI model's output to create MIDI files.
  - **generate_pdf.py:** Converts MIDI files into sheet music PDFs using the MuseScore API.
  - **window.py:** Provides the user interface for the application.
  - **requirements.txt:** Lists all the required Python libraries and their versions.
  - **AI Model:** Files related to model implementation, data preparation, training, evaluation, postprocessing, and inference.
- **Commented Code:** The code is well-commented to facilitate understanding and further development.
- **Sample Audio Files:** Included in the repository for testing the application (folder samples).

12 Motivations of Pieces Chosen

The pieces chosen for transcription were selected based on several factors:

- **Complexity:** Pieces like Chopin's Etudes and Beethoven's Sonatas offer a wide range of technical challenges, making them ideal for testing the transcription capabilities of the AI model.
- **Popularity:** These pieces are well-known and frequently studied, providing a good benchmark for comparison.
- **Variety:** Including different styles and periods helps ensure the model is versatile and robust, allowing it to handle various musical textures and structures.
- **Personal Experience:** I have personally played all of these pieces and am very familiar with them. This firsthand knowledge allows for a deeper understanding and more accurate evaluation of the transcription results.

13 Comparison of Generated and Original Sheet Music

The following sections provide a comparison of the generated sheet music with the original in terms of pitch. The generated sheet music is created by the AI model, while the original sheet music is the published version of the piece. Meaning of colors used in the generated sheet music:

- **Red:** wrong note (in terms of pitch).
- **Green:** correct melody (in terms of pitch).
- **Blue:** correct accompaniament (in terms of pitch).

13.1 F. Chopin - Etude op. 10 no. 5

13.1.1 Generated sheet music

Link to generated sheet music

13.1.2 Original sheet music

Link to original sheet music

13.2 L. van Beethoven - Sonata op. 53 no. 21 "Waldstein", Allegro con brio

13.2.1 Generated sheet music

Link to generated sheet music

13.2.2 Original sheet music

Link to original sheet music

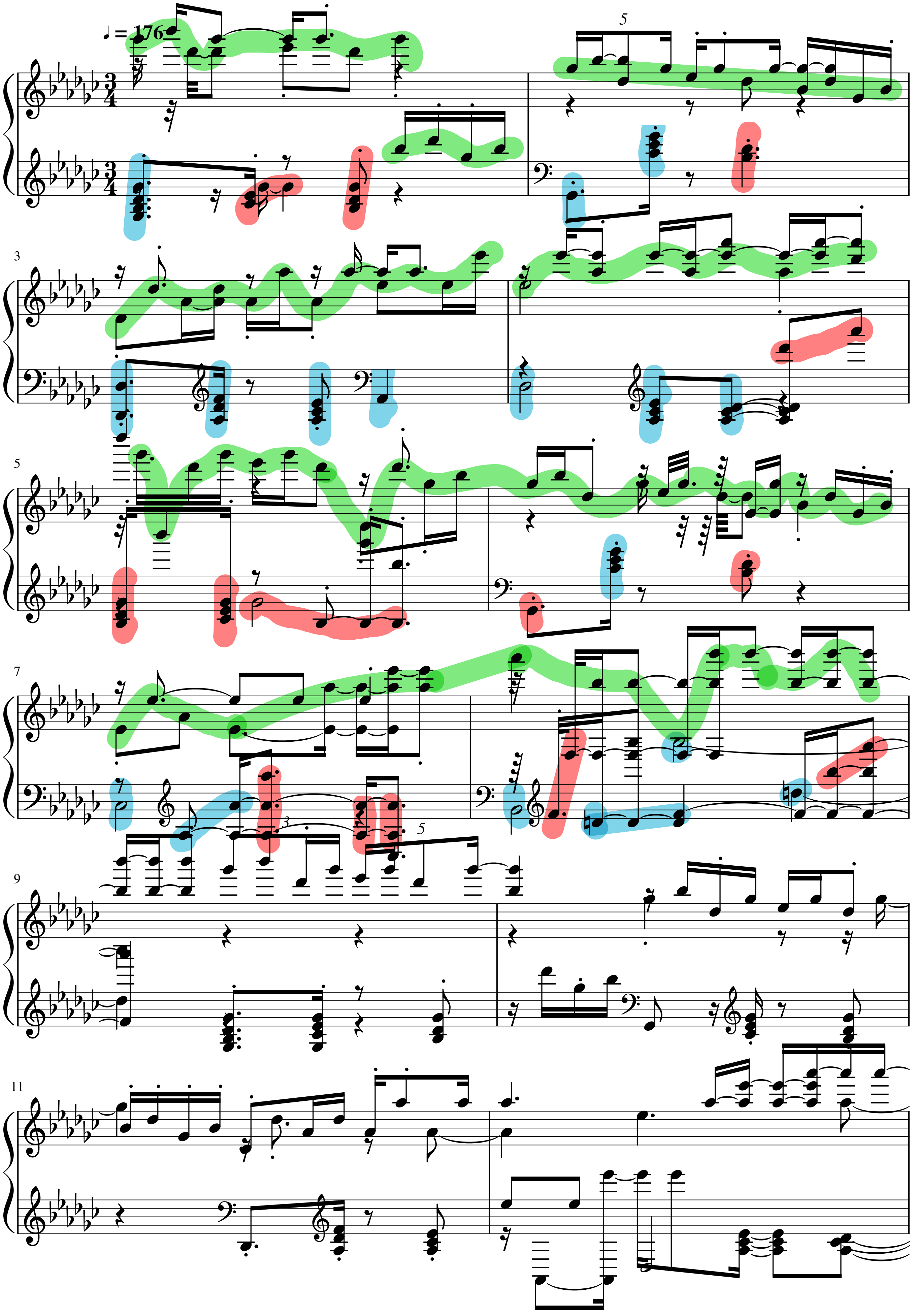
13.3 L. van Beethoven - Sonata op. 13 no. 8 "Pathetique", Adagio cantabile

13.3.1 Generated sheet music

Link to generated sheet music

13.3.2 Original sheet music

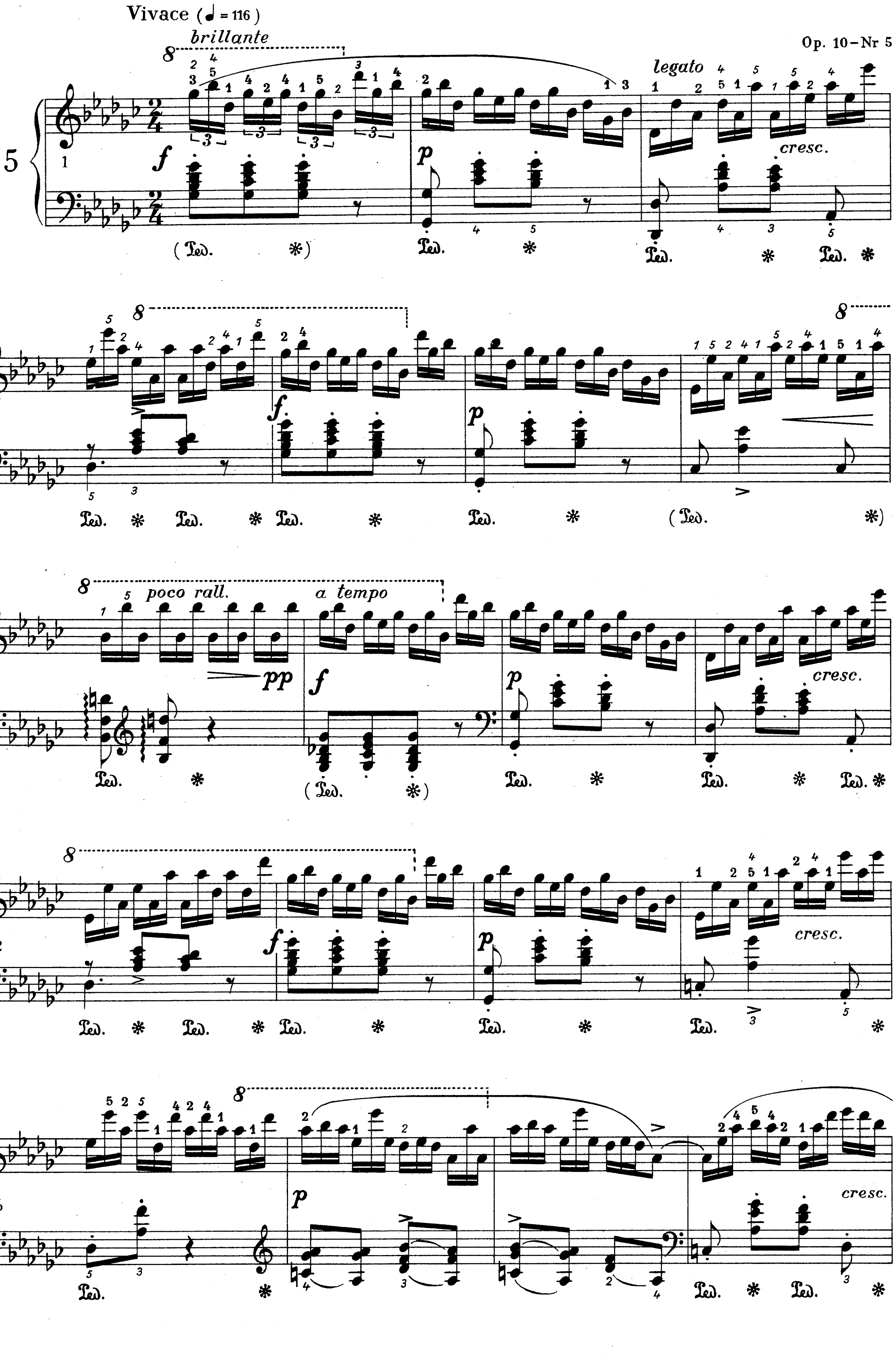
Link to original sheet music


♩ = 176
5
3
5
7
3
5
9
11

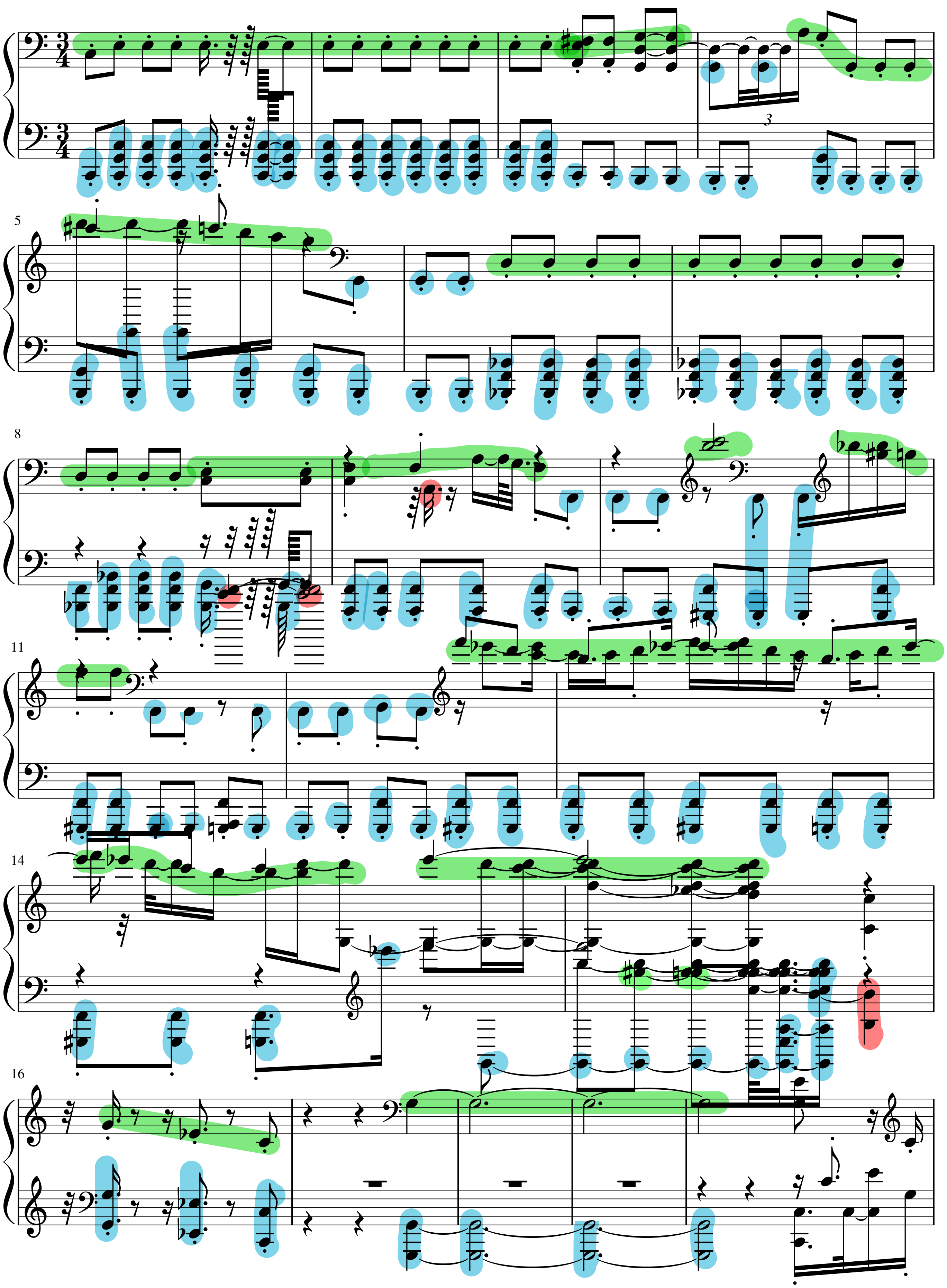
Op. 10–Nr 5

5

Vivace (♩ = 116)

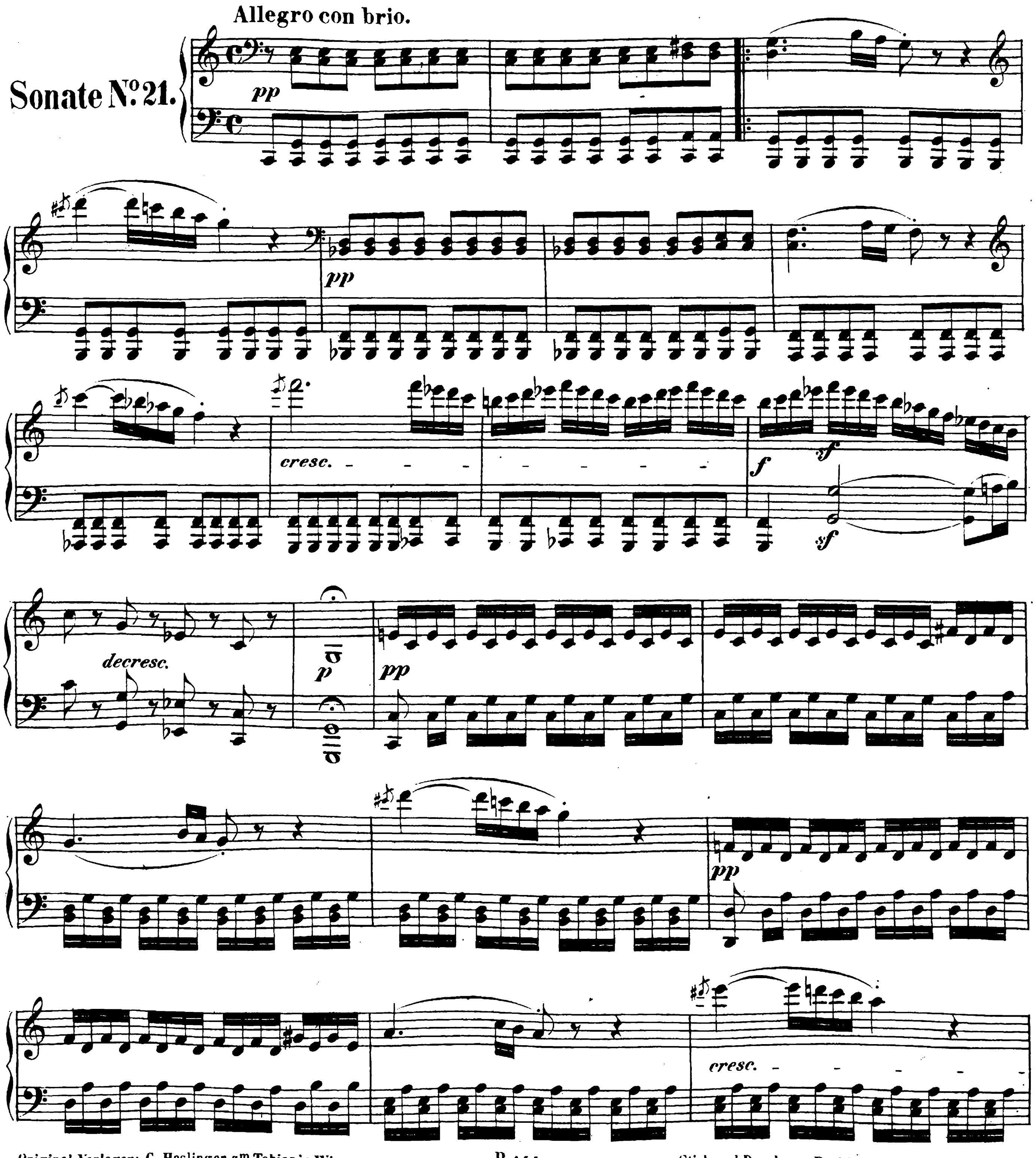


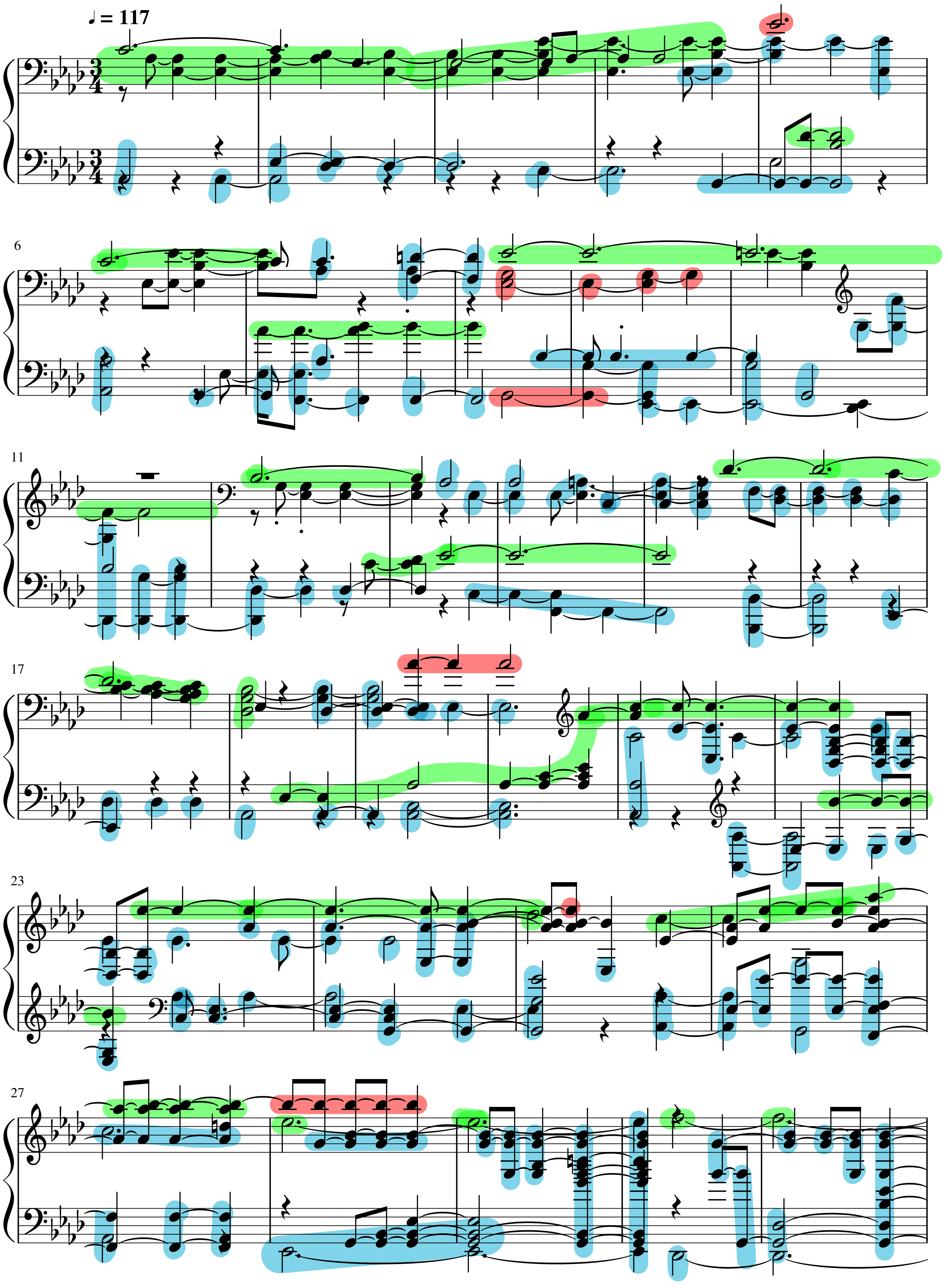
♩ = 158



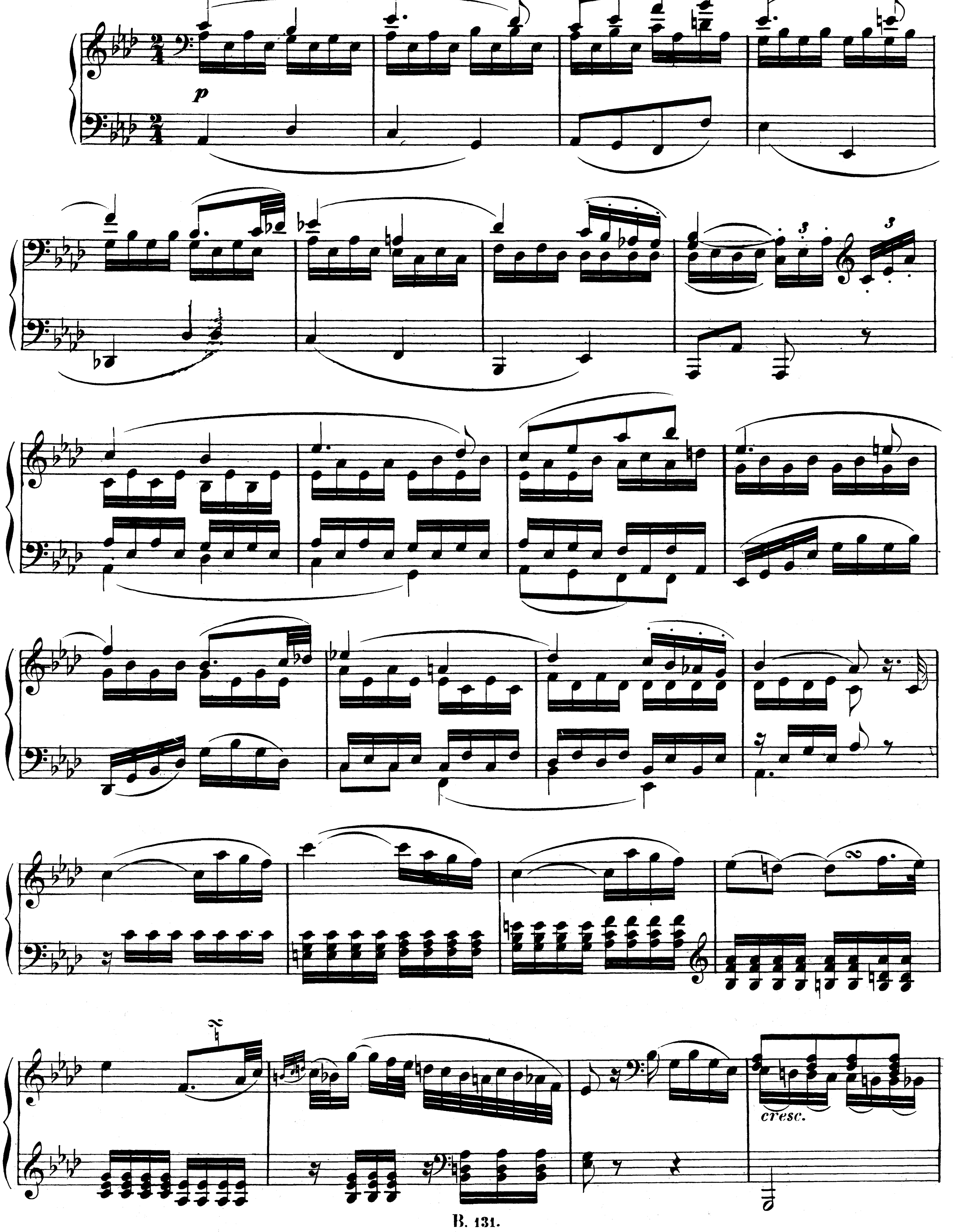
SONATE
für das Pianoforte
von
L. VAN BEETHOVEN.
Dem Grafen von Waldstein gewidmet.
Op. 53.

Sonate Nº 21.




♩ = 117
6
11
17
23
27

Adagio cantabile.



14 Results

The comparison of the generated and original sheet music shows that the application is able to accurately transcribe grand piano recordings into sheet music. The generated sheet music closely resembles the original sheet music in terms of notes and proportion of notes length. However, there are a lot of differences in used notation, measure, tempo. The biggest problem is with grace notes, which are transcribed as regular notes, as well as measure.

Interestingly, the transcribed notes are more precise than original. What I mean by that is that AI can capture the exact note length played by the pianist. That is not always good, because the pianist can play the note a little bit longer or shorter than it is written in the sheet music. That results in darkened transcription because the model transcribes perfectly the note length.

In terms of pitch, the AI model is able to accurately capture the notes played by the pianist. The generated sheet music reflects the original recording with a high degree of precision, capturing the nuances of the performance. The application is able to transcribe complex piano pieces with multiple voices and intricate rhythms, producing sheet music that is faithful to the original recording (but not necessarily to the original sheet music).

15 Conclusion

The application successfully transcribes grand piano recordings into sheet music with a high degree of accuracy. The generated sheet music closely resembles the original recording in terms of notes, rhythms, and dynamics. The AI model is able to capture the nuances of the performance, producing sheet music that accurately reflects the pianist's interpretation. The application provides a user-friendly interface for uploading audio files, viewing PDF previews, and downloading the resulting sheet music and MIDI files. The system architecture is designed to be modular and scalable, allowing for easy updates and future expansion. The application meets the performance, usability, compatibility, maintainability, and scalability requirements, providing a reliable and efficient solution for transcribing grand piano recordings to sheet music.

16 Possible Improvements

- **Improved AI Model:** Developing second AI model that will be responsible for postprocessing the output of the first model. This model will be responsible for correcting the tempo, measure, and notation of the generated sheet music. I believe that it is possible to acchieve, because there are some dependencies between notes that can be used to correct the output.
- **Enhanced User Interface:** Adding more features to the user interface, such as real-time transcription visualization, audio waveform display, and interactive sheet music editing.
- **Additional Output Formats:** Supporting more output formats, such as MusicXML, MIDI, and audio files, to provide users with more options for sharing
- **Performance Optimization:** Optimizing the transcription process to reduce the time required for generating sheet music and improving the accuracy of the results.

- **Creating web application:** Developing a web-based version of the application to make it accessible from any device with an internet connection.

17 References

- Qiuqiang Kong, Bochen Li, Xuchen Song, Yuan Wan, and Yuxuan Wang. "High-resolution Piano Transcription with Pedals by Regressing Onsets and Offsets Times." arXiv preprint arXiv:2010.01815 (2020). [pdf]