Audio Source Separation Using PU Learning

Internship report

Becerril Frédéric

Sugiyama-Yokoya-Ishida Lab

Machine Learning and Statistical Data Analysis

Tokyo University

August 2024

Contents

Abstract

The field of audio source separation aims to isolate distinct sound sources from a mixed audio signal, a task essential in various applications including music production and audio analysis. This study, conducted at Sugiyama-Yokoya-Ishida Lab at Tokyo University, explores the extension of Positive Unlabeled Learning (PU Learning) methodologies to separate different musical instruments such as bass, drums, and vocals.

Building on the foundational work by my advisor, Ito Nobutaka, who developed techniques to separate voice from noise, this research aimed to broaden the application of these techniques to handle multiple instruments within a musical piece. The core objective was to adapt and enhance PU Learning algorithms to achieve high-quality separation of individual instruments from complex audio mixtures.

By leveraging non-parallel training data consisting of mixed and noise signals, the study investigated the effectiveness of PU Learning in the context of multi-instrument separation. The research included developing new models, fine-tuning hyperparameters, and conducting extensive experiments to evaluate performance. Although conclusive results were not achieved, the insights gained provide a valuable foundation for future advancements in audio source separation using machine learning techniques.

Keywords Machine Learning · Source Separation · PU Learning

1 Introduction

1.1 Introduction to the Field and Internship Topic

Audio source separation is a critical area of research within the broader fields of semi-supervised machine learning and signal processing. The primary goal is to isolate distinct sound sources from a mixed audio signal, which is essential for applications in music production, audio analysis, and beyond. This internship focused on extending Positive Unlabeled Learning (PU Learning) methodologies to separate different musical instruments, including bass, drums, and vocals, building on the foundational work of my advisor, Ito Nobutaka, at Sugiyama-Yokoya-Ishida Lab, Tokyo University.

1.2 Origin of the Problem

Many of the initial advances in Music Source Separation (MSS) were inspired by research in speech separation. Speech separation involves isolating individual voices from a mixed audio signal, a significant area of study in audio processing. This research laid the groundwork for MSS by demonstrating the feasibility of separating audio components within a single channel. Unlike speech signals, music signals are more complex due to the presence of multiple overlapping sound sources with distinct frequency ranges, amplitudes, and timbres.

1.3 Relevance and Applications

Music Source Separation (MSS) has numerous applications. In music production, it allows for remixing and mastering by providing isolated tracks of individual instruments or vocals. In music education, it aids students in focusing on specific components of a piece, facilitating better learning and practice. Additionally, MSS is crucial for music information retrieval, enabling the extraction of elements such as melody, harmony, and rhythm from recordings.

1.4 Previous Work

Previous research in MSS primarily employed supervised learning techniques, which require extensive labeled datasets of mixed and isolated audio signals. While effective, these methods are not practical in real-world scenarios due to the scarcity of labeled data. Supervised learning models struggle to generalize across different audio conditions and types of music, limiting their applicability.

1.5 Summary of Work and Comparison with Prior Research

This research aimed to overcome the limitations of supervised learning by employing semi-supervised learning techniques, specifically Positive Unlabeled Learning (PU Learning). By leveraging non-parallel training data consisting of mixed and noise signals, the study aimed to improve the generalization and performance of MSS models. The use of PU Learning allows access to a broader database and enhances the robustness of the model in real-world applications.

1.6 Roadmap of the Report

The remainder of this report is structured as follows:

- **State of the art:** A review of relevant literature in audio signal enhancement and Positive Unlabeled Learning.
- **Experiments and Results:** Presentation and analysis of the experiments conducted, including hyperparameter tuning and performance evaluation.
- **Discussion:** Analysis of the results in comparison with prior works, discussion of challenges and limitations, and potential areas for improvement.
- **Conclusion:** Summary of the findings and suggestions for future research directions.

2 State of the Art

2.1 Convolution

Convolution is a mathematical operation that combines two functions to produce a third function. In the context of image processing, convolution is used to apply a filter (also known as a kernel) to an input image to produce an output image. This operation involves sliding the kernel over the input image, multiplying the values of the kernel and the corresponding pixel values in the image, and summing these products to produce the output pixel value.

Given an input image I and a kernel K, the convolution operation $I * K$ is mathematically defined as:

$$(I * K)(i, j) = \sum_m \sum_n I(i + m, j + n) \cdot K(m, n)$$

Here, $I(i + m, j + n)$ represents the pixel values of the image and $K(m, n)$ represents the values in the kernel. The summation runs over the kernel dimensions.

Parameters of Convolution:

- **Kernel Size:** The size of the kernel (e.g., 3x3, 5x5) determines the area of the image that the kernel covers at each step of the convolution.
- **Stride:** The stride determines the step size of the kernel as it moves across the image. A stride of 1 means the kernel moves one pixel at a time, while a stride of 2 means it moves two pixels at a time.
- **Padding:** Padding adds extra pixels around the border of the input image to control the spatial dimensions of the output image. Common padding types include 'valid' (no padding) and 'same' (padding that ensures the output size matches the input size).

Example: Convolution is a fundamental operation in image processing, particularly in Convolutional Neural Networks (CNNs). The figure below illustrates a convolution operation where a 3x3 kernel is applied to an input image to produce a blurred output image. The kernel used in this example is an averaging filter, which smooths the image by averaging the pixel values in the kernel's neighborhood.

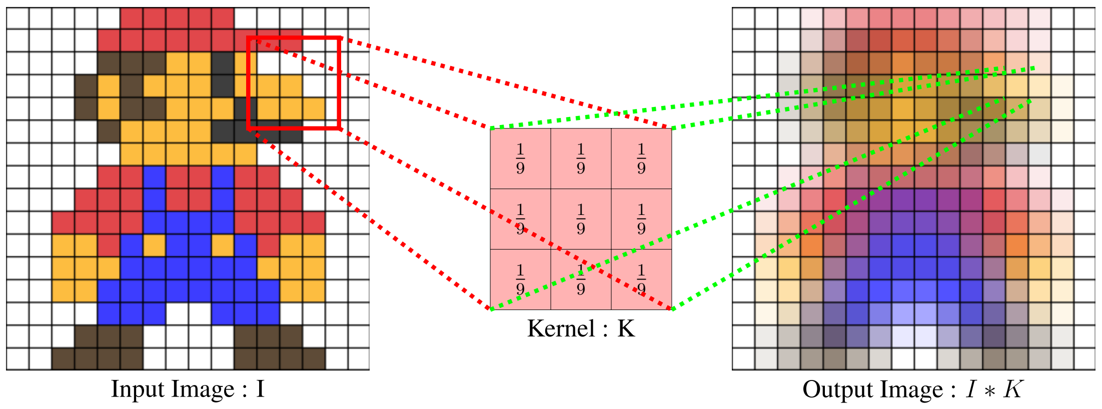


Figure 1: Illustration of a convolution operation where a 3x3 kernel is applied to an input image to produce a blurred output image.

2.2 Machine Learning

Machine Learning (ML) is a field of artificial intelligence that focuses on developing algorithms that allow computers to learn from and make decisions based on data. The primary goal of machine learning is to build models that can generalize from observed data to make accurate predictions or decisions when exposed to new, unseen data. This is achieved by training the model on a dataset and then evaluating its performance on new data, refining the model iteratively to improve its accuracy and reliability. Machine learning is widely used in various applications, including classification, regression, clustering, and dimensionality reduction.

2.3 Convolution Networks

Convolutional Neural Networks (CNNs) are a class of deep neural networks commonly used in tasks involving spatial data, such as image and audio signal processing. CNNs are characterized by their use of convolutional layers, which apply a set of learnable filters to the input data to extract relevant features.

A typical CNN architecture consists of the following components:

- **Convolutional Layers:** These layers perform convolution operations, applying filters to the input to create feature maps that capture spatial hierarchies.
- **Activation Functions:** Non-linear functions, such as ReLU (Rectified Linear Unit), are applied to the feature maps to introduce non-linearity and enable the network to learn complex patterns.
- **Pooling Layers:** These layers downsample the feature maps, reducing their spatial dimensions while retaining the most important features. Common pooling operations include max pooling and average pooling.

- **Fully Connected Layers:** These layers, typically found at the end of the network, flatten the feature maps and connect every neuron to the output layer, enabling classification or regression.
- **Dropout Layers:** Used to prevent overfitting by randomly setting a fraction of input units to zero during training, which helps the network generalize better.

CNNs are highly effective in capturing local patterns and hierarchical structures, making them suitable for tasks such as image classification, object detection, and audio signal processing.

2.3.1 Convolutional Layers

Convolutional layers are the core building blocks of a CNN. They are responsible for extracting features from the input data by applying convolution operations. Let's break down the process step by step, using a series of illustrations to visualize each stage of the convolution operation.

Input Image Representation: The first step is to represent the input image as a set of channels, typically red, green, and blue (RGB), which corresponds to the color channels in the image. The input image is represented as a three-dimensional array of pixel values, corresponding to three color channels: Red, Green, and Blue (RGB). Each channel is essentially a 2D matrix of pixel intensities. The dimensions of this input image are 400x400x3, where 400x400 is the spatial resolution and 3 represents the number of color channels.

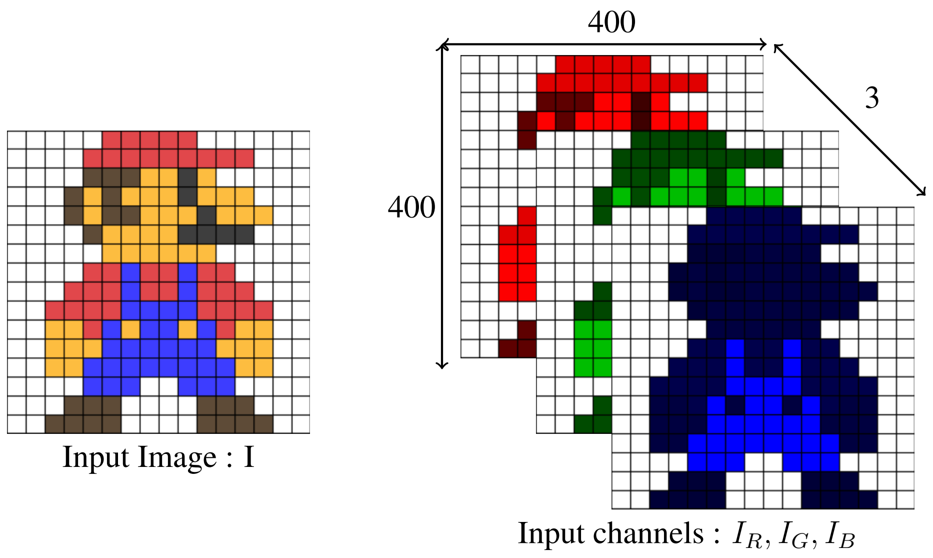


Figure 2: The input image

Convolution Operation: A convolutional layer applies a set of filters (kernels) to the input image. Each filter is a small matrix (typically 3x3, 5x5, etc.) that slides across the image, performing element-wise multiplication with the input and summing up the results to produce a feature map. In this figure, the convolution is applied to the red channel of the input image using a filter, producing an output feature map for that channel.

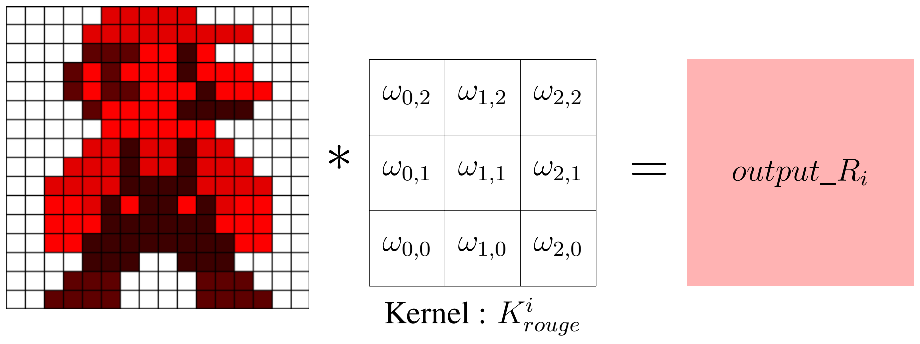

$*$
$\omega_{0,2}$ $\omega_{1,2}$ $\omega_{2,2}$
$\omega_{0,1}$ $\omega_{1,1}$ $\omega_{2,1}$
$\omega_{0,0}$ $\omega_{1,0}$ $\omega_{2,0}$
$=$
$output\_R_i$
Kernel : K^i_{rouge}

Figure 3: Apply the filter to the input image (foreach channel).

Combining Channels: The feature maps generated by applying the filters to each channel of the input image are then combined (usually by summing) to produce a single output feature map for that filter. This combination of outputs allows the convolutional layer to capture information across all channels, thereby preserving the important features from each.

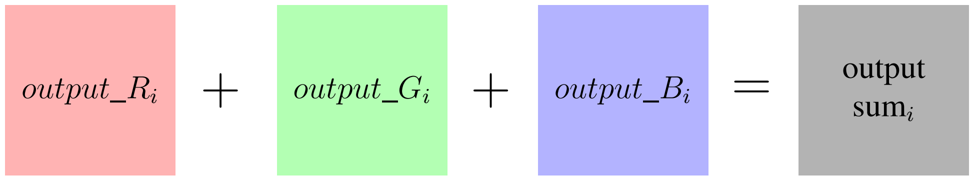


Figure 4: Combining Channels

Activation Function: After combining the outputs from different channels, the resulting feature map is passed through an activation function, such as ReLU (Rectified Linear Unit). The ReLU function introduces non-linearity into the model by replacing all negative values in the feature map with zero, while keeping the positive values unchanged. This operation allows the network to learn complex patterns and features. The result is a non-linear feature map.

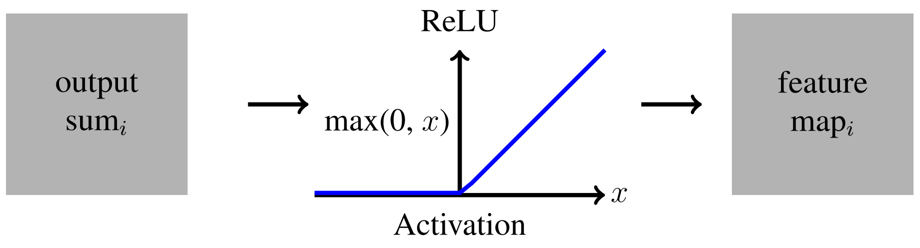


Figure 5: Activation function

Pipeline Overview: This final figure illustrates the entire pipeline of a convolutional layer. The input image is processed through a series of operations: convolution with filters, summing the results, applying an activation function, and producing a final feature map. This feature map is then passed to the next layer of the CNN for further processing, leading to increasingly abstract representations of the input data.

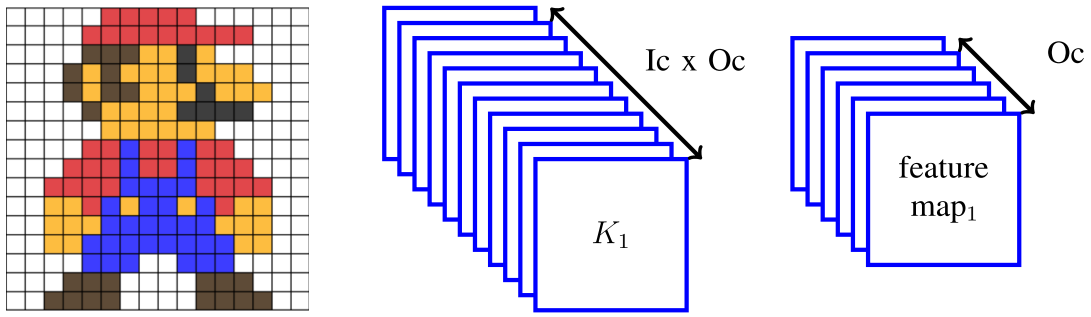


Figure 6: Pipeline Overview of a convolutional layer, **Ic** : Input channels, **Oc** : Output channels

2.4 Positive Unlabeled Learning

Positive Unlabeled Learning (PU Learning) is a weakly supervised learning technique that focuses on learning from datasets that contain only positive (P) and unlabeled (U) data, without explicit negative (N) examples. This approach is particularly useful when labeled negative data is difficult or impossible to obtain. The fundamental objective of PU Learning is to train a classifier to distinguish between positive and negative classes using only the P and U datasets.

Formally, let $\mathcal{X}_P := \{x_i^P\}_{i=1}^{n^P}$ be the set of P (positive) data and $\mathcal{X}_\mathcal{U} = \{x_i^U\}_{i=1}^{n^U}$ be the set of U (unlabelled) data, where $n^P = |X_P|$ and $n^U = |X_U|$ denote the number of P and U training examples, respectively.

Let $f_\theta : \mathbb{R}^d \to \mathbb{R}$ be a classifier parametrised by θ and $\hat{y} := sgn(f_\theta(x))$ be the predicted label for x, where sgn is the sign function

$$\text{sgn}(t) = \begin{cases} +1 & \text{if } t \geq 0, \\ -1 & \text{if } t < 0. \end{cases}$$

Define a loss function $\ell(x, y, \theta)$ as a non-negative measure of the deviation of the classifier f_θ from a data point (x, y). A common choice is the sigmoid loss:

$$\ell(x, y, \theta) = \sigma(-y f_\theta(x)),$$

where $\sigma(m) = \frac{1}{1+e^{-m}}$ is the logistic sigmoid function.

The generalization error or risk is defined as the expected loss:

$$R(\theta) = \mathbb{E}_{(x,y)\sim p(x,y)}[\ell(x, y, \theta)],$$

where $p(x, y)$ is the unknown joint data distribution. Since this expectation cannot be computed directly, we instead minimize the empirical risk $\hat{R}(\theta)$ obtained by averaging over the training data.

In supervised learning, an empirical risk is easily obtained as $R(\theta) \approx \frac{1}{n}\sum_{i=1}^{n} \ell(x_i, y_i, \theta)$ where $\{(x_i, y_i)\} \sim p(x, y)$ are labelled training data

In PU learning, we only have access to positive and unlabeled data, not negative data. Despite this limitation, it is possible to compute an empirical risk using only $\mathcal{X}_P$ and $\mathcal{X}_U$. By expressing the joint distribution $p(x, y)$ as:

$$p(x, y) = \begin{cases} \pi p(x \mid y = +1) & \text{if } y = +1, \\ (1 - \pi)p(x \mid y = -1) & \text{if } y = -1, \end{cases}$$

where $\pi = p(y = +1)$ is the class prior, we can derive the empirical risk $\hat{R}_{PU}(\theta)$ using only positive and unlabeled data.

However, this empirical risk can become negative in certain cases. To prevent this, a non-negative empirical risk $\hat{R}_{\text{nnPU}}(\theta)$ is used:

$$\hat{R}_{\text{nnPU}}(\theta) := \frac{\pi}{|\mathcal{X}^p|} \sum_{x\in\mathcal{X}^P} \ell(x, +1, \theta) + \left(\frac{1}{|\mathcal{X}^U|} \sum_{x\in\mathcal{X}^U} \ell(x, -1, \theta) - \frac{\pi}{|\mathcal{X}^P|} \sum_{x\in\mathcal{X}^P} \ell(x, -1, \theta) \right)$$

where $(x)^+ = \max(x, 0)$ ensures non-negativity.

Initial State: Positive and Unlabeled Data At the beginning of PU Learning, we have access to only positive and unlabeled data. The positive data is labeled, but the unlabeled data can contain both positive and negative examples, which are unknown.

This figure shows the initial state of the dataset, where the green points represent positive labeled data (P) and the black points represent unlabeled data (U). The unlabeled data could potentially belong to either the positive or negative class, but this information is not available at this stage. The challenge of PU Learning is to correctly classify these unlabeled points using the limited information provided by the positive examples.

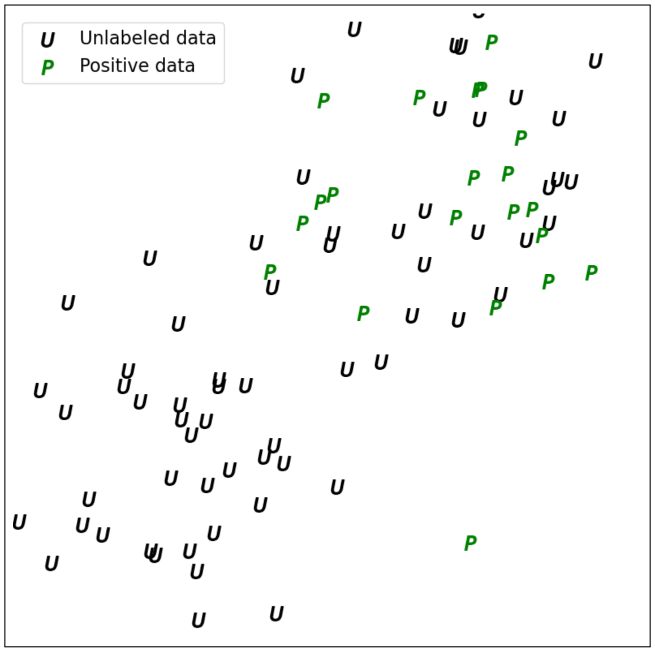


Figure 7: Initial state

Final State: Positive and Negative Data After applying PU Learning techniques, the model is able to distinguish between positive and negative examples within the initially unlabeled data, effectively learning a decision boundary that separates the two classes.

This figure represents the final state achieved after PU Learning, where the data has been separated into positive (green P) and negative (red N) classes. The blue line indicates the decision boundary learned by the model, which effectively discriminates between positive and negative data. The classifier was able to learn this boundary using only the initial positive and unlabeled data.

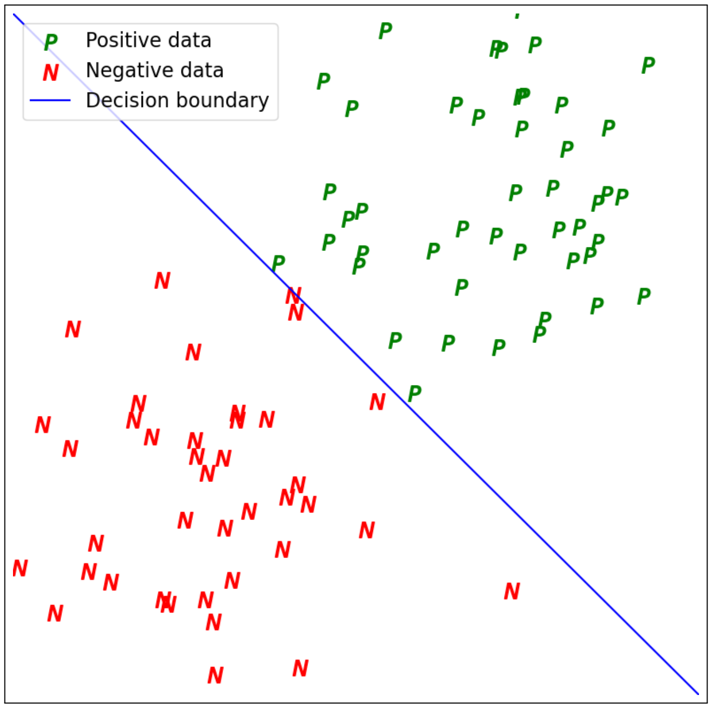


Figure 8: Final State

2.5 Literature Review

The purpose of this literature review is to provide an overview of existing research relevant to the field of audio source separation, particularly focusing on semi-supervised learning and signal processing techniques. Two significant works are highlighted: the study on Positive Unlabeled Learning (PU Learning) for audio signal enhancement by my advisor, Ito Nobutaka, and one of the state-of-the-art methods to solve MSS using machine learning: the paper "Music Source Separation using Band-split RNN."

2.5.1 PU Learning for Audio Signal Enhancement

Ito Nobutaka's work on audio signal enhancement explores the use of PU Learning to improve the quality of audio signals by separating voice from noise Ito and Sugiyama (2023). The study leverages non-parallel training data, which includes noisy signals and pure noise, to train a convolutional neural network (CNN) for classifying spectrogram patches.

The methodology involves defining signal inactivity and activity as positive (P) and negative (N) classes, respectively. Spectrogram patches of noise clips serve as P data, while those of noisy signal clips are used as unlabelled (U) data. The CNN learns to classify each patch as either P or N, enabling effective noise suppression and signal enhancement.

Key findings from this research demonstrate that PU Learning can achieve substantial improvements in audio signal enhancement without requiring clean, parallel data. Future work shall include exploring more sophisticated architectures, conducting more extensive experiments, and applying PU Learning to other audio tasks.

2.5.2 Music Source Separation with Band-split RNN

The paper on Music Source Separation with Band-split RNN presents a state-of-the-art approach to isolating different musical instruments from a mixed audio signal Luo and Yu (2023). This method combines several innovative techniques to enhance the performance of music source separation systems.

The core methodology involves splitting the spectrogram of the audio signal into multiple subbands, each corresponding to a specific frequency range. These subbands are processed using recurrent neural networks (RNNs) to capture both temporal and spectral patterns. The RNNs estimate masks for separating the target source from the mixture, which are then applied to the original spectrogram to isolate the desired instrument.

One of the main reasons the band-splitting is effective is that researchers can fine-tune the splitting using prior knowledge about the data, focusing on important frequencies. The choice of the band-splitting is not handled by any machine learning model; instead, it is manually adjusted by researchers to improve performance.

Experimental results show that the Band-split RNN approach significantly improves the separation performance due to its ability to manage different frequency bands more effectively. However, the study also notes limitations such as increased computational complexity, potential difficulties in handling a wide variety of music genres, and the need for better fine-tuning of the band-splitting for bass frequencies.

2.5.3 Insights and Future Directions

Both PU Learning for audio signal enhancement and Music Source Separation with Band-split RNN contribute valuable insights to the field of audio source separation. While PU Learning addresses the challenge of limited labeled data by using non-parallel datasets, the Band-split RNN method focuses on enhancing frequency resolution for better separation accuracy.

The insights gained from these papers inform the current research in several ways. From PU Learning, the use of semi-supervised techniques allows access to a broader database and improves model robustness. The band-splitting approach highlights the importance of leveraging prior knowledge about the data to enhance performance. These insights will guide the adaptation of PU Learning methodologies to separate various musical instruments, aiming for improved generalization and performance in complex audio mixtures.

3 Experiments and Results

3.1 Work Environment

During my internship, I had the privilege of working at the Sugiyama-Yokoya-Ishida Lab at the University of Tokyo. The lab focuses on Machine Learning and Statistical Data Analysis and is housed across two campuses: the Hongo campus and the Kashiwa campus. The lab provided an excellent environment for research, with multiple rooms dedicated to student collaboration and research activities.

I worked closely with Ito Nobutaka, a researcher who has previously applied Positive Unlabeled (PU) Learning to the task of speech enhancement. Under his guidance, I was able to deepen my understanding of PU Learning and its applications. We met once a week to discuss my progress, challenges, and any difficulties I encountered. These meetings were invaluable, as Ito-san provided me with specific guidelines to enhance my experiments and recommended relevant papers to further my understanding of the field.

My workspace was primarily located in the student room at the Kashiwa campus during the first semester of my internship. This space was conducive to collaboration, allowing me to engage with other students and exchange ideas about our respective research projects. In the second semester, I transitioned to the student room at the Hongo campus, where the collaborative atmosphere continued to be a significant asset to my work.

In addition to the daily interactions with my lab mates, the lab hosted two seminars each week, which were instrumental in broadening my academic perspective. The first was a research seminar, where Ph.D. students presented their ongoing research, offering insights into a wide range of topics within the field. The second seminar, known as the student seminar, was tailored for master's students like myself. Here, we presented papers we had read during our research, which helped in developing our presentation skills and deepening our understanding of various research methodologies.

A particularly memorable event was the "Super Paper Day" seminar. This was an exceptional seminar where every student was required to select a research paper and present it in less than five minutes. This event was both challenging and exciting, as it pushed us to distill complex information into a concise and engaging presentation.

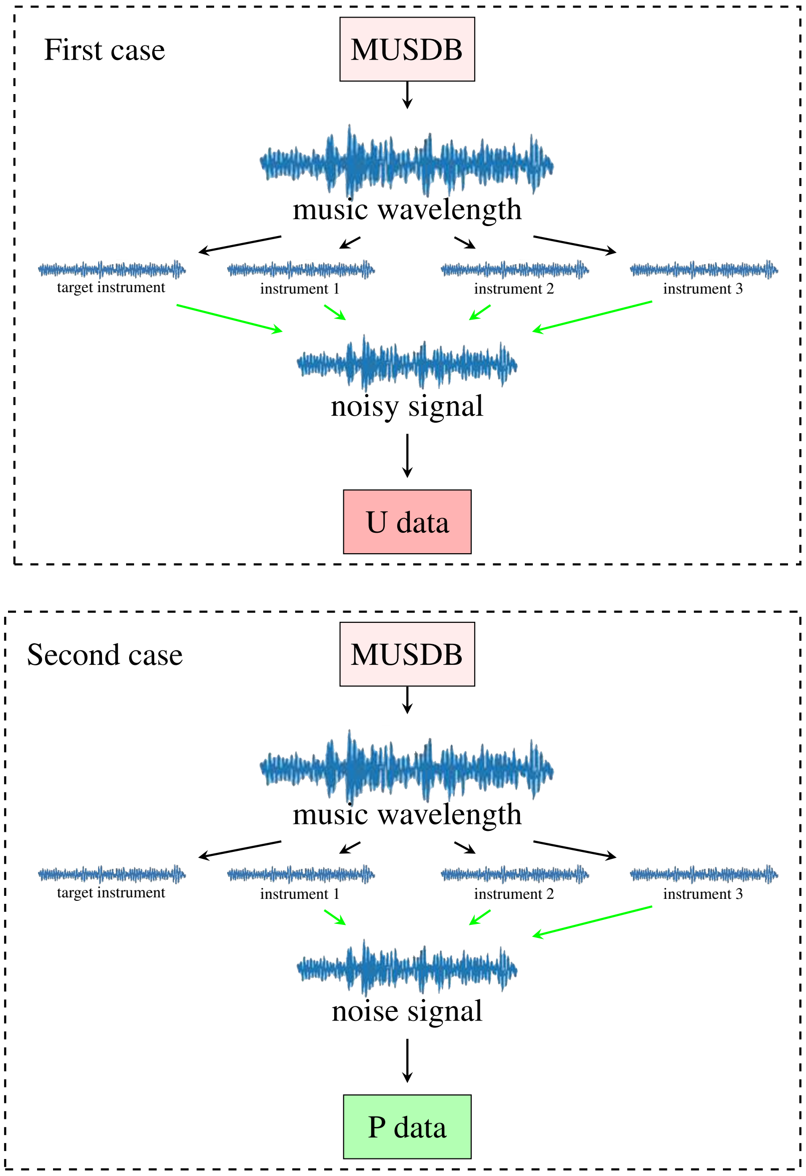
Overall, the supportive environment at the Sugiyama-Yokoya-Ishida Lab, coupled with the structured academic activities, greatly enhanced my research experience and contributed significantly to my personal and professional development during my internship.

3.2 Data and Data Preparation

3.2.1 Data

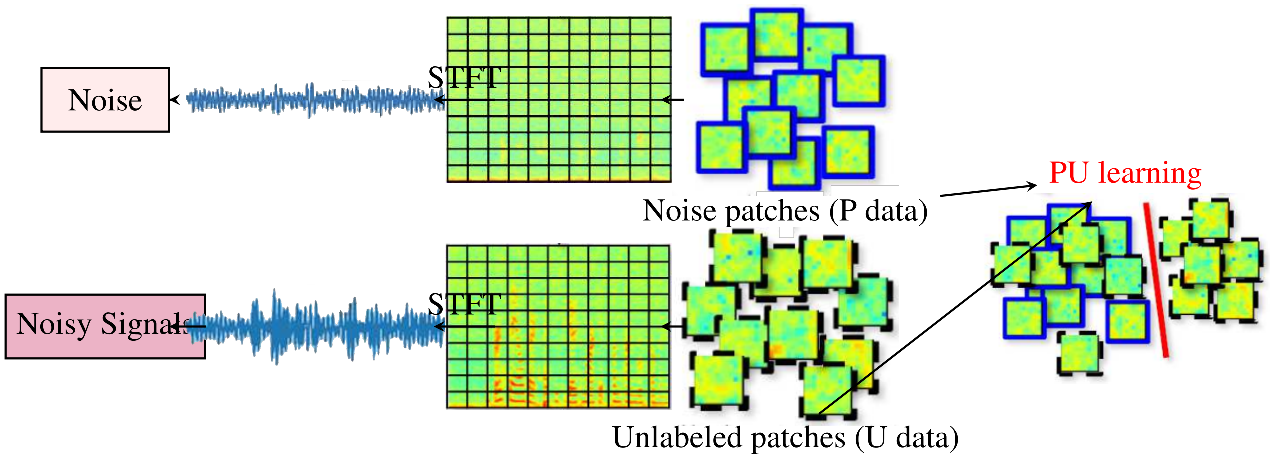
In this project, the primary dataset used was the MUSDB18 database, a widely recognized resource for music source separation tasks. The MUSDB18 dataset is typically used for supervised learning as it provides access to individual instrument tracks. However, for our semi-supervised Positive Unlabeled (PU) learning algorithm, we adapted the use of this dataset to create two types of audio data: positive data and unlabeled data.

The process of creating these datasets is illustrated in the figures below.



Data preparation

Figure 9: Illustration of the data creation process for PU Learning.



The first case involves creating what we term as "unlabeled data" (U data).To create U data, we take the original audio track that contains the target instrument along with other instruments. This audio is labeled as unlabeled data because, although it contains the target instrument, its presence varies across different frequencies and time segments.

The second case involves creating "positive data" (P data). For P data, we generate an audio track by mixing all instruments except the target instrument. The rationale behind this choice is that in an audio clip where the target instrument is absent, it is guaranteed to be absent at every frequency and at every time segment. This makes it ideal for use as positive data in PU Learning. To obtain the target instrument's isolated audio track, we subtract the P data from the original audio track.

This data creation approach leverages the detailed instrument tracks provided by the MUSDB18 dataset, transforming it into a suitable format for our semi-supervised PU Learning algorithm.

3.2.2 Data Preparation

After obtaining the positive (P) and unlabeled (U) data, the next step was to prepare these data for input into the PU learning model. This preparation process involved several key steps, which are illustrated in the following figure.

The preparation process begins by splitting the audio data into segments of t seconds. The duration t, along with other parameters such as the window size, overlap, and FFT size used in the Short-Time Fourier Transform (STFT), are considered hyperparameters that can be fine-tuned to optimize the model's performance.

After splitting the audio into segments, we apply the STFT to each segment to convert the time-domain audio signals into the frequency domain. The resulting spectrograms are then used as input data for the model. The spectrograms generated from the noise signals serve as the positive data (P data), while those generated from the noisy signals serve as the unlabeled data (U data).

To further refine the data, a Source Activity Detector (SAD) is applied to remove silent regions from the audio tracks. This step ensures that only segments with significant audio content (i.e., where the target instrument is active) are retained for data mixing. By filtering out silent regions, the model can focus on learning from the most relevant parts of the audio, which can lead to improved performance in separating the target instrument.

This data preparation process is critical for setting up the PU Learning framework, ensuring that the input data is properly formatted and optimized for the model to learn effectively.

3.3 Experiments

3.3.1 Experiments Environment

In order to conduct my experiments efficiently and take full advantage of the computational resources available to me, I utilized several different environments:

- **Personal Computer (Mac Studio):** I performed a significant portion of my development work and initial testing on my personal Mac Studio. This machine provided a consistent and accessible environment for coding and debugging, though it was somewhat limited in terms of computational power compared to other resources.
- **School Computer (Multiple Graphics Cards):** The more computationally intensive experiments were run on a school computer equipped with multiple graphics cards. This environment allowed for faster processing and the ability to run more complex models that would be infeasible on my personal machine.
- **Google Colab:** I also leveraged Google Colab for running experiments, especially when I needed additional computational resources or wanted to share and collaborate on code. Colab's flexibility and accessibility made it an invaluable tool for scaling my experiments across different environments.

To ensure consistency and reproducibility, I designed my code to be adaptable across these diverse environments. This flexibility allowed me to maximize the number of experiments I could run, as I could seamlessly transition between different computational resources without needing to modify the core codebase. This adaptability was crucial for efficiently managing the experiments and ensuring that I could take full advantage of the resources at my disposal.

3.3.2 Baseline Experiment Using Advisor's Hyperparameters

The first experiment I conducted served as a baseline, using the hyperparameters that had been previously fine-tuned by my advisor for his own work in speech enhancement. The primary goal of this experiment was to establish a reference point for my research and to assess the performance of these pre-established settings on my dataset.

Given that the hyperparameters were optimized for a different type of audio data (specifically, speech signals with a sample rate of 16 kHz), this experiment allowed me to observe how well these settings transferred to my own dataset, which involved musical instruments at a higher sample rate of 44.1 kHz. The results of this baseline experiment provided valuable insights into the initial performance of the model and highlighted areas where adjustments were necessary.

3.3.3 Hyperparameter Tuning

As I progressed with my experiments, it became clear that the differences between my dataset and the one used by my advisor necessitated changes to the hyperparameters. The audio data I was working with not only had a higher sample rate (44.1 kHz compared to 16 kHz), but it also involved different types of sounds, which required capturing more detailed frequency information.

To address these differences, I focused on fine-tuning the following hyperparameters:

- **Clip Duration:** Adjusting the duration of the audio clips used in the model was essential to ensuring that each clip contained enough relevant information for effective learning.
- **Kernel Size:** Given the higher sample rate of my audio data, I increased the kernel size in the convolutional layers to capture a broader range of frequencies. This change helped the model better understand the more complex audio signals in my dataset.
- **Window Size of the STFT:** The Short-Time Fourier Transform (STFT) window size was adjusted to optimize the balance between time and frequency resolution, improving the model's ability to analyze the audio signals.
- **Prior Value:** The prior value, which reflects the expected proportion of positive data, was fine-tuned to better align with the characteristics of my dataset. This adjustment was crucial for improving the model's classification accuracy.

These adjustments were aimed at improving the model's performance by better aligning the hyperparameters with the characteristics of my specific dataset. The results of these experiments demonstrated the importance of tailoring the model to the nuances of the data being analyzed.

3.3.4 Data Preparation and Alignment with Advisor's Experiment

In an effort to further improve the results, I conducted an additional experiment where I maintained the original hyperparameters used by my advisor but modified the data preparation process to more closely align with the conditions of his experiment. The rationale behind this approach was that, by making the data more similar to what my advisor used, the original hyperparameters would be more effective.

For this experiment, I focused specifically on isolating the voice as the target instrument, similar to my advisor's work on speech enhancement. This involved adjusting the data preparation steps to ensure that the input data more closely resembled the types of audio signals (i.e., voice mixed with other sounds) that my advisor's model was originally designed to process.

By making these adjustments, I aimed to leverage the strengths of the existing hyperparameters while ensuring that the data fed into the model was more compatible with those settings. The goal was to see whether this approach could lead to improved performance in isolating the voice from the rest of the music, much like how my advisor successfully isolated voice from noise.

3.4 Results

3.4.1 Summary of Results

Unfortunately, despite extensive efforts to optimize the model and adjust the data preparation, none of the experiments yielded good results. Regardless of how much I fine-tuned the hyperparameters or altered the data preparation process, the model consistently failed to produce valuable outcomes. At certain epochs, the model would filter out all the audio, resulting in blank outputs across all experiments. This was a significant challenge, as it indicated that the model was unable to effectively learn from the data provided.

3.4.2 Discussion of Results

In an attempt to better understand the limitations of the PU learning approach in this context, I also implemented a supervised learning method using the same dataset, as the database was originally intended for supervised learning. The supervised model produced average results, managing to isolate the target instrument to some extent. This comparison highlighted a key issue: while supervised learning could achieve some level of success, the PU learning model struggled significantly.

The implications of these results are clear: I was unable to identify suitable hyperparameters for the PU learning model, which in turn prevented the model from performing effectively. This outcome suggests that either the PU learning approach may not be well-suited for this particular task, or that the specific implementation and parameter settings require further refinement beyond what was achievable within the scope of this project.

3.4.3 Challenges and Limitations

This project presented several significant challenges, the most prominent being the long compute times required for each experiment. The extensive duration of each experiment made it difficult to quickly identify and rectify mistakes or issues in the code or data preparation. As a result, debugging the model and optimizing the process was a slow and painstaking task.

Throughout the project, I encountered and resolved numerous issues related to both the code and the data. However, the time-consuming nature of the experiments meant that progress was often slow, and the iterative process of trial and error was particularly challenging. This limitation significantly impacted the overall pace of the project and may have contributed to the difficulty in achieving successful results.

In summary, while the project did not yield the desired outcomes, it provided valuable insights into the complexities and challenges of applying PU learning to music source separation. The experience gained through these challenges will be instrumental in guiding future work in this area.

4 Conclusion

4.1 Summary of Work

This report has documented the exploration of **Positive Unlabeled (PU) Learning** in the context of music source separation, a complex and challenging task. The internship was focused on extending methodologies previously applied to speech enhancement by my advisor to the more intricate domain of isolating musical instruments from mixed audio signals.

The project began with the adaptation of the MUSDB18 dataset for use in a semi-supervised PU learning framework. Following this, a series of experiments were conducted, which included fine-tuning the model's hyperparameters and adjusting data preparation processes in an attempt to optimize performance. Unfortunately, despite extensive efforts, the experiments did not yield successful results. The model often failed to produce meaningful outputs, and in some cases, it filtered out all audio, resulting in blank outputs.

To better understand these challenges, a supervised learning model was also implemented using the same dataset. Although the supervised approach produced only average results, it succeeded in isolating the target instrument to some degree, further underscoring the difficulties encountered with the PU learning approach.

Throughout the project, several significant challenges were encountered, including long compute times that slowed down the iterative process of debugging and optimization. The complexity of the task, combined with the inherent limitations of the PU learning framework in this context, contributed to the overall difficulty in achieving the desired outcomes.

4.2 Future Directions

While the results of this project were not as successful as anticipated, the experience has provided valuable insights that can guide future research in this area. There are several potential directions for future work:

- **Exploring Alternative Machine Learning Frameworks:** Given the challenges with PU learning, it may be beneficial to explore other semi-supervised or unsupervised learning methods that could be more suited to the task of music source separation.

- **Refining Data Preparation Techniques:** Future work could involve more sophisticated data preparation methods, such as advanced source activity detection or more nuanced data augmentation techniques, to better align the input data with the capabilities of the model.

- **Improving Computational Efficiency:** To address the challenge of long compute times, future projects could benefit from utilizing more powerful computational resources or optimizing the code for faster execution. This would allow for more rapid iteration and experimentation, which is crucial for tasks of this complexity.

- **Investigating Hybrid Approaches:** A hybrid approach that combines elements of supervised and semi-supervised learning might provide a more effective solution. This could involve using a supervised model as a baseline and then gradually incorporating PU learning elements to refine the separation process.

In conclusion, while the project did not achieve its initial objectives, it served as a valuable learning experience. The insights gained through this work will be instrumental in guiding future research and experimentation in the field of music source separation, contributing to the ongoing development of machine learning techniques in this challenging domain.

References

Nobutaka Ito and Masashi Sugiyama. 2023. Audio signal enhancement with learning from positive and unlabeled data. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

Yi Luo and Jianwei Yu. 2023. Music source separation with band-split rnn. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1893–1901.