Melody Estimation from Polyphonic Music Audio

Harsh Jain

November, 2023

Abstract

This report outlines the implementation and approach to melody estimation from polyphonic music audio. It describes the dataset preparation, preprocessing steps, and the architecture of the neural network employed to predict the melody from a given music piece.

1 Introduction

Extracting the melody from polyphonic music is a challenging task in music information retrieval. This report follows a structured approach, beginning with the dataset description, followed by preprocessing, and then detailing the neural network architecture used for melody estimation.

2 Key Properties of the method

2.1 Auxillary network, VAD

We include a non-voice/voice label while preprocessing the annotation files where the frequency is non-zero/zero. This auxiliary network benefits the main network for pitch estimation to target only those regions where singing voice is present. This auxiliary network also benefits from the main network for voice activity detection with limited data set. Experiments have shown that this non-voice labels is also known to have regularization effect.

2.2 Classification Vs Regression

It is clear that using a regression model directly on raw frequency is not wise. A better option is to do regression on midi note numbers. It is pretty argumentative to classify something as discrete or continuous when it comes to measurement problems in computers or electronics. It highly depends on the level of precision or resolution we are concerned with.

The regression model involves heavy training and a large data set to cover the entire dynamic range which increases with the level of precision. For this problem, I am using the precision of 6.25 cents which will be later compared

with the results of a regression model targeting midi numbers. (Gives poor performance, especially at the edges where there is a shift between speech/no speech)

3 Dataset

The dataset constitutes a fundamental component of the study, providing the raw material upon which the melody extraction models are trained, validated, and tested.

3.1 Description of the Dataset

The dataset utilized in this Project is derived from MedleyDB, a large-scale collection of annotated music. It includes 65 songs with singer and background music (mixed tracks) and 65 corresponding songs with isolated singer tracks (vocal-only tracks). The annotations for each song are provided in CSV files, where each row corresponds to a timestamped instance. The CSV files contain two columns: the first column records timestamps at intervals of 2.9ms, and the second column specifies the fundamental frequency (F0) of the singing voice at each timestamp.

3.2 Dataset Splits

The dataset was divided into three subsets to facilitate the model's training and evaluation:

- **Training Set:** A subset of songs used to train the model, consisting of both mixed and vocal-only tracks.
- **Validation Set:** A separate subset used to fine-tune the model parameters and prevent overfitting, which includes songs not present in the training set.
- **Testing Set:** A reserved subset of songs used solely for evaluating the model's performance, ensuring no overlap with the training or validation sets.

The exact distribution of songs across these subsets is determined based on the need for a balanced representation of musical genres and difficulty levels.

3.3 Dataset Augmentation

To enhance the model's ability to generalize from the training data and to increase robustness against variations, the dataset underwent an augmentation process. Techniques such as pitch shifting, addition of noise, and time stretching were applied to the training set, thereby artificially expanding the diversity of training samples. The augmentation process ensures that the model is exposed

to a broader range of scenarios, simulating different recording conditions and vocal techniques.

4 Preprocessing

4.1 Audio Resampling

The audio files were first merged into a mono channel and then downsampled to a sampling rate below 8 kHz. This resampling process is crucial as it focuses on the frequency range where the majority of the singing voice spectrum is distributed, thus reducing computational complexity and avoiding unnecessary information.

4.2 Windowing and Framing

To prepare the data for spectral analysis, each audio file was segmented into frames using a 1024-point Hann window with a hop size of 80 samples, which corresponds to 10 ms. This framing technique allows the capture of the temporal dynamics of the audio signal, which is vital for tracking the melody in the time-varying audio signal.

4.3 Spectrogram Computation

For each frame obtained in the previous step, a spectrogram was computed to transform the time-domain signal into the frequency domain. The magnitude of the spectrogram was then compressed into a logarithmic scale to better match the human perception of sound. This step results in a two-dimensional representation of the frequency content of the signal over time, where each column corresponds to a vector of frequency bins, representing the distribution of energy at various frequencies at a given time point.

5 Data Preparation

5.1 Feature Extraction

The spectrograms serve as the primary features for the regression model. Audio files were first merged into mono and downsampled to below 8 kHz. Spectrograms were computed using a 1024-point Hann window with a hop size of 80 samples (**10 ms**). We extracted 513 frequency bins from the spectrogram, covering the range from 0 Hz to 4000 Hz. For temporal context, **31 consecutive frames** were included as the input for each data point for the **auxiliary network for VAD**

5.2 Normalization

Before feeding the spectrogram into the regression model, normalization was performed to scale the frequency bin magnitudes. This step ensures that the model receives the input data in a consistent range, facilitating the training process and improving convergence speed.

5.3 Label Encoding

For the regression model, the pitch labels, ranging from D2 (73.416 Hz) to B5 (987.77 Hz), were converted from a categorical format to a continuous frequency value. This conversion allows the model to predict the pitch directly in Hertz. The labels were computed to a resolution of 1/16 semitone (6.25 cents), and a special "non-voice" label was designated for frames without singing voice. Each frame was considered correctly estimated if the difference between the predicted pitch and the ground-truth pitch was within ± 50 cents tolerance, as evaluated using the mir-eval library.

6 Model Architecture

6.1 Convolutional Blocks

The Convolutional Block (ConvBlock) is the initial module responsible for extracting high-level features from the input spectrogram. It consists of two convolutional layers with 3x3 filters, each followed by batch normalization (BN) to stabilize learning and a leaky rectified linear unit (LReLU) activation function to introduce non-linearity. This configuration allows the model to learn a variety of features from the spectral inputs.

6.2 Residual Blocks

Following the ConvBlock, the architecture includes three Residual Blocks (ResBlocks). Each ResBlock is a variant of the ConvBlock, with an additional batch normalization and leaky ReLU, followed by a max-pooling layer that downsamples the feature maps in the frequency axis while preserving the time dimension. A skip connection is also included to facilitate gradient flow during training, which allows each ResBlock to learn residual functions with reference to the layer inputs.

6.3 Pooling and LSTM Layers

After the ResBlocks, a Pooling Block (PoolBlock) comprising batch normalization, leaky ReLU, and max pooling is applied, reducing the dimensionality and preparing the feature maps for sequence processing. The bi-directional Long Short-Term Memory (Bi-LSTM) layer processes the output from the PoolBlock. It considers both past and future context for each time frame in the sequence,

enabling the model to predict pitch labels for each frame in the sequence-to-sequence manner.

7 Loss Function

The loss function is a critical component in training neural networks, providing a measure of prediction accuracy and guiding the optimization process. For the task of melody extraction, a specialized loss function was developed to handle both pitch estimation and voice detection.

7.1 Pitch Estimation Loss

For pitch estimation, the loss function was designed to handle the classification nature of the task. The continuous frequency values for pitches were converted to **midi numbers** as targets for the classification model. To account for the proximity of predicted pitches to the ground truth, a tolerance window was implemented, considering a prediction correct if it fell within ± 50 cents of the true pitch value. This allowed the model to be rewarded for near-correct predictions, smoothing the loss surface and aiding in gradient-based optimization.

7.2 Voice Detection Loss

The voice detection task was framed as a binary classification problem, distinguishing between voiced and unvoiced frames. The corresponding loss function was binary cross-entropy, which measures the discrepancy between the predicted probabilities of voice presence and the ground truth binary labels.

7.3 Joint Loss Function

To jointly optimize the network for both pitch estimation and voice detection, a combined loss function was used. This joint loss function is a weighted sum of the pitch estimation loss and the voice detection loss. The weighting factor balances the contribution of each task to the overall loss, allowing the model to optimize for both tasks simultaneously without one dominating the learning process. The weight factor was empirically set to 0.5.

8 Training

8.1 Optimization Algorithm

The network parameters were initialized using He uniform initialization to facilitate deeper network training by maintaining the variance of activations across layers. Training was performed using the Adam optimizer, a popular choice for its adaptive learning rate capabilities, which can help converge to the optimal set of weights more efficiently. The initial learning rate was set to 0.002 and was

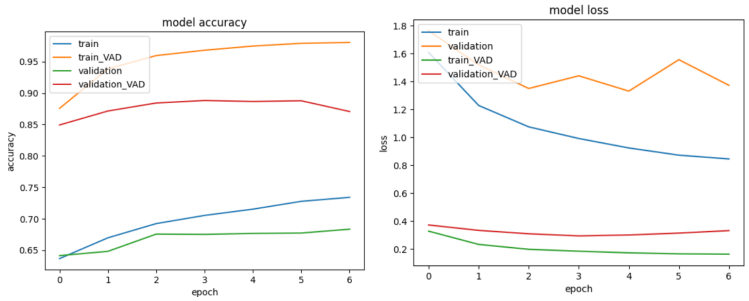
adjusted based on the validation loss; if no improvement was observed within three epochs, the learning rate was reduced to 80% of its value to fine-tune the network's weights.

8.2 Results Discussion

A Jupiter notebook of the implementation and final model weights are submitted along with this report. The submission also contains python script for generating annotations for an audio file where first column is the timestamp and second column is the F0 estimated.

On a similar training environment with the same data set, the classification model gave 78% accuracy for pitch estimation and 98% for VAD whereas the similar regression model gave a mere 54% accuracy for pitch estimation on the validation set.

Below is the graph of training the model on a new Dataset



Kum, S.; Nam, J. Joint Detection and Classification of Singing Voice Melody Using Convolutional Recurrent Neural Networks. Appl. Sci. 2019, 9, 1324. https://doi.org/10.3390/app9071324