Deep Learning Model to Isolate Vocals from Mono Music Files

Hien To
hto@ucsd.edu

Yuchen Jin
yujin@ucsd.edu

Abstract

For this project, We implemented a CNN-based model to separate and output the vocals from a song (a mixture, including both vocals and instruments in a mono audio file. Specifically, the main technique we will be using is binary masking of spectrogram image of the signal, and then convert the binary-masked spectrogram image back to waveform for outputting. The resulting separation was successful in isolating accompaniments of frequencies more distinct from typical vocal frequencies, but it still under-performs on classifying instruments with more similarities to human voice such as brass instruments, and also sometimes it misclassify vocals, which results in stutters of the vocal in the result.

1 Introduction

Convolutional neural network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other [4]. CNN have multiple layers; including convolutional layer, non-linearity layer, pooling layer and fully-connected layer. It has had ground breaking results over the past decade in a variety of fields related to pattern recognition; from image processing to voice recognition [1].

The other important concept we used is binary mask. Binary masks are used to change specific bits in the original value to the desired setting or to create a specific output value [2]. A binary mask is used to change one or more value from the original matrix to 0 if the corresponding value in the binary mask is 0, and if the value in binaru mask is 0, it will not change the corresponding value in the matrix.

In this project, we aim to use deep learning to isolate vocals from mono music. Inspired by "Audio AI: isolating vocals from stereo music using Convolutional Neural Networks" [3], we use a Convolutional Neural Network (CNN) as our model to train on the spectrograms of each signal segment. The output of the model is a binary mask where value of 1 at certain frequency means that there is vocal. The method we are using is a hybrid of regression and classification, because we classify each pixels of the spectrogram and then append it to previous results to create a larger binary mask. By applying that mask via STFT on the original signal, we can get the isolated vocal signal.

2 Related Works

There are previous works done on this topic. The data set we used is MUSDB18 from SigSep, which is originally used as the data set for SiSEC 2018 Evaluation Campaign [5]. The campaign received more than 30 submissions using various methods including CNN.

What's more, the provided reading, "Audio AI: isolating vocals from stereo music using Convolutional Neural Networks", showed that the naive approach without deep learning cannot solve the problem of isolating vocals from music very well. Inspired by image processing with CNN, it proposes that transforming voice signals into spectrograms, we can train a CNN with those spectrograms as input,

which could be used to generate more accurate binary masks. And using such binary masks, we can get more accurate result signals.

3 Architectures

For our model, we decided to use Convolution Neural Network as the main architecture. This is because we are extracting the vocals by binary masking the spectrogram image of the input signal, and Convulutional Neural Networks are advantageous in training with spatial data, and in this case, 2D spatial data. The activation function that we chose is LeakyReLU, since it is both fast to compute and removes the vanishing gradient problem. We also decided to use max pooling since it helps in extracting low-level features used for classification, and dropout as a generalization technique. In the end, we have 2 Dense layers in order to generate the appropriate binary mask.

A summary of the model:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 1025, 100, 32)     320
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 1025, 100, 32)     0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 1025, 100, 16)     4624
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 1025, 100, 16)     0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 341, 33, 16)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 341, 33, 16)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 341, 33, 64)       9280
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 341, 33, 64)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 341, 33, 16)       9232
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 341, 33, 16)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 113, 11, 16)       0
_________________________________________________________________
dropout_4 (Dropout)          (None, 113, 11, 16)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 19888)             0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               5091584
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 256)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1025)              263425
=================================================================
Total params: 5,378,465
Trainable params: 5,378,465
Non-trainable params: 0
_________________________________________________________________
```

4 Experiments

4.1 Getting the training data

For getting the waveform from audio file, we tried using scipy.wavefile's read method and librosa's load method. Librosa's load takes longer to process the data overall, which is because librosa's load method also normalizes the input signal while reading the audio file. In the end, we chose librosa's load method in this case because training with the normalized data shortens training time per epoch by almost 50%. Therefore, this will allow us to experiment more with different parameters for the model during designing and training.

Then, after getting the waveform signal, we compute the normalized magnitude spectrogram by using librosa's stft method, and then taking the log1p (np's method) of the absolute value of that complex-valued array. Then, we decided to take 100 time frames of the stft to consider at each pass, so we padded the number of time frames of the spectrogram image such that it is a multiple of 100 in order to make the shape of the inputs consistent

4.2 Design of the model

Inspired by the model in "Audio AI: isolating vocals from stereo music using Convolutional Neural Networks", we designed a model with 17 layers including Conv2D layers, Dense layers, Flatten layers, ReLu layers, and Dropout layers. To make it suitable for the dimension of our inputs, we changed the parameters for the Conv2D layers.

However, we encountered hardware problems while compiling and training the model. The GPU memory is limited and it cannot handle the load. So, we have to change the parameters for the Dropout layers as well as the Dense layer.

4.3 Training the model

At first, we used binary cross entropy as our loss function and SGD as our optimizer. However, the loss goes to NaN and the model doesn't learn anything after the first epoch. We then changed the optimizer to Adam and got a trained model.

Since the result that it produced is still not optimal, we tried other loss functions like categorical cross entropy and hinge, but those two loss functions seem to be not compatible with our model. Categorical cross entropy becomes larger and larger in the first few epochs and then goes to NaN. Hinge decreased with an even pace during training, but the the prediction that the model made failed to filter out the accompaniments. We finally decided to use Adam optimizer with binary cross entropy as loss function.

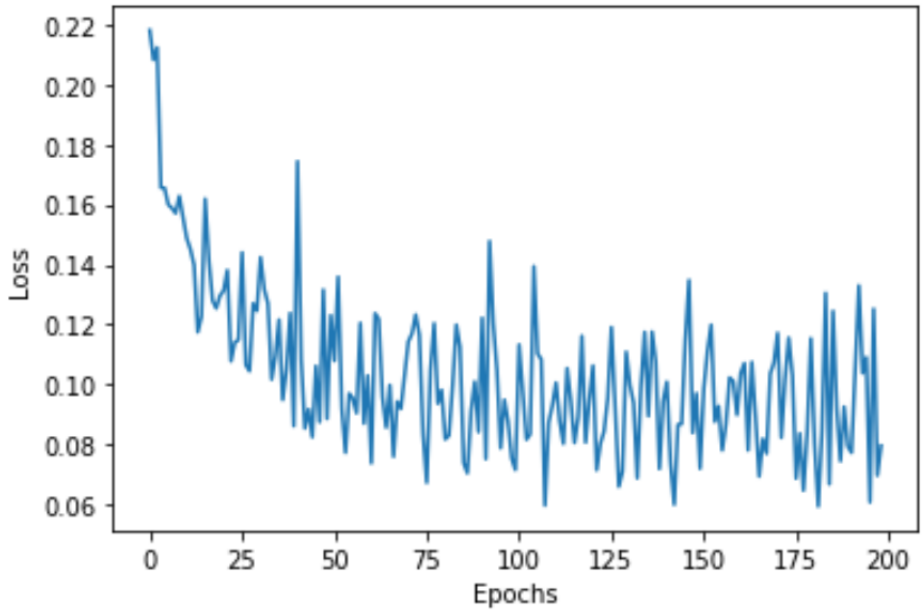
4.4 Inference time

Since our model takes in a spectrogram image and outputs a vocal binary mask, then we have to perform additional steps in inference time that utilizes the binary mask and returns an audio signal. In our first experiment: First, we pad the mask so that it matches the dimension of the input data's spectrogram image, then we apply the mask by multiplying it element-wise with the spectrogram image, and finally we used Griffin-Lim method to convert the masked spectrogram to signal. This method provides the required masked signal as intended, but the Griffin-Lim method is not a perfect reconstruction, so some information is loss during the reconstruction which is clear upon observation. This leads to our second experiment.

For the second experiment, We get the mask similarly, but then we apply the mask by multiplying the resulting mask with the complex-valued array representation of the spectrogram of the input signal (and not the normalized real-valued array). Afterwards, we used librosa's istft method to convert the masked complex-valued spectrogram to get the signal. This reconstruction loses less information than Griffin-Lim's reconstruction, and so the resulting audio sounds much clearer and better than the first experiment.

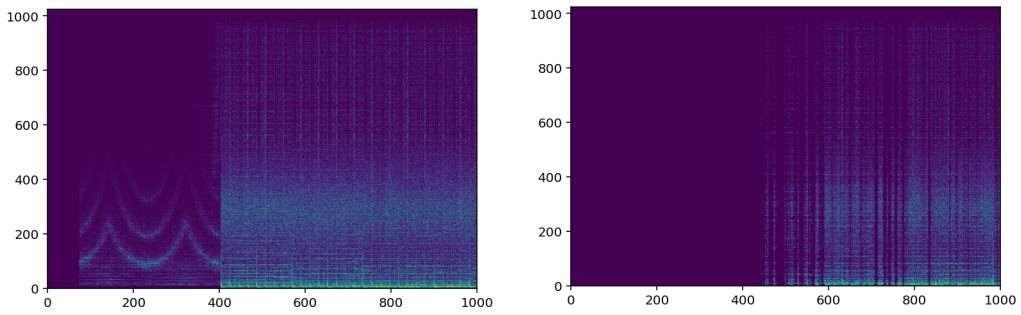
5 Results

We trained our model for 200 epochs, and the training loss is:

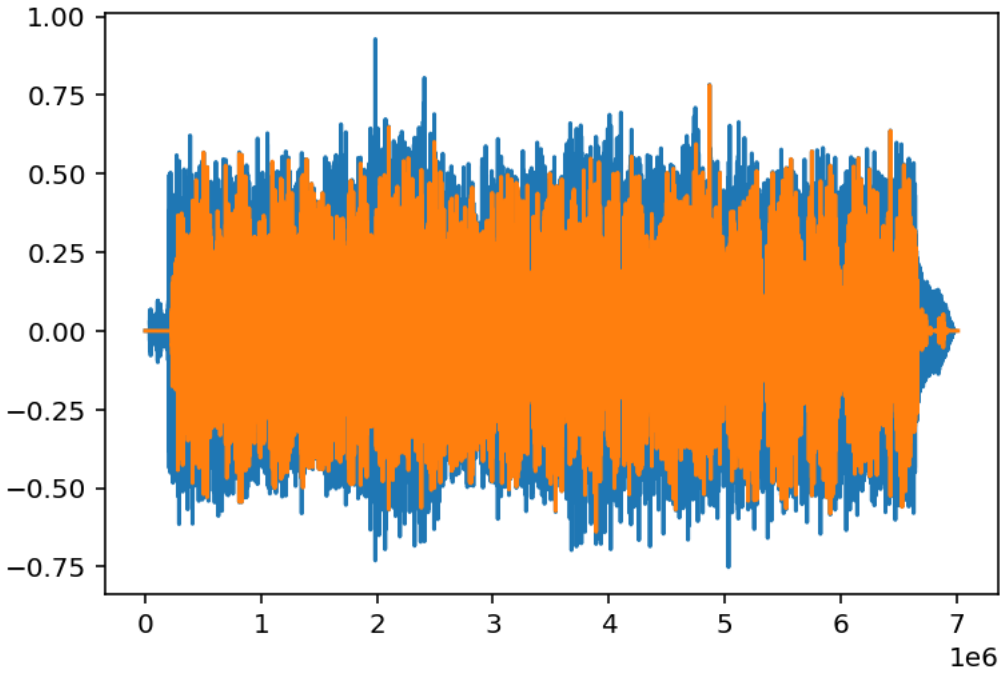


The decrease in loss looks like an exponential decrease, where the loss decreases most significantly during the first 50 epochs, and oscillating up and down for concurrent epochs. Additionally, despite the oscillation, the trough of the loss also sometimes decreases with each iteration, where in this case our loss was the lowest at epoch 166.

By listening to the input and result audio file, and by observing and comparing the graphical representation of the input and result data, we see that our model performs as expected in removing some accompaniments from the vocals. For instance, take our result from "Girls Under Glass - We Feel Alright", the before and after spectrogram looks like:



Here, we are plotting the first 1000 frames of the spectrogram, and we can observe that the result (right) contains much less information than the input (left), especially in the beginning few hundreds of steps, which is because the beginning consists of solely accompaniments. On another note, the difference between the waveform signal between them is:



Although the model performs nicely for certain parts, we can see that the model is not perfect because some information is lost during the masking and reconstruction, which are either vocal data or sound data in general. An additional metric that we used for evaluating the results is the Sound Distortion Ratio of the input and the result signal. In order to achieve this, we used museval's evaluate method. Some of the results were:

Song artiest and name	SDR
Al James - Schoolboy Facination	0.90147978
AM Contra - Heart Peripheral	1.87045608
ANiMAL - Clinic A	1.9558578
Girls Under Glass - We Feel Alright	3.46559538

Link to the code:
`https://github.com/ycc0571/CSE190Project/blob/main/project.ipynb`

Link to media results:
`https://github.com/ycc0571/CSE190Project/tree/main/out`

6 Conclusion and Discussion

This implementation of the CNN model created many paths for continuation for the project. As mentioned in the results, having the ability to train the model with more data, with a more powerful network, and training it for more epochs would aid in yielding a better separation of the vocals from the mixture. However, one main hurdle to these improvements was the lack of hardware resources to train our model. This constrained a lot of our experimentation because we could not tweak and experiment more with the hyper-parameters and the dataset of our project. A future solution would be relying on better infrastructures of computing resources, such as UCSD's DataHub.

Vocals separation is not a simple task, and so there is a vast margin of experiments that we could do with our model and techniques. For instance, a flaw in our model is that sometimes it misclassify instruments that have similar frequencies to human-vocals as vocals themselves. A solution to this would be to increase the number of frequency channels of the data, which will result in better clarity and more distinctions between different components. Another approach is train separate models for different instruments that are under-performing, and then pass the generally-masked result to

those extra masks. Furthermore, we could also improve the result by de-generalizing, filtering, and concentrating our dataset and model such that it works best with a certain genre of music, since different genres usually differs in instruments and vocal techniques, and thus differs in target frequencies. In the end, there are a lot of continuous ongoing research in the industry to solve this problem, which utilizes far more complicated techniques and models than ours. Then, we look forward to exploring further into this topic and improve our research with additional findings.

References

[1] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. *Understanding of a convolutional neural network*. 2018. URL: https://ieeexplore.ieee.org/abstract/document/8308186?casa_token=tkVKd6w5V2cAAAAA:dKxjUGDI57k93bWa-G01-EkEWN1hJ_8hqwH1ew1KY89Nxq5jRKoJ0jAfM1fIVLOBRHTMOeHUJQ.

[2] InetDaemon Enterprises. *Binary Masks*. 2018. URL: https://www.inetdaemon.com/tutorials/basic_concepts/number_systems/binary/masks.shtml.

[3] Ale Koretzky. *Audio AI: isolating vocals from stereo music using Convolutional Neural Networks*. 2019. URL: https://towardsdatascience.com/audio-ai-isolating-vocals-from-stereo-music-using-convolutional-neural-networks-210532383785.

[4] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[5] Fabian-Robert Stoter, Antoine Liutkus, and Nobutaka Ito. *The 2018 Signal Separation Evaluation Campaign*. 2018. URL: https://arxiv.org/pdf/1804.06267.pdf.