Predominant Melody Extraction using U-Net and Mel Spectrograms

Daniel Sheehan

Introduction

As a lifelong musician, I have learned that backing tracks for practicing are hard to come by for more obscure songs. For this project, I attempted to solve this problem by splitting up a track into its melody and accompaniment. This problem falls under the Music Information Retrieval (MIR) subfield of artificial intelligence. This subfield focuses on utilizing artificial intelligence to extract information about music.

My solution to this problem is to apply the U-Net neural network architecture to the Mel-Spectrogram of a song. This architecture was originally used for image segmentation (particularly in the biomedical field) [5]. The algorithm is best for separating objects in an image (such as cells) from the background. Since music can be represented as an image (such as spectrograms), the U-Net architecture is apt for this problem. My algorithm uses U-Net to create an approximation of the spectrogram of the predominant melody of a song.

Similar experiments for achieving this have been performed, as with *On the Use of U-Net for Dominant Melody Estimation in Polyphonic Music* by Doras, Esling, and Peeters, 2019. They perform melody extraction using U-Net, utilizing the HCQT (Hybrid

Constant Q Transformation) as the representation of audio. HCQT represents audio in its time, harmonic, and frequency dimensions. Their model was trained using curriculum training. Curriculum Training involves using increasingly harder tasks to train the model, which is similar to how humans learn. In contrast, in my experiment, I utilized the Mel-Spectrogram as the audio representation and trained in the standard way.

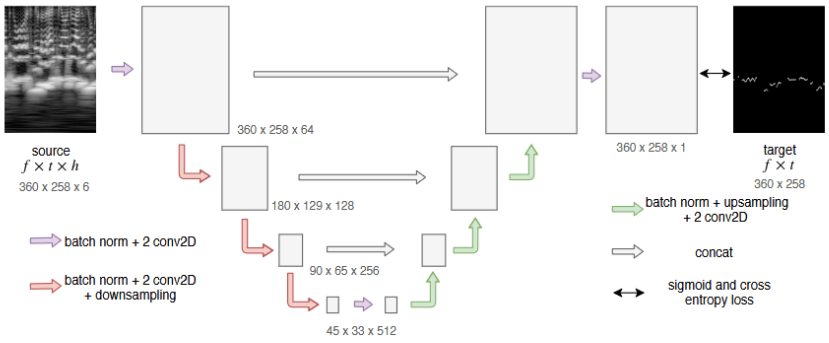

Figure 1: Architecture of U-Net for source separation as proposed by Doras, Esling, and Peeters (2018).

Mel-spectrograms are a frequency/time graph of an audio signal, with the frequency is measured in Mels. Mels measure frequency on a logarithmic scale, similar to how humans perceive audio (i.e. the frequency of C2 is twice the frequency of C1, and so on). I used a more human representation of audio instead of HCQT for easier representation (2 dimensions instead of 3) and the fact that I hoped that the human-centric nature of the Mel scale would fill in for the loss of harmonic data.

The U-Net architecture can also be directly applied to the audio's waveform, as in *Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation* by Stoller, Ewert, and Dixon (2018). However, due to the high computational power required for parts of their architecture, I was not able to implement it. However, I will be

using their method for separating the accompaniment from the source. They model the mixture as the summation of its parts. By first extracting the melody using the U-Net architecture, you can then extract the accompaniment by simply subtracting the melody estimation from the source audio. This will result in the original track only containing frequencies from the accompaniment.

Methods

Tools

My implementation of U-Net for melody extraction was done utilizing the PyTorch Python library. Pytorch is a Machine Learning library focused on the easy implementation of Neural Networks. From PyTorch, I used their built-in functions for loading data, convolutional neural network layers, activation functions, loss functions, and normalization functions. My Dataset was custom-written, though I did superclass PyTorch's Dataset class. All transformations, as well as loading the audio from a .wav file into a tensor, were done with PyTorch's TorchAudio library.

Data

The dataset used was DSD100, compiled by Rafii, Liutkus, Soter, and Mimilakis (2017) [3]. I chose this dataset over its newer version (MusDB18) because it is smaller, lessening the burden on my hardware, and because it contains many of the same audio tracks. This dataset contains the audio tracks for 100 tracks, as well as separate tracks

for each element in the mixture (vocals, bass, drums, and other). Each song is presented in the .wav format, with a sample rate of 44100 and 2 channels. In audio, the sample rate is the amount of data contained in 1 second. For training and testing, due to hardware limitations, I segmented the audio to only contain the first 30 seconds of the song. With longer times, the speed of the training was greatly reduced and would also exceed the amount of storage my GPU (NVidia RTX3050) had. Additionally, for greater precision, I resampled the data to have a rate of 16000. This “stretched” out the data, which made it so that each individual second stored less data, and therefore made the algorithm more precise. Lastly, all data was converted into a Mel-Spectrogram. A spectrogram is a visual representation of an audio file’s frequencies, with time on the x-axis and frequency on the y. The Mel-Spectrogram and Resampled audio were both created using the torchaudio library.

U-Net

The U-Net architecture contains 2 main components: the encoder and the decoder. Each component in the encoder layer contains a convolutional layer, followed by a batch normalization layer, and then applying the ReLU activation function. That pattern is again repeated and subsequently followed by a max-pooling layer. Each level of the encoding layer increases the number of channels by 2. The decoding layer follows a similar pattern, but instead of applying a max pooling layer at the end, a transposed convolutional layer is applied at the beginning. Additionally, after the application of the transposed convolutional layer, the data is padded so that it has the same dimensions as the encoding layer at that level. Before applying the convolutional block, the padded

logit is concatenated with the output of the corresponding encoding layer. This "skip-layer" architecture is what separates the U-Net from standard encoder/decoder convolutional architectures. In my implementation, I encoded to 1024 channels and decoded back down to 2 channels. For the output layer, I used the same convolutional architecture, but to ensure the output had the same dimensions as the input, I applied an average pooling layer and padded the result to have the same dimensions. Since padding is only done at the edges of the image, where only the extremely high and low frequencies in a spectrogram lie, this padding should not greatly affect the output.

Training

My training set consisted of 75 tracks from the DSD100 dataset. Since DSD100 initially comes with 50 tracks for training and 50 for testing, I randomly selected 25 tracks from the testing set and manually moved them over to the training set. When training, I discovered

that, unlike in [1], better results were obtained using ReLU for the final activation function instead of Sigmoid and Mean-Squared Error loss for the loss function, as suggested by Stoller, Esler, and Dixon. I utilized PyTorch's SGD (standard gradient descent) optimizer with a learning rate of 0.001 and a momentum of 0.9. Training was done over the training set with batch shuffling with a batch size of 8 over 50 epochs.

Output

To extract the melody from the spectrogram the model outputs, you must first perform element-wise multiplication between it and the original mix's spectrogram. This should

give you the spectrogram for just the melody. Subtracting the extracted melody's spectrogram from the mixture's spectrogram will result in the spectrogram for the accompaniment (as per [2]). To recover the actual audio from the Mel-spectrogram using PyTorch, you must apply the InverseMel transformation to convert it from the Mel scale, and then apply the Griffin-Lim algorithm to convert the spectrogram to waveform. Lastly, since my model resamples the data from 44100 samples/second to 16000 samples/second, you must resample the new waveform back to its original rate.

Results

My model performs well for tracks containing a more obvious predominant melody. Figure 2 shows the estimation predicted by the model compared to the actual melody.

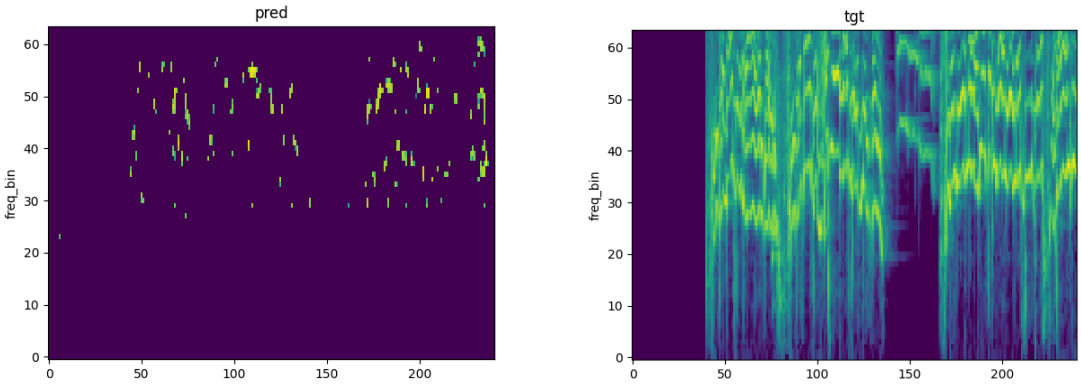

Figure 2: Estimation compared to the target of a song with an "obvious" predominant melody: the overall shape of the prediction matches the changes in pitch and rhythm of the target.

However, if a track (or parts of a track) does not contain the melody, my implementation cannot detect this. It will then assume that the background elements are the predominant melody, because they appear the loudest and most frequently on the spectrogram. In such a scenario, the model appears to output a binary mask estimation of the accompaniment as shown in Figure 3.

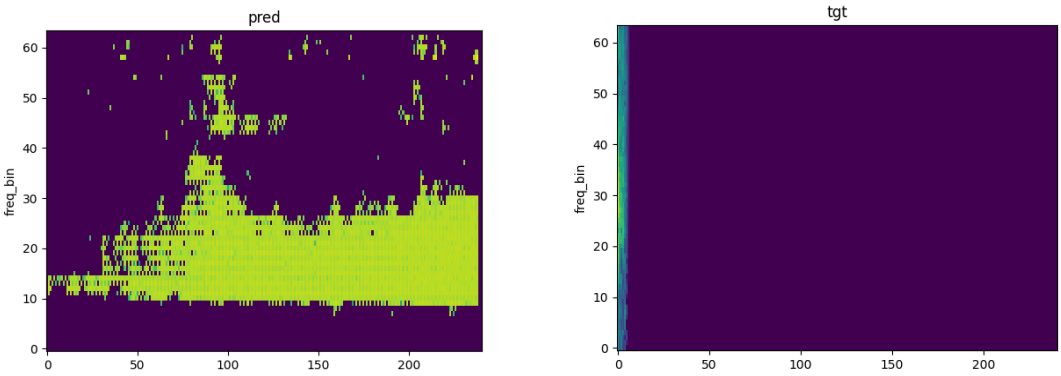

Figure 3: Estimation of the melody for a track without a predominant melody: The shape of this prediction appears to have predicted the accompaniment because there is no predominant melody in the track.

After 50 epochs of training, the model achieved an average loss of 6.422 using the Mean Squared error loss function.

Conclusions

From this experiment, I have determined that the U-Net applied to the Mel-Spectrogram is not enough to properly segment the audio into melody and accompaniment. In my experiments, I discovered that when there is no predominant melody or if the melody does not play for a large portion of the track, the model will assume that the

accompaniment is the predominant melody. Due to this oversight and inability to recognize when the melody is not playing, I conclude that the U-Net alone is not enough to properly segment the audio.

In addition to this, I also do not believe that the Mel-spectrogram contains enough of the data necessary to perform the extraction. In comparison to [1], their approach using HCQT transforms seemed to be able to achieve greater accuracy than my Mel-spectrogram only approach. Additionally, Wave-U-Net's approach has made it one of the top-performing models for this task when evaluated by some metrics.

A potential solution in my approach could be to add instrument or vocal recognition layers. If utilizing these, when the model incorrectly considers the accompaniment to be the melody, the recognition layer would be able to take those segments out. Such an approach was previously discussed in *Singing Voice Separation with Deep U-Net Convolutional Networks* by Jansson, Humphrey, Montecchio, Bittner, Kumar, and Weyde (2017). In addition to this new voice recognition layer, this approach uses attention gates to improve the attention gates of the U-Net architecture.

One limitation of this task is the lack of data sets. There are very few datasets dedicated to this task, and the few available only contain the melody in vocal form. Additionally, many MIR datasets do not contain the audio segmented into their different instruments, but rather information about the instruments playing. Additionally, most publicly available, free datasets only contain music of one or a small handful of genres. For example, DSD100 contains almost exclusively rock, pop, and hip-hop.

In conclusion, the U-Net architecture applied to the Mel-spectrogram is not the most effective method for music source separation.

Works Cited:

[1] Doras, Guillaume & Esling, Philippe & Peeters, Geoffroy. (2019). On the Use of U-Net for Dominant Melody Estimation in Polyphonic Music. 66-70. 10.1109/MMRP.2019.00020. https://www.researchgate.net/publication/332434939_On_the_Use_of_U-Net_for_Dominant_Melody_Estimation_in_Polyphonic_Music?enrichId=rgreq-69a2414bfd681d4b05ab3fbf55f3f0e2-XXX&enrichSource=Y292ZXJQYWdlOzMzMjQzNDkzOTtBUzo3Njg1MTQ1MjA1MjY4NDlAMTU2MDIzOTU0ODUxOA%3D%3D&el=1_x_2

[2] Stoller, D., Ewert, S., & Dixon, S. (2018). “Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation”. ArXiv. https://arxiv.org/abs/1806.03185

[3] A. Liutkus et al. (2017). “The 2016 Signal Separation Evaluation Campaign,”. Latent Variable Analysis and Signal Separation - 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings, pp. 323–332. https://sigsep.github.io/datasets/dsd100.html

[4] Janson et. al. (2018). “Singing Voice Separation with Deep U-Net Convolutional Networks”, International Society for Musical Information Retrieval and Spotify Research. https://archives.ismir.net/ismir2017/paper/000171.pdf

[5] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation." ArXiv. https://arxiv.org/abs/1505.04597