โปรแกรมแยกเสียงดนตรี
Separation musical

นายยศพล เกตุดิษฐ์ 6452300105

การค้นคว้าอิสระเสนอคณะวิศวกรรมศาสตร์และเทคโนโลยีสถาบันการจัดการปัญญาภิวัฒน์

เพื่อเป็นส่วนหนึ่งของการศึกษา

หลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์และปัญญาประดิษฐ์

พ.ศ. ๒๕๖๗

Music separation program

Separation musical

Mister Yodsaphon Keddid 6452300105

A Senior Project Submitted in Partial Fulfilment of the Requirements

For the Degree of Bachelor of Computer Engineering

Faculty of Engineering and Technology

Academic Year 2025

เรื่อง	โปรแกรมแยกเสียงดนตรี
	Separation musical
โดย	นายยศพล เกตุดิษฐ์
คณะ	วิศวกรรมศาสตร์และเทคโนโลยี
สาขาวิชา	สาขาวิชาวิศวกรรมคอมพิวเตอร์และปัญญาประดิษฐ์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร. วีรวุฒิ ทัฬหิกรรม

ได้รับการอนุมัติเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร ปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

..... คณบดีคณะวิศวกรรมศาสตร์และเทคโนโลยี

(รศ.ดร.พิสิษฐ์ ชาญเกียรติก้อง)

..... ประธานกรรมการ

(รศ.ดร.ปริญญา สงวนสัตย์)

..... กรรมการ

(ผศ.ดร.อดิศร แขกซอง)

..... กรรมการ

(ผศ.ติณณภพ ดินดำ)

..... หัวหน้าสาขาวิชาวิศวกรรมคอมพิวเตอร์

(รศ.ดร.ปริญญา สงวนสัตย์)

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของการศึกษารายวิชา 1321306 โครงงานวิศวกรรมคอมพิวเตอร์และปัญญาประดิษฐ์ ซึ่งมีวัตถุประสงค์เพื่อศึกษาค้นคว้าและพัฒนาโครงงานในสาขาวิชาวิศวกรรมคอมพิวเตอร์และปัญญาประดิษฐ์ โดยมุ่งเน้นการประยุกต์ใช้ความรู้ทางเทคนิคและวิทยาการคอมพิวเตอร์ในการแก้ปัญหาเชิงปฏิบัติ

ในการจัดทำรายงานนี้ ผู้จัดทำได้ทำการศึกษาค้นคว้าและรวบรวมข้อมูลจากแหล่งข้อมูลต่าง ๆ รวมถึงเอกสารวิชาการ งานวิจัยที่เกี่ยวข้อง และแหล่งข้อมูลออนไลน์ เพื่อให้ได้ข้อมูลที ่ครอบคลุมและเชื่อถือได้ นอกจากนี้ยังได้ทำการทดสอบและประเมินผลการทดลองต่างๆ เพื่อให้ได้ผลลัพธ์ที่น่าเชื่อถือและมีประโยชน์ต่อ การศึกษาวิจัยในอนาคต

ผู้จัดทำหวังเป็นอย่างยิ่งว่ารายงานฉบับนี้จะเป็นประโยชน์แก่ผู้ที่สนใจและสามารถนำไปใช้ประโยชน์ได้ตาม สมควร หากมีข้อเสนอแนะหรือติชมใด ๆ ผู้จัดทำยินดีรับฟังเพื่อนำไปปรับปรุงในการจัดทำรายงานครั้งต่อไป

ยศพล เกตุดิษฐ์

กิตติกรรมประกาศ

การค้นคว้าอิสระ เรื่อง โปรแกรมแยกเสียงดนตรี ฉบับนี้สำเร็จลงได้ด้วยดี เนื่องจากการให้รับความช่วยเหลือ และคำแนะนำจาก ผู้ช่วยศาสตราจารย์ ดร.วีรวุฒิ ทัฬหิกรรม ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงานวิจัย ซึ่งเป็นผู้ให้ความรู้ คำแนะนำ แนวทางการศึกษา ตลอดจนให้ความกรุณา ในการตรวจทานแก้ไขโครงงานวิจัยนี้จนทำให้ประสบความสำเร็จอย่างสมบูรณ์ และขอบกราบขอบพระคุณ คณะกรรมการสอบโครงงานวิจัย ผศ.ดร. อดิศร แขกซอง ดร.ติณณภพ ดินดำ และรองศาสตราจารย์ ดร. ปริญญา สงวนสัตย์ที่กรุณาให้คำแนะนำตลอดจนแก้ไขข้อบกพร่องต่างๆของเนื้อหาโครงงานวิจัยจนเสร็จสมบูรณ์ ทางทีมผู้จัดทำโครงงาน จึงต้องขอขอบพระคุณอย่างสูง มา ณ โอกาสนี้

สุดท้ายนี้ ทางผู้จัดทำหวังเป็นอย่างยิ่งว่า ความรู้จากโครงงานฉบับนี้จะเป็นประโยชน์ต่อผู้ที่สนใจ ไม่มากก็ น้อย หากมีส่วนใดที่บกพร่องหรือมีความผิดพลาดประการใด ทางผู้จัดทำต้องขออภัยมา ณ ที่นี้ด้วย

ยศพล เกตุดิษฐ์

เรื่อง	โปรแกรมแยกเสียงดนตรี
โดย	นายยศพล เกตุดิษฐ์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.วีรวุฒิ ทัฬหิกรรม
คณะ	วิศวกรรมศาสตร์และเทคโนโลยี
สาขา	วิชาวิศวกรรมคอมพิวเตอร์และปัญญาประดิษฐ์
ปีการศึกษา	พ.ศ. 2567

บทคัดย่อ

โปรเจกต์นี้เป็นระบบที่พัฒนาขึ้นเพื่อแยกเสียงร้องและเสียงเครื่องดนตรีต่าง ๆ จากไฟล์เสียงผสม (Mix Audio) โดยอาศัยเทคโนโลยีการเรียนรู้เชิงลึก (Deep Learning) และสถาปัตยกรรมโครงข่ายประสาทเทียมแบบ U-Net ในการประมวลผลและวิเคราะห์ข้อมูลเสียงที่ได้จากการแปลงเป็น Mel-Spectrogram ระบบจะทำการเรียนรู้และแยกประเภทของเสียงออกเป็น 4 ประเภท ได้แก่ เสียงร้อง (Vocals), กลอง (Drums), เบส (Bass) และเสียงเครื่องดนตรีอื่น ๆ (Other Instruments)

ระบบถูกออกแบบมาให้สามารถทำงานได้แบบอัตโนมัติ โดยรับไฟล์เสียงจากผู้ใช้ในรูปแบบ MP3 หรือ WAV ผ่านเว็บแอปพลิเคชัน จากนั้นทำการประมวลผลและแยกเสียงแต่ละประเภทออกมาเป็นไฟล์เสียงที่แยกกันได้ เพื่อให้ผู้ใช้สามารถนำไฟล์เสียงที่แยกได้ไปใช้ในการมิกซ์เสียงหรือการประมวลผลอื่น ๆ ได้ตามต้องการ

โดยสรุป โปรเจกต์นี้มุ่งเน้นในการพัฒนาระบบต้นแบบสำหรับแยกเสียงร้องและเสียงเครื่องดนตรีออกจากกันอย่างแม่นยำและสะดวกต่อการใช้งานในชีวิตประจำวัน ทั้งนี้ ระบบที่พัฒนาขึ้นในโครงการนี้เป็นเพียงต้นแบบสำหรับการศึกษาและวิจัย ไม่ใช่ระบบที่นำไปใช้งานจริงในเชิงพาณิชย์

Title	Separation musical
Author's Name	Mr.Yodsaphon Keddid
Advisor	Assistant Professor Dr. Weerawut Thanhikam
Degree	Engineering and Technology Program in computer engineering
Academic Year	2024

Abstract

This project presents a system developed to separate vocals and various musical instruments from mixed audio files by leveraging deep learning technology, specifically the U-Net neural network architecture. The system processes and analyzes audio data by converting it into Mel-Spectrogram representations, allowing it to learn and separate the audio into four distinct categories: vocals, drums, bass, and other instruments.

The system is designed to operate automatically, allowing users to upload audio files in MP3 or WAV formats through a web application. It then processes the input and separates each type of sound into individual audio files, enabling users to further utilize the separated tracks for music production or other processing purposes.

In summary, this project focuses on developing a prototype system for the separation of vocals and musical instruments with accuracy and ease of use in daily life. The developed system is intended for educational and research purposes only and is not intended for commercial deployment.

สารบัญ

สารบัญ(ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ในปัจจุบัน การสร้างและจัดการเสียงดนตรีมีบทบาทสำคัญอย่างมากในวงการบันเทิง การศึกษา และสื่อสารมวลชน ความต้องการเทคโนโลยีที่สามารถแยกเสียงร้องออกจากเครื่องดนตรีประเภทต่าง ๆ เช่น กลอง เบส และเครื่องดนตรีอื่น ๆ ได้อย่างมีประสิทธิภาพจึงเพิ่มขึ้นอย่างรวดเร็ว ไม่ว่าจะเพื่อการรีมิกซ์เพลง การศึกษาดนตรี การพัฒนาระบบคาราโอเกะ หรือการสร้างสื่อการสอน

อย่างไรก็ตาม กระบวนการแยกเสียงจากไฟล์เสียงผสม (Mixed Audio) ยังคงเป็นเรื่องที่ท้าทาย แม้จะมีการใช้เทคนิคการประมวลผลสัญญาณแบบดั้งเดิม เช่น Spectral Subtraction หรือ Non-negative Matrix Factorization (NMF) แต่ก็ยังมีข้อจำกัดในด้านความแม่นยำ และไม่สามารถจัดการกับเสียงที่มีการทับซ้อนกันอย่างซับซ้อนได้อย่างมีประสิทธิภาพ

การนำเทคโนโลยี ปัญญาประดิษฐ์ (Artificial Intelligence: AI) และ การเรียนรู้ของเครื่อง (Machine Learning: ML) โดยเฉพาะ การเรียนรู้เชิงลึก (Deep Learning: DL) เข้ามาประยุกต์ใช้ ได้กลายเป็นแนวทางที่ได้รับความนิยมในงานแยกเสียงดนตรี ตัวอย่างเช่นการใช้สถาปัตยกรรมโครงข่ายประสาทเทียมแบบ U-Net ซึ่งสามารถประมวลผลข้อมูล Mel-Spectrogram ของเสียงได้อย่างแม่นยำ และมีประสิทธิภาพในการแยกองค์ประกอบเสียงที่มีความซับซ้อนสูง

โครงการนี้มีเป้าหมายในการพัฒนาระบบต้นแบบที่สามารถแยกเสียงร้องและเครื่องดนตรีประเภทต่าง ๆ ออกจากไฟล์เสียงผสมได้ เช่น MP3 หรือ WAV แล้วรับไฟล์เสียงที่แยกส่วนสำเร็จได้อย่างสะดวก รวดเร็ว และมีประสิทธิภาพ ทั้งยังมุ่งหวังให้เป็นต้นแบบสำหรับต่อยอดในงานวิจัยหรือการพัฒนาเชิงพาณิชย์ในอนาคต

1.2 วัตถุประสงค์

1.2.1 เพื่อพัฒนาโมเดล Deep Learning (U-Net) สำหรับแยกเสียงร้องและเสียงเครื่องดนตรีจากไฟล์เสียงผสม

1.2.2 เพื่อแปลงสัญญาณเสียงเป็นข้อมูล Mel-Spectrogram และนำมาใช้ในการฝึกสอนโมเดล

1.2.3 เพื่อพัฒนา Web Application ที่สามารถอัปโหลดไฟล์เสียงและแยกไฟล์เสียงอัตโนมัติ

1.2.4 เพื่อประเมินประสิทธิภาพของโมเดลแยกเสียงผ่านการวัดค่าความผิดพลาด เช่น Mean Squared Error (MSE) และ Mean Absolute Error (MAE)

1.2.5 เพื่อสร้างต้นแบบที่สามารถนำไปประยุกต์ใช้ได้จริงในสภาพแวดล้อมต่าง ๆ

1.3 สมมติฐานการวิจัย

1.3.1 โมเดล U-Net ที่ฝึกด้วยข้อมูล Mel-Spectrogram จะสามารถแยกเสียงร้องและเสียงเครื่องดนตรีได้อย่างแม่นยำ

1.3.2 การใช้ข้อมูล Mel-Spectrogram ช่วยเพิ่มประสิทธิภาพในการเรียนรู้ของโมเดลเมื่อเทียบกับการใช้ข้อมูล waveform ดิบ

1.4 ขอบเขตของการวิจัย

1.4.1 ขอบเขตด้านข้อมูล

1.4.1.1 ใช้ชุดข้อมูลเสียงจาก MUSDB18 ซึ่งประกอบด้วยไฟล์เสียงแบบ Stem (vocals, drums, bass, others) จำนวน 100 ไฟล์สำหรับการฝึก และ 50 ไฟล์สำหรับการทดสอบ

1.4.1.2 รองรับไฟล์เสียงขาเข้าในรูปแบบ MP3 และ WAV

1.4.2 ขอบเขตด้านเทคนิค

1.4.2.1 ใช้เทคนิคการแปลงเสียงเป็น Mel-Spectrogram

1.4.2.2 ใช้สถาปัตยกรรม U-Net ขนาดใหญ่ (Large U-Net) ในการประมวลผล

1.4.2.3 ใช้ TensorFlow/Keras สำหรับการพัฒนาโมเดล

1.4.3 ขอบเขตด้านระบบ

1.4.3.1 ระบบต้นแบบจะมีความสามารถพื้นฐานในการแยกเสียง แต่ไม่ได้รับรองความแม่นยำระดับเชิงพาณิชย์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ได้ต้นแบบโมเดลแยกเสียงที่มีประสิทธิภาพสำหรับงานด้านดนตรี การศึกษา หรือความบันเทิง

1.5.2 ได้ระบบที่ช่วยลดขั้นตอนการแยกเสียงด้วยมือ และเพิ่มความแม่นยำในการประมวลผล

1.5.3 ได้องค์ความรู้ใหม่ด้าน Deep Learning for Audio Processing เพื่อต่อยอดการวิจัยในอนาคต

1.5.4 สร้างต้นแบบสำหรับพัฒนาระบบอัตโนมัติด้านดนตรี หรือเสียงพูดอื่น ๆ ในอนาคต

1.6 นิยามคำศัพท์เฉพาะ

1.6.1 Separation: การแยกส่วนสัญญาณเสียงแต่ละประเภทออกจากสัญญาณเสียงผสม

1.6.2 Mel-Spectrogram: การแทนข้อมูลเสียงในรูปแบบความถี่-เวลา บนมาตรฐาน Mel scale เพื่อสะท้อนการได้ยินของมนุษย์

1.6.3 U-Net: โครงข่ายประสาทเทียมแบบ Encoder-Decoder พร้อม Skip Connections เหมาะสำหรับงานแยกเสียงและแยกภาพ

1.6.4 Deep Learning (DL): การเรียนรู้ของเครื่องในระดับโครงข่ายประสาทหลายชั้น (Neural Networks) ที่ซับซ้อน

1.6.5 MUSDB18: ชุดข้อมูลเสียงที่จัดทำขึ้นเพื่อการวิจัยด้านแยกเสียงในเพลง

บทที่ 2
วรรณกรรมที่เกี่ยวข้อง

ในการจัดทำโครงงานครั้งนี้ คณะผู้จัดทำมีวัตถุประสงค์ในการเพิ่มความรู้ ความเข้าใจ และพัฒนาระบบแยกเสียงดนตรีออกจากกันด้วยเทคโนโลยีปัญญาประดิษฐ์ (Artificial Intelligence: AI) เพื่อให้สามารถแยกเสียงร้อง เสียงกลอง เสียงเบส และเสียงเครื่องดนตรีอื่น ๆ ออกจากไฟล์เสียงผสมได้อย่างแม่นยำและรวดเร็ว โดยเน้นการประมวลผลในรูปแบบ Mel-Spectrogram และการประยุกต์ใช้โครงข่ายประสาทเทียมแบบ U-Net

คณะผู้จัดทำได้ดำเนินการศึกษาและค้นคว้าข้อมูลที่เกี่ยวข้องอย่างละเอียด เพื่อให้เกิดความเข้าใจที่ลึกซึ้งในเรื่องเทคโนโลยีที่จำเป็น การประมวลผลสัญญาณเสียง การออกแบบสถาปัตยกรรมโครงข่ายประสาทเทียม

ข้อมูลที่เกี่ยวข้องในโครงงานครั้งนี้ประกอบด้วย แนวคิดพื้นฐานเกี่ยวกับการแยกเสียงดนตรี การแปลงสัญญาณเสียงให้อยู่ในรูปแบบที่เหมาะสมกับการประมวลผลด้วย AI เทคนิคการสร้างและฝึกโมเดล U-Net สำหรับงานแยกเสียง รวมถึงงานวิจัยที่เกี่ยวข้อง โดยมีรายละเอียดดังต่อไปนี้

2.1 แนวคิดเกี่ยวกับการแยกเสียง (Audio Source Separation)

การแยกเสียง (Audio Source Separation) คือกระบวนการแยกองค์ประกอบเสียงต่าง ๆ เช่น เสียงร้อง เสียงกลอง เสียงเบส หรือเครื่องดนตรีอื่น ๆ ออกจากสัญญาณเสียงผสม (Mixed Audio) ซึ่งในสัญญาณผสมดังกล่าว อาจมีเสียงหลายแหล่งซ้อนกันอยู่ การแยกเสียงออกมาได้อย่างแม่นยำมีความสำคัญอย่างมากในหลากหลายสาขา เช่น การทำคาราโอเกะ การผลิตเพลง การวิเคราะห์เสียงทางการแพทย์ และการประมวลผลสัญญาณเสียงพูด

เดิมทีเทคนิคการแยกเสียงจะอาศัยวิธีการแบบดั้งเดิม เช่น

Non-negative Matrix Factorization (NMF): การแยกสัญญาณโดยสมมติว่า spectrogram ของเสียงสามารถแยกได้เป็นผลคูณของเมทริกซ์สองตัวที่ไม่เป็นลบ

Independent Component Analysis (ICA): การแยกสัญญาณที่เป็นอิสระจากกันในลักษณะสถิติ

แต่ด้วยความซับซ้อนของเสียงดนตรีจริง ๆ ที่มีการทับซ้อนของความถี่และเวลา ทำให้เทคนิคดั้งเดิมไม่สามารถแยกเสียงได้ดีเท่าที่ควร จึงมีการพัฒนาเทคนิคที่ใช้ Deep Learning ขึ้นมา เพื่อให้การแยกเสียงมีความแม่นยำสูงขึ้น

2.2 เทคโนโลยีการแปลงสัญญาณเสียง (Mel-Spectrogram)

การแปลงสัญญาณเสียงให้อยู่ในรูปแบบที่เหมาะสมกับการป้อนเข้าสู่โมเดล Deep Learning เป็นขั้นตอนที่สำคัญ

Mel-Spectrogram คือการแปลงสัญญาณเสียงให้อยู่ในรูป Time-Frequency ที่สอดคล้องกับการรับรู้ของหูมนุษย์ โดยเน้นการแจกแจงความถี่แบบ Mel Scale ซึ่งมีการกระจายความถี่ถี่ขึ้นในช่วงความถี่ต่ำและห่างขึ้นในช่วงความถี่สูง

ข้อดีของ Mel-Spectrogram คือ

สามารถแทนคุณสมบัติของเสียงได้อย่างมีประสิทธิภาพ

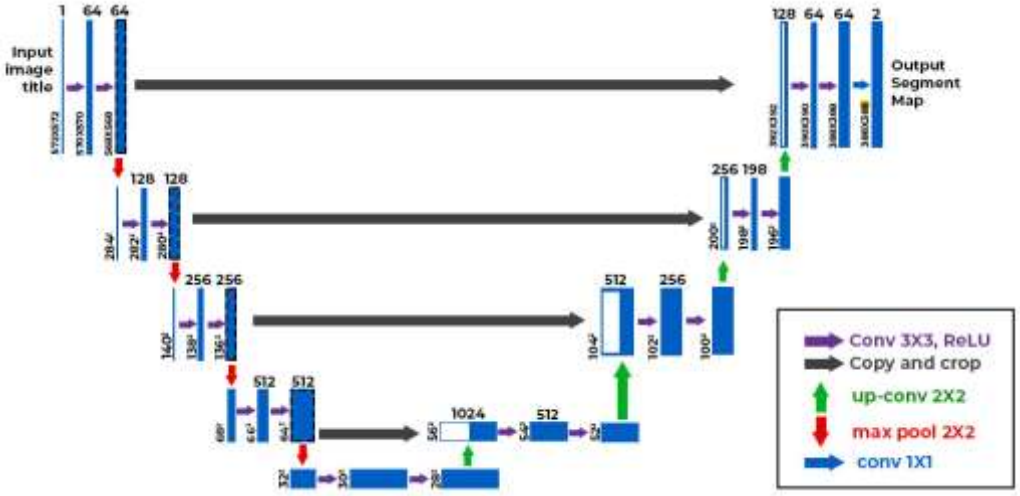
เหมาะสำหรับการป้อนเข้าสู่โมเดล Deep Learning เช่น CNN หรือ U-Net

ลดขนาดข้อมูลลงจาก waveform ดิบ ทำให้ฝึกโมเดลได้เร็วขึ้นและเสถียรกว่า

สรุป: การใช้ Mel-Spectrogram เป็นแนวทางมาตรฐานสำหรับงานแยกเสียงในปัจจุบัน

2.3 สถาปัตยกรรมโครงข่ายประสาทเทียม U-Net

การแยกแหล่งกำเนิดเสียงออกจากกันจากสัญญาณเสียงผสมเป็นโจทย์ที่มีความซับซ้อน เนื่องจากแหล่งเสียงแต่ละประเภท เช่น เสียงร้อง เสียงเบส หรือเสียงกลอง มักมีการทับซ้อนกันในทั้งเชิงเวลาและความถี่ ทำให้การวิเคราะห์ด้วยโมเดลทั่วไปที่ไม่สามารถจับความสัมพันธ์ในเชิงโครงสร้างและตำแหน่งของข้อมูลได้อาจให้ผลลัพธ์ที่ไม่แม่นยำ ดังนั้น ในโครงการนี้จึงเลือกใช้สถาปัตยกรรมโครงข่ายประสาทเทียมแบบ U-Net ซึ่งมีโครงสร้างที่เหมาะสมกับการวิเคราะห์ข้อมูลภาพ และสามารถนำมาประยุกต์ใช้กับข้อมูลเสียงในรูปของ Mel-Spectrogram ได้อย่างมีประสิทธิภาพ



ภาพที่ 2.1 โครงสร้างสถาปัตยกรรม U-Net

สถาปัตยกรรม U-Net มีต้นแบบจากงานทางด้านการวิเคราะห์ภาพทางการแพทย์ โดยมีลักษณะเป็นโครงข่ายแบบ encoder–decoder ซึ่งประกอบด้วยสองส่วนหลัก คือ ส่วน encoder ทำหน้าที่บีบอัดข้อมูล (downsampling) เพื่อดึงคุณลักษณะสำคัญจากข้อมูลเสียง และส่วน decoder ทำหน้าที่ขยายข้อมูลกลับคืน (upsampling) เพื่อสร้างผลลัพธ์ที่มีขนาดเท่ากับ input โดยในแต่ละระดับของการขยายขนาดจะมีการเชื่อมโยงกับชั้นที่มีขนาดเท่ากันในฝั่ง encoder ผ่านกลไกที่เรียกว่า skip connection ซึ่งช่วยให้โมเดลสามารถถ่ายโอนข้อมูลรายละเอียดตำแหน่งจากชั้นตื้นไปยังชั้นลึกได้โดยตรง

2.4 งานวิจัยที่เกี่ยวข้อง

2.4.1 Open-Unmix: A Reference Implementation for Music Source Separation (Stöter et al., 2019)

นำเสนอโมเดล Open-Unmix ซึ่งใช้โครงข่าย RNN สำหรับแยกเสียงร้อง กลอง เบส และเครื่องดนตรีอื่น ๆ

ข้อดีคือ โมเดลมีขนาดเล็กและฝึกได้ง่าย แต่ข้อเสียคือมีข้อจำกัดด้านความแม่นยำเมื่อแยกเสียงที่ซับซ้อนมาก ๆ

2.4.2 Spleeter: A Fast and Efficient Music Source Separation Tool (Deezer Research)

Spleeter ใช้สถาปัตยกรรม U-Net ขนาดเล็กถึงขนาดกลาง แยกเสียงได้รวดเร็วและแม่นยำ

รองรับการแยก 2-stem (เสียงร้อง/ดนตรี) และ 4-stem (เสียงร้อง/เบส/กลอง/อื่น ๆ) ได้ดี

เป็นแรงบันดาลใจสำคัญในการเลือกใช้ U-Net ในโครงการนี้

2.4.3 Demucs: Deep Extractor for Music Sources (Défossez et al., 2019)

ใช้โมเดลที่ทำงานบนสัญญาณเสียง waveform ตรง ๆ โดยไม่แปลงเป็น spectrogram

แม้จะได้คุณภาพเสียงดีมาก แต่ใช้ทรัพยากรสูงกว่าการทำงานบน Mel-Spectrogram

บทที่ 3
วิธีการดำเนินการวิจัย

ในการดำเนินโครงงาน "โปรแกรมแยกเสียงดนตรี (Separation Musical)" คณะผู้จัดทำได้ดำเนินการวางแผนการพัฒนาอย่างเป็นระบบ โดยแบ่งการทำงานออกเป็นขั้นตอนที่ชัดเจน เพื่อให้สามารถบรรลุวัตถุประสงค์ของโครงงานได้อย่างมีประสิทธิภาพ ดังนี้

3.1 กำหนดสภาพแวดล้อมของการพัฒนา

3.1.1 ภาษาโปรแกรม (Programming Languages)

3.1.1.1 ภาษาไพธอน (Python) เป็นภาษาหลักที่ใช้ในการพัฒนาโปรแกรม เนื่องจากมีไลบรารีด้านการประมวลผลเสียง และ Machine Learning ที่มีประสิทธิภาพสูง และได้รับความนิยมอย่างแพร่หลายในวงการ Deep Learning

3.1.2 ไลบรารีที่ใช้ (Libraries)

3.1.2.1 Os ทำหน้าที่จัดระเบียบเส้นทางไฟล์และโฟลเดอร์ เช่น การตรวจสอบว่าไฟล์หรือโฟลเดอร์มีอยู่หรือไม่ การสร้างโฟลเดอร์ใหม่ และการจัดเส้นทางของไฟล์ให้ถูกต้อง ไลบรารีนี้ถูกใช้ในเกือบทุกไฟล์ในโปรเจกต์เพื่อควบคุมการอ่านและเขียนไฟล์เสียงหรือข้อมูล Mel-Spectrogram

3.1.2.2 NumPy ใช้สำหรับคำนวณเชิงตัวเลขและการจัดการข้อมูลในรูปแบบของอาร์เรย์ เช่น การแพดข้อมูลให้มีขนาดเท่ากัน การรวมข้อมูลหลายแทร็ก และการจัดรูปข้อมูลให้อยู่ในรูปแบบที่เหมาะสมกับการป้อนเข้าโมเดล deep learning เช่น การตัดแบ่งข้อมูล การแพดให้ขนาดเท่ากัน และการจัดให้ข้อมูลอยู่ในรูปแบบที่โมเดลสามารถประมวลผลได้

3.1.2.3 Librosa ใช้สำหรับการประมวลผลสัญญาณเสียง ในโปรเจกต์นี้ใช้สำหรับโหลดไฟล์เสียง .wav การแปลง waveform ให้กลายเป็น Mel-Spectrogram การปรับสเกลพลังงานของเสียงให้เหมาะสม และการแปลงกลับจาก Spectrogram เป็น waveform อีกทั้งยังใช้วาดกราฟแสดง waveform เพื่อนำมาเปรียบเทียบผลการแยกเสียง

3.1.2.4 TensorFlow ใช้สำหรับสร้างและฝึกโมเดล deep learning โดยเฉพาะผ่านโมดูล keras ที่อำนวยความสะดวกในการสร้างเลเยอร์ต่าง ๆ ของโมเดล เช่น Convolution, Pooling, UpSampling และ Concatenate นอกจากนี้ยังใช้สำหรับกำหนดค่าการฝึก การเลือก optimizer การคำนวณ loss และการใช้ callback เพื่อควบคุมการหยุดฝึกอัตโนมัติเมื่อค่า validation loss ไม่ดีขึ้น

3.1.2.5 matplotlib.pyplot ใช้สำหรับสร้างกราฟแสดงผล เช่น กราฟ loss และ MAE ระหว่างการฝึกโมเดล ซึ่งช่วยให้สามารถวิเคราะห์พฤติกรรมของโมเดลในแต่ละ epoch ได้ รวมถึงใช้วาดกราฟ waveform ของเสียงต้นฉบับและเสียงที่แยกออกมาได้ เพื่อให้สามารถประเมินความเหมือนได้ด้วยสายตา

3.1.2.6 Soundfile ไว้สำหรับอ่านและเขียนไฟล์เสียง .wav โดยให้คุณภาพเสียงสูงและมีประสิทธิภาพในการประมวลผลมากกว่าไลบรารีอื่นในบางกรณี ใช้บันทึกไฟล์เสียงที่ได้จากการแยกเสียง และใช้โหลดแทร็กเสียงจากชุดข้อมูลเพื่อเตรียมใช้งาน

3.1.2.7 sklearn.metrics โดยเฉพาะฟังก์ชัน mean_squared_error ถูกใช้เพื่อวัดความคลาดเคลื่อนระหว่างเสียงต้นฉบับกับเสียงที่โมเดลสร้างขึ้น ซึ่งเป็นการประเมินในเชิงตัวเลขว่าผลลัพธ์ที่ได้มีความแม่นยำมากน้อยเพียงใด

3.1.2.8 scipy.spatial.distance โดยใช้ฟังก์ชัน cosine เพื่อใช้คำนวณค่าความเหมือนเชิงเวกเตอร์หรือ Cosine Similarity ระหว่างเสียงต้นฉบับกับเสียงที่ได้จากการแยก ซึ่งช่วยให้เข้าใจความใกล้เคียงกันของลักษณะ waveform ได้อย่างชัดเจน

3.1.2.9 stempeg เป็นไลบรารีเฉพาะทางที่ใช้ในการอ่านไฟล์ .stem.mp4 ซึ่งเป็นฟอร์แมตที่ใช้ในชุดข้อมูล MUSDB18 ที่เก็บแทร็กเสียงหลายชนิดไว้ในไฟล์เดียว ไลบรารีนี้จึงถูกใช้เพื่อแยกแทร็กต่าง ๆ ออกมาเป็นไฟล์ .wav สำหรับนำไปใช้ในขั้นตอนการเตรียมข้อมูล

3.1.2.10 tqdm เป็นไลบรารีสำหรับแสดงแถบความคืบหน้า (progress bar) ขณะรันลูปในขั้นตอนต่าง ๆ เช่น การแปลงไฟล์เสียงเป็น Mel-Spectrogram หรือการแยกไฟล์จากชุดข้อมูล เพื่อให้ผู้ใช้งานทราบว่าขั้นตอนต่าง ๆ ใช้เวลานานเท่าใดและดำเนินการถึงไหนแล้ว

3.1.3 เครื่องมือสำหรับพัฒนา (Development Tools)

3.1.3.1 Google Colaboratory หรือที่เรียกสั้น ๆ ว่า Google Colab เป็นเครื่องมือพัฒนาออนไลน์ที่ให้บริการฟรีจาก Google โดยมีลักษณะคล้ายกับ Jupyter Notebook ซึ่งทำให้ผู้ใช้สามารถเขียนโค้ดภาษา Python และรันบนเซิร์ฟเวอร์ของ Google ได้โดยไม่ต้องติดตั้งอะไรในเครื่องคอมพิวเตอร์ของตนเอง ในโครงการนี้ Google Colab ถูกนำมาใช้ในขั้นตอนการพัฒนาและทดสอบโค้ดเบื้องต้น รวมถึงการประมวลผลเบื้องหลังด้วย GPU หรือ TPU สำหรับงานที่ใช้การคำนวณจำนวนมาก เช่น การฝึกโมเดล Deep Learning โดยเฉพาะช่วงที่ต้องโหลดข้อมูลจาก Google Drive และแปลงเสียงเป็น Mel-Spectrogram ซึ่งต้องใช้เวลานานและกินทรัพยากรสูง Google Colab จึงเป็นเครื่องมือหลักที่ช่วยให้การทดลองและพัฒนาโมเดลสามารถทำได้สะดวกและมีประสิทธิภาพโดยไม่ต้องพึ่งพาคอมพิวเตอร์ส่วนตัว

3.1.3.2 PyCharm เป็นโปรแกรมพัฒนาไอดีอี (IDE) ที่ออกแบบมาสำหรับภาษา Python โดยเฉพาะ ซึ่งมีฟีเจอร์ที่ครอบคลุมการเขียนโค้ด การดีบัก การจัดการไฟล์ และการสร้างสภาพแวดล้อมในการทำงานที่เป็นระบบและมืออาชีพ ในโปรเจกต์นี้ PyCharm ถูกใช้ในการพัฒนาโค้ดหลักที่ทำงานในเครื่องคอมพิวเตอร์ของผู้พัฒนาโดยตรง โดยเฉพาะในขั้นตอนการฝึกโมเดล U-Net ที่ต้องควบคุมไฟล์ข้อมูลจำนวนมากและมีโครงสร้างโปรเจกต์ที่ซับซ้อน PyCharm ช่วยให้การเขียนโค้ด และ ช่วยในการจัดการไฟล์ .py ในแต่ละชุด เช่น train_unet.py หรือ predict_and_evaluate_fulltrack_save_results_rms.py เป็นไปอย่างมีระเบียบ ง่ายต่อการ

ตรวจสอบและปรับปรุง นอกจากนี้ยังมีระบบเชื่อมต่อกับ Virtual Environment และระบบจัดการไฟล์ที่สนับสนุนการทำงานร่วมกับ TensorFlow และไลบรารีอื่น ๆ ได้อย่างมีประสิทธิภาพ

3.1.3.3 Google Drive เป็นบริการพื้นที่จัดเก็บข้อมูลแบบออนไลน์ (Cloud Storage) จาก Google ซึ่งถูกใช้เป็นคลังเก็บข้อมูลกลางของโปรเจกต์ โดยเฉพาะข้อมูลชุด MUSDB18 และข้อมูล Mel-Spectrogram ที่มีขนาดใหญ่และต้องใช้งานซ้ำหลายครั้งในการฝึกและทดสอบโมเดล การใช้ Google Drive ทำให้สามารถเก็บข้อมูลทั้งหมดไว้ในระบบคลาวด์และเข้าถึงจาก Google Colab ได้โดยตรงผ่านการเชื่อมต่อไดรฟ์ ซึ่งช่วยลดภาระของฮาร์ดดิสก์ภายในเครื่องของผู้พัฒนา นอกจากนี้ยังสามารถใช้ Google Drive ในการสำรองโมเดลที่ฝึกแล้วหรือผลลัพธ์การแยกเสียงที่ได้จากโมเดลเพื่อนำไปประเมินผลหรือใช้งานต่อในภายหลังได้อย่างมีประสิทธิภาพและปลอดภัย

3.1.4 เครื่องมือสำหรับทดสอบ (Testing Tools)

3.1.4.1 Sound Similar Free เป็นเครื่องมือสำหรับวิเคราะห์และเปรียบเทียบความเหมือนของสัญญาณเสียงโดยเฉพาะ ซึ่งสามารถใช้ตรวจสอบความคล้ายคลึงของไฟล์เสียงสองไฟล์ในเชิงคณิตศาสตร์ โดยมีการคำนวณค่าความเหมือน เช่น ค่าความคลาดเคลื่อนหรือการเปรียบเทียบลักษณะของคลื่นเสียงอย่างละเอียด ในโครงการนี้ Sound Similar Free ถูกใช้เพื่อตรวจสอบประสิทธิภาพของโมเดลแยกเสียงที่พัฒนาขึ้น โดยการนำเสียงที่ถูกแยกออกมาจากโมเดลไปเปรียบเทียบกับเสียงต้นฉบับของแต่ละแทร็ก เช่น เสียงร้อง เสียงกลอง เสียงเบส และเสียงเครื่องดนตรีอื่น ๆ ซึ่งช่วยให้สามารถประเมินได้ว่าโมเดลสามารถแยกเสียงได้ดีเพียงใดในมุมมองของผู้ใช้ปลายทาง โดยไม่ต้องอาศัยการคำนวณทางเทคนิคที่ซับซ้อนเหมือนในโค้ด Python

3.1.4.1 Audacity เป็นซอฟต์แวร์โอเพนซอร์สสำหรับบันทึกและแก้ไขเสียงที่ได้รับความนิยมสูง โดยมีความสามารถในการตัดต่อ ปรับแต่ง และแสดงภาพคลื่นเสียงอย่างละเอียด ในโปรเจกต์นี้ Audacity ถูกนำมาใช้เพื่อฟังและตรวจสอบไฟล์เสียงที่ได้จากการแยกโดยโมเดล ซึ่งช่วยให้สามารถประเมินคุณภาพเสียงในเชิงฟังจริงได้ทันที เช่น ตรวจสอบว่าเสียงที่ได้มีความผิดเพี้ยนหรือมีเสียงรบกวนอยู่หรือไม่ รวมถึงสามารถซ้อนแทร็กเสียงเพื่อเปรียบเทียบระหว่างเสียงต้นฉบับและเสียงที่แยกออกมาได้อย่างง่ายดาย การใช้ Audacity จึงเป็นส่วนเสริมที่สำคัญในการตรวจสอบคุณภาพผลลัพธ์จากโมเดลในมิติของการรับรู้ด้วยประสาทหูของมนุษย์นอกเหนือจากการวัดผลด้วยค่าทางตัวเลขเพียงอย่างเดียว

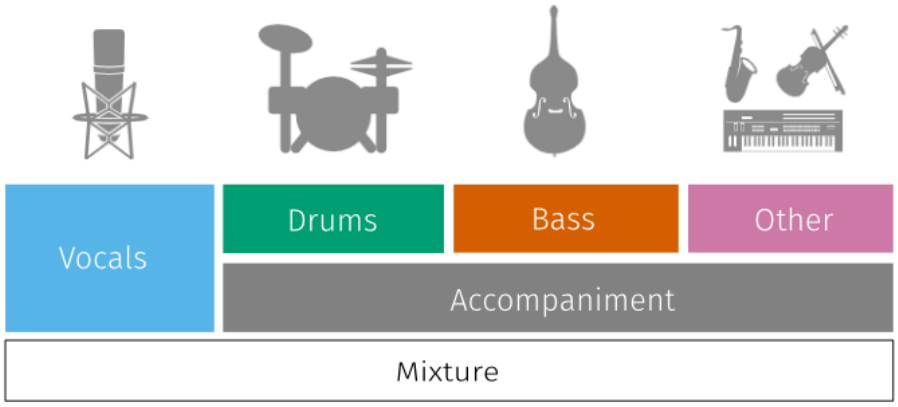
3.2 การเตรียมชุดข้อมูล

ในส่วนนี้จะอธิบายถึงขั้นตอนการเตรียมข้อมูลและการสร้างชุดงานวิจัยนี้ โดยผู้วิจัยได้ทำการจัดเตรียมและรวบรวมข้อมูลไฟล์เสียง ซึ่งเป็นกระบวนการที่สำคัญในการสร้างชุดเครื่องมือสำหรับการทำโมเดลแยกเสียง โดยมีการเก็บ รวบรวมและจัดเตรียมข้อมูลดังต่อไปนี้

3.2.1 การเลือกชุดข้อมูล (Dataset Selection)

เป็นขั้นตอนสำคัญที่มีผลโดยตรงต่อความสามารถในการเรียนรู้ของโมเดลปัญญาประดิษฐ์ โดยเฉพาะอย่างยิ่งในโครงการนี้ซึ่งมีเป้าหมายในการพัฒนาโมเดลแยกเสียงดนตรีออกเป็นแทร็กย่อย เช่น เสียงร้อง เสียงกลอง เสียงเบส และเสียงเครื่องดนตรีอื่น ๆ การเลือกชุดข้อมูลที่มีคุณภาพ ครอบคลุม และมีการจัดระเบียบที่ดีจึงเป็นพื้นฐานที่สำคัญต่อความสำเร็จของระบบที่พัฒนา

ในโครงการนี้ ผู้จัดทำเลือกใช้ชุดข้อมูล MUSDB18 ซึ่งเป็นชุดข้อมูลมาตรฐานที่ใช้กันอย่างแพร่หลายในงานวิจัยด้าน Music Source Separation ชุดข้อมูลนี้เผยแพร่โดยองค์กร SigSep (Signal Separation Evaluation Campaign) และได้รับการออกแบบมาโดยเฉพาะสำหรับการฝึกและประเมินประสิทธิภาพของโมเดลแยกเสียงโดยตรง ความโดดเด่นของ MUSDB18 คือ ไฟล์เสียงแต่ละไฟล์จะประกอบไปด้วย แทร็กย่อยแบบมัลติแชนแนล (multi-track audio) ซึ่งมีการแยกเสียงเป็น 5 ส่วน ได้แก่ mix (เสียงรวมทั้งหมด), vocals (เสียงร้อง), drums (เสียงกลอง), bass (เสียงเบส) และ other (เสียงเครื่องดนตรีอื่น ๆ ที่เหลือ) ทำให้สามารถนำข้อมูลไปใช้ฝึกโมเดลแบบมี supervision ได้อย่างมีประสิทธิภาพ โดยไม่ต้องเสียเวลาทำ data annotation เพิ่มเติมเอง



ภาพที่ 3.1 ประเภทชุดของมูลของ MUSDB18

(ที่มา: https://sigsep.github.io/datasets/musdb.html)

นอกจากนี้ MUSDB18 ยังมีความละเอียดของเสียงสูง โดยใช้ sampling rate 44.1 kHz, ความลึก 16 บิต และความยาวต่อเพลงอยู่ที่ประมาณ 3–5 นาที ซึ่งเพียงพอสำหรับการฝึกโมเดลที่ต้องการข้อมูลในระดับคุณภาพสูง ข้อมูลทั้งหมดถูกจัดเก็บในฟอร์แมต .stem.mp4 ซึ่งเป็นฟอร์แมตมาตรฐานสำหรับไฟล์เพลงแบบหลายแทร็ก โดยสามารถแยกเสียงออกมาได้ด้วยไลบรารี stempeg อย่างถูกต้อง

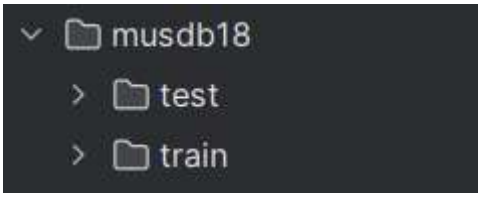
อีกหนึ่งปัจจัยที่สนับสนุนการเลือกชุดข้อมูลนี้คือ MUSDB18 ได้รับการแบ่งย่อยเป็น ชุดฝึก (training set) จำนวน 100 เพลง และ ชุดทดสอบ (test set) จำนวน 50 เพลง โดยมีการกระจายแนวเพลงหลากหลาย เช่น ร็อก ป็อป ฮิปฮอป และดนตรีอิเล็กทรอนิกส์ ซึ่งช่วยให้โมเดลมีโอกาสเรียนรู้จากความหลากหลายของลักษณะเสียงและสามารถประยุกต์ใช้ได้ในสถานการณ์จริงที่หลากหลายมากขึ้น

ดังนั้น การเลือกใช้ MUSDB18 จึงถือว่าเป็นทางเลือกที่เหมาะสมที่สุดสำหรับโครงการนี้ ทั้งในแง่ของความครอบคลุม ความพร้อมใช้งาน และการสนับสนุนด้านเทคนิคที่มีอยู่มากในงานวิจัยที่เกี่ยวข้อง อีกทั้งยังสามารถใช้งานร่วมกับเครื่องมือและไลบรารีต่าง ๆ ในระบบได้อย่างสะดวกโดยไม่ต้องปรับแก้โครงสร้างข้อมูลเพิ่มเติม ทำให้สามารถเริ่มพัฒนาโมเดลได้ทันทีและมั่นใจว่าข้อมูลที่ใช้มีความถูกต้องและเชื่อถือได้ในระดับสากล

3.2.2 การดาวน์โหลดและเตรียมชุดข้อมูล (Data Acquisition and Preparation)

ในโครงการนี้ผู้จัดทำได้เลือกใช้ชุดข้อมูลเสียงดนตรีมัลติแทร็กชื่อว่า MUSDB18 ซึ่งเป็นชุดข้อมูลมาตรฐานที่ใช้ในการวิจัยด้านการแยกแหล่งกำเนิดเสียง (Music Source Separation) โดยเฉพาะ ชุดข้อมูลนี้เผยแพร่โดยกลุ่มงาน SigSep (Signal Separation Evaluation Campaign) และสามารถนำมาใช้ในงานฝึกโมเดลประเภท Supervised Learning ได้โดยตรง เนื่องจากภายในแต่ละไฟล์จะมีการจัดเก็บแทร็กเสียงแบบแยกประเภทไว้อย่างชัดเจน ทั้งเสียงร้อง เสียงกลอง เสียงเบส เสียงเครื่องดนตรีอื่น ๆ และเสียงรวมทั้งหมด (mix)

ชุดข้อมูล MUSDB18 สามารถดาวน์โหลดได้จากเว็บไซต์อย่างเป็นทางการของ SigSep โดยผู้ใช้งานจะต้องทำการลงทะเบียนเพื่อขอสิทธิ์ดาวน์โหลด และข้อมูลที่ได้มาจะอยู่ในรูปแบบไฟล์ .stem.mp4 ซึ่งเป็นรูปแบบไฟล์เสียงมัลติแทร็กที่บรรจุแทร็กเสียงย่อยหลายแทร็กไว้ภายในไฟล์เดียว หลังจากดาวน์โหลดแล้ว ผู้พัฒนาได้ทำการจัดเก็บไฟล์ทั้งหมดลงในโฟลเดอร์ย่อย 2 โฟลเดอร์ ได้แก่ train สำหรับข้อมูลฝึกจำนวน 100 เพลง และ test สำหรับข้อมูลทดสอบจำนวน 50 เพลง เพื่อใช้แยกการใช้งานระหว่างการฝึกโมเดลกับการประเมินผล



ภาพที่ 3.2 โฟลเดอร์ชุดข้อมูลของ MUSDB18

ขั้นตอนนี้เน้นการจัดเตรียมชุดข้อมูลให้พร้อมใช้งานในขั้นตอนถัดไป โดยไม่มีการประมวลผลหรือแปลงข้อมูลเสียงใด ๆ ข้อมูลจะยังคงอยู่ในรูปของไฟล์ .stem.mp4 ทั้งหมด การจัดระเบียบไฟล์ให้อยู่ในโฟลเดอร์ที่ชัดเจน และแยกประเภทการใช้งานอย่างมีระบบ ถือเป็นสิ่งจำเป็นที่ช่วยให้กระบวนการพัฒนาโมเดลสามารถดำเนินไปได้อย่างมีประสิทธิภาพในระยะยาว ทั้งยังเอื้อต่อการเขียนโค้ดสำหรับโหลดข้อมูลในลำดับถัดไปอีกด้วย

3.2.3 การแยกแทร็กเสียงจากไฟล์ .stem.mp4 (Stem Extraction)

หลังจากดำเนินการดาวน์โหลดและจัดระเบียบไฟล์ข้อมูล .stem.mp4 จากชุดข้อมูล MUSDB18 เรียบร้อยแล้ว ขั้นตอนถัดไปคือการแยกแทร็กเสียงที่ฝังอยู่ภายในไฟล์ออกมาเป็นไฟล์ .wav แบบแยกเสียงแต่ละประเภท ได้แก่ เสียงร้อง (vocals), เสียงกลอง (drums), เสียงเบส (bass), เสียงเครื่องดนตรีอื่น ๆ (other) และเสียงผสมรวมทั้งหมด (mix) เพื่อให้สามารถนำไปใช้งานต่อในขั้นตอนการประมวลผลและฝึกโมเดลได้อย่างมีประสิทธิภาพ โดยในขั้นตอนนี้ได้ใช้ไลบรารี stempeg สำหรับโหลดไฟล์ .stem.mp4 และใช้ไลบรารี soundfile ในการบันทึกเสียงออกมาเป็นไฟล์ .wav

ขั้นตอนการแยกแทร็กเสียงสามารถแบ่งออกได้เป็น 4 ส่วนหลักดังนี้

3.2.3.1 กำหนดตำแหน่งของไฟล์ต้นทางและโฟลเดอร์ปลายทาง

ในขั้นตอนแรกจะทำการกำหนดเส้นทางของโฟลเดอร์ที่เก็บไฟล์ .stem.mp4 ทั้งสำหรับชุดฝึกและชุดทดสอบ และสร้างโฟลเดอร์ปลายทางที่ใช้สำหรับจัดเก็บไฟล์เสียง .wav ที่แยกออกมาหากยังไม่มีโฟลเดอร์ดังกล่าวในระบบ

```
# Paths
MUSDB_TRAIN_DIR = "musdb18/train"
MUSDB_TEST_DIR = "musdb18/test"

OUTPUT_STEM_TRAIN = "Data_set/Stem/train"
OUTPUT_STEM_TEST = "Data_set/Stem/test"

# Make output directories
os.makedirs(OUTPUT_STEM_TRAIN, exist_ok=True)
os.makedirs(OUTPUT_STEM_TEST, exist_ok=True)
```

ภาพที่ 3.3 โค๊ดส่วนของการกำหนดตำแหน่งของไฟล์ต้นทางและโฟลเดอร์ปลายทาง

3.2.3.2 ฟังก์ชันสำหรับแยกแทร็กเสียงและบันทึกไฟล์

ในส่วนนี้ได้เขียนฟังก์ชัน process_stem() สำหรับแยกแทร็กเสียงจากไฟล์ .stem.mp4 โดยใช้ฟังก์ชัน stempeg.read_stems() เพื่อโหลดข้อมูลแทร็กทั้งหมด และนำมาแยกเป็น 5 แทร็กตามมาตรฐานของชุดข้อมูล MUSDB18 จากนั้นจึงทำการบันทึกไฟล์เสียงแต่ละแทร็กออกมาเป็นไฟล์ .wav ด้วย soundfile.write() และจัดเก็บลงในโฟลเดอร์ย่อยของแต่ละเพลง

```
# Function to process stems
2 usages
def process_stem(file_path, output_folder):
    # Load stems
    stems, rate = stempeg.read_stems(file_path)

    if stems.shape[0] < 5:
        print(f" Not enough stems: {file_path}")
        return

    # Correct stem order according to MUSDB18 format
    mix    = stems[0]
    drums  = stems[1]
    bass   = stems[2]
    other  = stems[3]
    vocals = stems[4]

    song_name = os.path.splitext(os.path.basename(file_path))[0]
    song_folder = os.path.join(output_folder, song_name)
    stems_folder = os.path.join(song_folder, "stems")
    mix_folder = os.path.join(song_folder, "mix")

    os.makedirs(stems_folder, exist_ok=True)
    os.makedirs(mix_folder, exist_ok=True)

    sf.write(os.path.join(stems_folder, "vocals.wav"), vocals, rate)
    sf.write(os.path.join(stems_folder, "drums.wav"), drums, rate)
    sf.write(os.path.join(stems_folder, "bass.wav"), bass, rate)
    sf.write(os.path.join(stems_folder, "other.wav"), other, rate)
    sf.write(os.path.join(mix_folder, "mix.wav"), mix, rate)
```

ภาพที่ 3.4 โค๊ดส่วนของการกำหนดโฟลเดอร์ชุดข้อมูลของ MUSDB18

3.2.3.3 การประมวลผลข้อมูลฝึกและข้อมูลทดสอบทั้งหมด

ขั้นตอนสุดท้ายคือการวนลูปประมวลผลไฟล์ .stem.mp4 ทุกไฟล์ในโฟลเดอร์ train และ test โดยใช้ฟังก์ชัน process_stem() เพื่อแปลงเป็นไฟล์ .wav และจัดเก็บในโฟลเดอร์เป้าหมายให้เรียบร้อย พร้อมแสดงแถบสถานะความคืบหน้าด้วย tqdm เพื่อให้ผู้พัฒนาเห็นความคืบหน้าของการทำงาน

```
# Process train files
train_files = [f for f in os.listdir(MUSDB_TRAIN_DIR) if f.endswith(".stem.mp4")]
for file_name in tqdm(train_files, desc="Processing train stems"):
    file_path = os.path.join(MUSDB_TRAIN_DIR, file_name)
    process_stem(file_path, OUTPUT_STEM_TRAIN)

# Process test files
test_files = [f for f in os.listdir(MUSDB_TEST_DIR) if f.endswith(".stem.mp4")]
for file_name in tqdm(test_files, desc="Processing test stems"):
    file_path = os.path.join(MUSDB_TEST_DIR, file_name)
    process_stem(file_path, OUTPUT_STEM_TEST)
```

ภาพที่ 3.5 โค๊ดส่วนของการประมวลผลข้อมูลฝึกและข้อมูลทดสอบทั้งหมด

เมื่อดำเนินการแยกแทร็กเสียงเสร็จเรียบร้อยแล้ว แต่ละเพลงจะมีโฟลเดอร์ย่อยจัดเก็บไฟล์เสียงแยกตามประเภท โครงสร้างนี้จะถูกใช้โดยตรงในขั้นตอนการแปลงไฟล์เสียงเป็น Mel-Spectrogram และการฝึกโมเดล deep learning ในลำดับถัดไป การแยกแทร็กเสียงอย่างเป็นระบบจึงถือเป็นรากฐานสำคัญของการเตรียมข้อมูลที่มีคุณภาพสูงและสอดคล้องกับเป้าหมายของโครงการอย่างแท้จริง

```
Data_set/
└── Stem/
    ├── train/
    │   └── Song_Name/
    │       ├── mix/
    │       │   └── mix.wav
    │       └── stems/
    │           ├── vocals.wav
    │           ├── drums.wav
    │           ├── bass.wav
    │           └── other.wav
    └── test/
        └── ...
```

ภาพที่ 3.6 โครงสร้างผลลัพธ์ที่ได้หลังแยกแทร็กเสียง

3.2.5 การเตรียมข้อมูลสำหรับการฝึกโมเดล (Data Generator)

หลังจากดำเนินการแปลงไฟล์เสียง .wav เป็น Mel-Spectrogram และจัดเก็บในรูปแบบไฟล์ .npy แล้ว ข้อมูลที่ได้ยังไม่สามารถนำไปใช้ฝึกโมเดลได้โดยตรง จำเป็นต้องจัดการข้อมูลเหล่านี้ให้อยู่ในรูปแบบที่สามารถโหลดเข้าโมเดลได้อย่างมีประสิทธิภาพ โดยเฉพาะเมื่อข้อมูลมีขนาดใหญ่เกินกว่าจะโหลดทั้งหมดพร้อมกันเข้าสู่หน่วยความจำของระบบ ดังนั้นจึงมีการพัฒนา Data Generator ขึ้นมาเพื่อจัดการโหลดข้อมูลแบบเป็นกลุ่ม (batch) และสามารถสุ่มลำดับใหม่ทุก epoch ได้

ในโครงการนี้ได้พัฒนา Data Generator โดยอิงจาก tf.keras.utils.Sequence ซึ่งเป็นคลาสพื้นฐานของ Keras ที่รองรับการทำงานแบบ multi-thread และเหมาะสำหรับการจัดการข้อมูลจำนวนมาก โดยเก็บไว้ในไฟล์ชื่อ data_generator_mel.py โดยมีขั้นตอนได้แก่

3.2.5.1 กำหนดโครงสร้างคลาส MelDataGenerator

ในขั้นตอนนี้เป็นการกำหนด constructor ซึ่งจะโหลดพาธของไฟล์ .npy ที่อยู่ในโฟลเดอร์ของข้อมูลผสม (mix) และจัดเรียงลำดับ index ของข้อมูลทั้งหมด พร้อมสุ่มหากต้องการ การใช้ assert ตรวจสอบให้แน่ใจว่าจำนวน input และ target ตรงกัน

```
class MelDataGenerator(tf.keras.utils.Sequence):
    def __init__(self, input_root_dir, target_root_dir, batch_size=8, shuffle=True):
        self.input_root_dir = input_root_dir
        self.target_root_dir = target_root_dir
        self.batch_size = batch_size
        self.shuffle = shuffle

        self.input_paths = self._load_file_paths(self.input_root_dir)
        self.target_paths = self._load_file_paths(self.target_root_dir)

        assert len(self.input_paths) == len(self.target_paths), "Mismatch between input and target samples!"

        self.indexes = np.arange(len(self.input_paths))
        if self.shuffle:
            np.random.shuffle(self.indexes)
```

ภาพที่ 3.7 โค๊ดส่วนของการกำหนดโครงสร้างคลาส MelDataGenerator

3.2.5.2 ฟังก์ชันช่วยในการโหลดไฟล์

โค้ดส่วนนี้จะค้นหาไฟล์ .npy ที่อยู่ในโฟลเดอร์ mix ของแต่ละเพลง แล้วเก็บเส้นทางทั้งหมดไว้ในลิสต์ โดยเรียงลำดับอย่างเป็นระบบ ซึ่งเป็นข้อมูล input สำหรับโมเดล

```python
def _load_file_paths(self, root_dir):
    paths = []
    for song_folder in os.listdir(root_dir):
        song_mix_folder = os.path.join(root_dir, song_folder, "mix")
        if not os.path.exists(song_mix_folder):
            continue
        for fname in sorted(os.listdir(song_mix_folder)):
            if fname.endswith(".npy"):
                full_path = os.path.join(song_mix_folder, fname)
                paths.append(full_path)
    return paths
```

ภาพที่ 3.8 โค๊ดส่วนของฟังก์ชันช่วยในการโหลดไฟล์

3.2.5.3 การสร้าง batch ของข้อมูล

เมื่อต้องการ batch ที่ตำแหน่งใด ตัว generator จะเลือก input และ target ของ batch นั้นมาทำการโหลด ซึ่งช่วยให้สามารถฝึกโมเดลได้โดยไม่ต้องโหลดทุกไฟล์พร้อมกัน

```python
def __len__(self):
    return int(np.ceil(len(self.input_paths) / self.batch_size))

def __getitem__(self, index):
    batch_indexes = self.indexes[index * self.batch_size:(index + 1) * self.batch_size]
    batch_input_files = [self.input_paths[k] for k in batch_indexes]
    batch_target_files = [self._get_target_path_from_input_path(p) for p in batch_input_files]

    X, Y = self.__data_generation(batch_input_files, batch_target_files)

    return X, Y
```

ภาพที่ 3.9 โค๊ดส่วนของการสร้าง batch ของข้อมูล

3.2.5.4 ฟังก์ชันสำหรับจับคู่ target จาก input

ฟังก์ชันนี้ใช้จับคู่ชื่อไฟล์ .npy ของ input กับ target ทั้ง 4 แทร็กที่ต้องการให้โมเดลเรียนรู้แยกออกจากกัน

```
1 usage
def _get_target_path_from_input_path(self, input_path):
    song_name = input_path.split(os.sep)[-3]
    segment_name = os.path.basename(input_path)

    target_paths = [
        os.path.join(self.target_root_dir, song_name, "stems", "vocals", segment_name),
        os.path.join(self.target_root_dir, song_name, "stems", "drums", segment_name),
        os.path.join(self.target_root_dir, song_name, "stems", "bass", segment_name),
        os.path.join(self.target_root_dir, song_name, "stems", "other", segment_name),
    ]

    return target_paths
```

ภาพที่ 3.10 โค๊ดส่วนของฟังก์ชันสำหรับจับคู่ target จาก input

3.2.5.5 การโหลดและเตรียมข้อมูลแต่ละ batch

ข้อมูล input และ target ที่โหลดมาจะถูกปรับขนาดให้ตรงกัน โดยการตัดหรือแพดให้มีความกว้าง 864 เฟรม (ความยาวคงที่ที่โมเดลต้องการ) จากนั้นจึงจัดรูปให้อยู่ในรูปแบบที่โมเดลสามารถรับเข้าได้ (เช่นเพิ่ม channel axis)

```
def __data_generation(self, batch_input_files, batch_target_files):
    X_batch = []
    Y_batch = []

    for input_file, target_files in zip(batch_input_files, batch_target_files):
        x = np.load(input_file)
        if x.shape[1] < 864:
            pad_width = 864 - x.shape[1]
            x = np.pad(x, pad_width=((0, 0), (0, pad_width)), mode='constant')
        x = x[:, :864]
        x = np.expand_dims(x, axis=-1)

        y_list = []
        for tf_path in target_files:
            y = np.load(tf_path)
            if y.shape[1] < 864:
                pad_width = 864 - y.shape[1]
                y = np.pad(y, pad_width=((0, 0), (0, pad_width)), mode='constant')
            y = y[:, :864]
            y_list.append(y)

        y = np.stack(y_list, axis=-1)
        X_batch.append(x)
        Y_batch.append(y)

    X_batch = np.array(X_batch)
    Y_batch = np.array(Y_batch)

    return X_batch, Y_batch
```

ภาพที่ 3.11 โค๊ดส่วนของการโหลดและเตรียมข้อมูลแต่ละ batch

3.2.5.6 การสุ่มลำดับข้อมูลใหม่หลังจบแต่ละ epoch

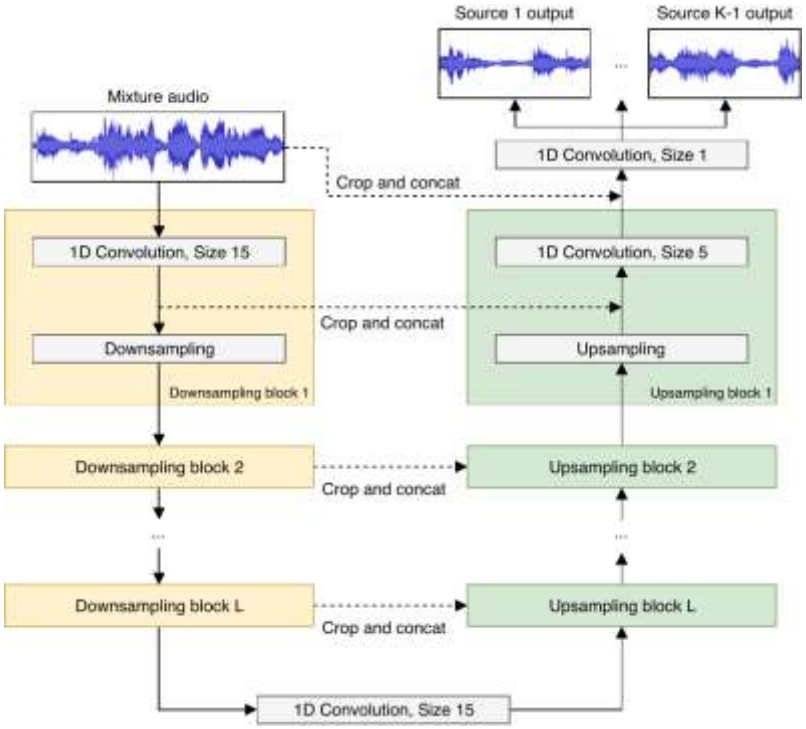
หลังจากจบการฝึกในแต่ละ epoch ตัว generator จะทำการสับลำดับข้อมูลใหม่ เพื่อป้องกันการ overfitting และเพิ่มประสิทธิภาพการเรียนรู้ของโมเดล

```
def on_epoch_end(self):
    if self.shuffle:
        np.random.shuffle(self.indexes)
```

ภาพที่ 3.12 โค๊ดส่วนของการสุ่มลำดับข้อมูลใหม่หลังจบแต่ละ epoch

3.3 การออกแบบและพัฒนาโมเดล (Model Development)

ในการพัฒนาโมเดลสำหรับแยกแหล่งกำเนิดเสียงดนตรีในโครงการนี้ ได้มีการออกแบบและเลือกใช้สถาปัตยกรรมโครงข่ายประสาทเทียมแบบ U-Net ซึ่งเป็นโครงสร้างที่ได้รับความนิยมอย่างแพร่หลายในงานแยกวัตถุ (segmentation) ทั้งในงานประมวลผลภาพและเสียง สาเหตุหลักที่เลือก U-Net มาใช้ เนื่องจากสามารถเรียนรู้คุณลักษณะจากข้อมูลที่อยู่ในรูปแบบ Mel-Spectrogram ได้อย่างมีประสิทธิภาพ โดยใช้แนวคิดการเชื่อมโยงระหว่าง encoder และ decoder ผ่าน skip connection เพื่อรักษาข้อมูลรายละเอียดในระดับตำแหน่งไว้ขณะทำการลดและเพิ่มขนาดของ feature map



ภาพที่3.13 โครงสร้าง Wave-U-Net สำหรับแยกเสียง

โมเดลถูกสร้างและฝึกผ่านไฟล์ train_unet.py ซึ่งมีการกำหนดค่าพารามิเตอร์เบื้องต้น เช่น input shape, batch size, จำนวน epoch, learning rate และฟังก์ชัน loss เป็น mean squared error โดยใช้ Adam optimizer ในการเรียนรู้ พร้อมทั้งใช้ EarlyStopping และ ModelCheckpoint เพื่อควบคุมไม่ให้โมเดลฝึกนานเกินความจำเป็นหากค่า validation loss ไม่ดีขึ้นอีกต่อไป

ด้วยการออกแบบที่อิงจากสถาปัตยกรรม U-Net ที่เหมาะสมกับลักษณะของข้อมูลเสียงแบบภาพ และการปรับขนาดของโมเดลให้สามารถฝึกได้บนเครื่องมือจำกัด เช่น Google Colab หรือเครื่องคอมพิวเตอร์ส่วนบุคคล โมเดลที่พัฒนาขึ้นนี้สามารถเรียนรู้และแยกเสียงได้อย่างมีประสิทธิภาพภายใต้ข้อจำกัดด้านหน่วยความจำและเวลาในการประมวลผลของระบบที่ใช้งานในระดับนักศึกษาหรืองานวิจัยเบื้องต้น

3.3.1 การเตรียมข้อมูลสำหรับการฝึกโมเดล (Data Generator)

โมเดลที่ใช้ในโครงการนี้ถูกพัฒนาขึ้นโดยอิงตามสถาปัตยกรรม U-Net และพัฒนาอยู่ในรูปแบบของฟังก์ชัน build_unet_large() ซึ่งนิยามไว้ในไฟล์ model_unet.py โมเดลประกอบด้วยสองส่วนหลัก คือ Encoder สำหรับลดขนาดข้อมูลเพื่อดึงคุณลักษณะเชิงลึก และ Decoder สำหรับขยายข้อมูลกลับคืน พร้อมทั้งใช้การเชื่อมโยงแบบ skip connection เพื่อคงรายละเอียดของข้อมูลที่สูญหายจากการ downsampling

3.3.1.1 กำหนด Input Layer

โมเดลรับข้อมูลขนาด (128, 864, 1) โดยที่ 128 คือจำนวน Mel-band, 864 คือจำนวน frame ต่อ segment, และ 1 คือจำนวน channel ของภาพ input ซึ่งเป็นภาพ grayscale ที่แทน Mel-Spectrogram ของเสียงผสม

```
def build_unet_large(input_shape=(128, 864, 1), num_outputs=4):
    inputs = Input(shape=input_shape)
```

ภาพที่3.14 โค๊ดส่วนของการกำหนด Input Layer

3.3.1.2 Encoder – ลดขนาดข้อมูลและเพิ่มจำนวนฟีเจอร์

แต่ละบล็อกประกอบด้วย 2 convolutional layers ที่ใช้ kernel ขนาด 3×3 และ activation function แบบ ReLU ตามด้วยการลดขนาดด้วย MaxPooling2D เพื่อลด resolution ของข้อมูลลงครึ่งหนึ่งในแต่ละระดับ ขณะที่จำนวน filter เพิ่มขึ้นเรื่อย ๆ เพื่อให้โมเดลสามารถเรียนรู้คุณลักษณะที่ซับซ้อนขึ้นตามลำดับ

```
# Encoder
c1 = Conv2D(64, (3, 3), activation='relu', padding='same')(inputs)
c1 = Conv2D(64, (3, 3), activation='relu', padding='same')(c1)
p1 = MaxPooling2D((2, 2))(c1)

c2 = Conv2D(128, (3, 3), activation='relu', padding='same')(p1)
c2 = Conv2D(128, (3, 3), activation='relu', padding='same')(c2)
p2 = MaxPooling2D((2, 2))(c2)

c3 = Conv2D(256, (3, 3), activation='relu', padding='same')(p2)
c3 = Conv2D(256, (3, 3), activation='relu', padding='same')(c3)
p3 = MaxPooling2D((2, 2))(c3)

c4 = Conv2D(512, (3, 3), activation='relu', padding='same')(p3)
c4 = Conv2D(512, (3, 3), activation='relu', padding='same')(c4)
c4 = Dropout(0.3)(c4)
p4 = MaxPooling2D((2, 2))(c4)
```

ภาพที่3.15 โค๊ดส่วนของโครงสร้าง Encoder

3.3.1.3 Bottleneck – ชั้นลึกสุดของโมเดล

ในชั้นกลางของ U-Net หรือที่เรียกว่า bottleneck ใช้ convolution จำนวน 2 ชั้น พร้อมการใส่ dropout เพื่อลดโอกาสเกิด overfitting โดยยังคงใช้ filter ขนาด 3×3 และจำนวน filter สูงถึง 768 ตัว เพื่อเรียนรู้คุณลักษณะที่ลึกที่สุด

```
# Bottleneck
c5 = Conv2D(768, (3, 3), activation='relu', padding='same')(p4)
c5 = Conv2D(768, (3, 3), activation='relu', padding='same')(c5)
```

ภาพที่3.16 โค๊ดส่วนของ Bottleneck

3.3.1.4 Decoder – ขยายขนาดข้อมูลพร้อมเชื่อม skip connection

ฝั่ง decoder จะขยายขนาดข้อมูลด้วย UpSampling2D และนำมารวมกับข้อมูลจาก encoder ชั้นที่มีขนาดเท่ากันผ่าน Concatenate() ซึ่งเป็นกลไกสำคัญของ U-Net ที่ช่วยให้โมเดลสามารถนำคุณลักษณะที่สูญหายไปในขั้นตอน downsampling กลับมาใช้ใหม่ได้ในขั้นตอน upsampling

```
# Decoder
u6 = UpSampling2D((2, 2))(c5)
u6 = Concatenate()([u6, c4])
c6 = Conv2D(512, (3, 3), activation='relu', padding='same')(u6)
c6 = Conv2D(512, (3, 3), activation='relu', padding='same')(c6)

u7 = UpSampling2D((2, 2))(c6)
u7 = Concatenate()([u7, c3])
c7 = Conv2D(256, (3, 3), activation='relu', padding='same')(u7)
c7 = Conv2D(256, (3, 3), activation='relu', padding='same')(c7)

u8 = UpSampling2D((2, 2))(c7)
u8 = Concatenate()([u8, c2])
c8 = Conv2D(128, (3, 3), activation='relu', padding='same')(u8)
c8 = Conv2D(128, (3, 3), activation='relu', padding='same')(c8)

u9 = UpSampling2D((2, 2))(c8)
u9 = Concatenate()([u9, c1])
c9 = Conv2D(64, (3, 3), activation='relu', padding='same')(u9)
c9 = Conv2D(64, (3, 3), activation='relu', padding='same')(c9)
```

ภาพที่3.17 โครงสร้าง Wave-U-Net สำหรับแยกเสียง

3.3.1.5 Output Layer – แยกเสียงออกเป็น 4 ช่อง

ในเลเยอร์สุดท้ายใช้ convolution ขนาด 1×1 เพื่อให้ output มีขนาดเท่ากับ input แต่มีจำนวน channel เท่ากับจำนวนเสียงที่ต้องการแยกคือ 4 (vocals, drums, bass, other) โดยใช้ activation แบบ linear เพื่อให้ค่า output มีช่วงต่อเนื่องสำหรับการประมาณค่าพลังงานเสียงในแต่ละพิกเซล

```
outputs = Conv2D(num_outputs, (1, 1), activation='linear')(c9)
model = Model(inputs=[inputs], outputs=[outputs])
return model
```

ภาพที่3.18 โค๊ดส่วนของ Decoder

3.4 การฝึกโมเดล (Model Training)

หลังจากที่ได้ทำการออกแบบและพัฒนาโครงสร้างของโมเดลเสร็จสมบูรณ์แล้ว ขั้นตอนถัดไปคือการนำโมเดลเข้าสู่กระบวนการฝึก (training) ด้วยข้อมูลที่ถูกเตรียมไว้อย่างเป็นระบบในรูปแบบของ Mel-Spectrogram และจัดการผ่าน Data Generator เพื่อให้สามารถจัดการกับข้อมูลจำนวนมากได้อย่างมีประสิทธิภาพ กระบวนการฝึกโมเดลเป็นขั้นตอนสำคัญที่จะกำหนดว่าโมเดลสามารถเรียนรู้รูปแบบของเสียงและแยกแหล่งกำเนิดเสียงได้ดีเพียงใด โดยจะต้องอาศัยการตั้งค่าพารามิเตอร์ต่าง ๆ อย่างเหมาะสม ทั้งในส่วนของ loss function, optimizer, learning rate, batch size, จำนวน epoch และกลไกในการหยุดการฝึกอัตโนมัติเพื่อป้องกัน overfitting

3.4.1 การกำหนดพาธและสร้างโฟลเดอร์สำหรับจัดเก็บผลลัพธ์

ระบบจะระบุที่อยู่ของข้อมูลฝึกและข้อมูลทดสอบ ทั้งในส่วนของ input (mix) และ target (เสียงที่แยกแล้ว) และสร้างโฟลเดอร์สำหรับบันทึกโมเดลและกราฟผลลัพธ์

```
# Paths
TRAIN_INPUT_DIR = "Data_set/Spectrogram/train"
TRAIN_TARGET_DIR = "Data_set/Spectrogram/train"
VAL_INPUT_DIR = "Data_set/Spectrogram/test"
VAL_TARGET_DIR = "Data_set/Spectrogram/test"
OUTPUT_MODEL_DIR = "outputs"
os.makedirs(OUTPUT_MODEL_DIR, exist_ok=True)
```

ภาพที่3.19 โค๊ดส่วนของการกำหนดพาธและสร้างโฟลเดอร์สำหรับจัดเก็บผลลัพธ์

3.4.2 การกำหนดพารามิเตอร์หลักสำหรับการฝึก

มีการตั้งค่าพารามิเตอร์พื้นฐาน เช่น ขนาด input ของภาพ Mel-Spectrogram, ขนาด batch, จำนวนรอบการฝึก (epoch) และจำนวนรอบที่ระบบจะรอเมื่อ validation ไม่ดีขึ้นก่อนหยุดการฝึก (patience)

```
# Parameters
input_shape = (128, 864, 1)
batch_size = 4
epochs = 30
patience = 15
```

ภาพที่3.20 โค๊ดส่วนของการกำหนดพารามิเตอร์หลักสำหรับการฝึก

3.4.3 การสร้างและ compile โมเดล

เรียกใช้ฟังก์ชัน build_unet_large() เพื่อสร้างโมเดล U-Net ที่มีโครงสร้าง encoder-decoder และกำหนด optimizer, loss function และ metric ที่ใช้ในการประเมินผล

```
# Load model
model = build_unet_large(input_shape=input_shape, num_outputs=4)
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),
    loss='mean_squared_error',
    metrics=['mean_absolute_error']
)
```

ภาพที่3.21 โค๊ดส่วนของการสร้างและ compile โมเดล

3.4.4 การโหลดข้อมูลฝึกและทดสอบด้วย Data Generator

ใช้คลาส MelDataGenerator ซึ่งพัฒนาไว้เพื่อโหลดข้อมูลจากไฟล์ .npy แบบ batch โดยไม่ต้องโหลดทั้งหมดเข้าสู่หน่วยความจำพร้อมกัน

```
# Load data
train_generator = MelDataGenerator(TRAIN_INPUT_DIR, TRAIN_TARGET_DIR, batch_size=batch_size)
val_generator = MelDataGenerator(VAL_INPUT_DIR, VAL_TARGET_DIR, batch_size=batch_size)
```

ภาพที่3.22 โค๊ดส่วนของการโหลดข้อมูลฝึกและทดสอบด้วย Data Generator

3.4.5 การกำหนด callback สำหรับควบคุมกระบวนการฝึก

ใช้ EarlyStopping เพื่อหยุดการฝึกหาก validation loss ไม่ดีขึ้นตามจำนวนรอบที่กำหนด และ ModelCheckpoint เพื่อบันทึกโมเดลที่มีประสิทธิภาพดีที่สุดตาม validation loss

```
# Callbacks
callbacks = [
    tf.keras.callbacks.EarlyStopping(
        monitor='val_loss',
        patience=patience,
        verbose=1,
        restore_best_weights=True
    ),
    tf.keras.callbacks.ModelCheckpoint(
        filepath=os.path.join(OUTPUT_MODEL_DIR, "best_model_unet.keras"),
        monitor='val_loss',
        save_best_only=True,
        verbose=1
    )
]
```

ภาพที่3.23 โค๊ดส่วนของการกำหนด callback สำหรับควบคุมกระบวนการฝึก

3.4.6 การฝึกโมเดลด้วยข้อมูลที่เตรียมไว้

ใช้ model.fit() เพื่อทำการฝึกโมเดลด้วยชุดข้อมูลฝึกและทดสอบ โดยระบุ generator, จำนวน epoch, callback และการแสดงผล

```
# Plot Loss Graph
plt.figure(figsize=(8, 6))
plt.plot( *args: history.history['loss'], label='Train Loss')
plt.plot( *args: history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.savefig(os.path.join(OUTPUT_MODEL_DIR, 'loss_curve_unet.png'))
plt.close()
```

ภาพที่3.24 โค๊ดส่วนของการฝึกโมเดลด้วยข้อมูลที่เตรียมไว้

3.4.7 การวิเคราะห์ผลการฝึกและบันทึกกราฟ

หลังจากฝึกโมเดลเสร็จแล้ว มีการสร้างกราฟแสดงค่า loss และ MAE ทั้งในชุดฝึกและชุดทดสอบ เพื่อใช้ในการประเมินประสิทธิภาพของโมเดลในแต่ละ epoch

3.5 การประเมินโมเดล

ในการประเมินโมเดลเราจะใช้โปรแกรม Sound Similar Free ซึ่งเป็นเครื่องมือสำหรับวิเคราะห์และเปรียบเทียบความเหมือนของสัญญาณเสียงโดยเฉพาะ ซึ่งสามารถใช้ตรวจสอบความคล้ายคลึงของไฟล์เสียงสองไฟล์ในเชิงคณิตศาสตร์ โดยมีการคำนวณค่าความเหมือน เช่น ค่าความคลาดเคลื่อนหรือการเปรียบเทียบลักษณะของคลื่นเสียงอย่างละเอียด

ในโครงการนี้ Sound Similar Free ถูกใช้เพื่อตรวจสอบประสิทธิภาพของโมเดลแยกเสียงที่พัฒนาขึ้น โดยการนำเสียงที่ถูกแยกออกมาจากโมเดลไปเปรียบเทียบกับเสียงต้นฉบับของแต่ละแทร็ก เช่น เสียงร้อง เสียงกลอง เสียงเบส และเสียงเครื่องดนตรีอื่น ๆ ซึ่งช่วยให้สามารถประเมินได้ว่าโมเดลสามารถแยกเสียงได้ดีเพียงใดในมุมมองของผู้ใช้ปลายทาง โดยไม่ต้องอาศัยการคำนวณทางเทคนิคที่ซับซ้อนเหมือนในโค้ด Python

บทที่ 4
ผลการวิจัย

ในการดำเนินการวิจัยและพัฒนาระบบแยกเสียงดนตรีในครั้งนี้ คณะผู้จัดทำมีความมุ่งมั่นที่จะสำรวจแนวทางการประยุกต์ใช้เทคโนโลยีปัญญาประดิษฐ์ (AI) เพื่อจำแนกและแยกองค์ประกอบของเสียงจากไฟล์เพลง โดยเฉพาะการแยกเสียงร้อง (Vocals) ออกจากเสียงดนตรี (Drums, Bass และ Other instruments) เพื่อศึกษาความเป็นไปได้ในการสร้างต้นแบบระบบแยกเสียง ที่สามารถนำไปประยุกต์ใช้ได้ในอนาคต การดำเนินการทดลองนี้ได้ถูกวางแผนอย่างเป็นระบบ และมีการประเมินผลที่ครอบคลุมทั้งด้านคุณภาพของโมเดลและการใช้งานจริง รายละเอียดของผลการวิจัยมีดังต่อไปนี้

4.1 ข้อมูลเสียงและชุดข้อมูลที่ใช้ในการสร้างโมเดล

4.1.1 ชุดข้อมูลเสียง

ใช้ชุดข้อมูล MUSDB18 ซึ่งเป็นชุดข้อมูลมาตรฐานในงานวิจัยการแยกเสียง (Source Separation)

ประกอบด้วย:

- 100 ไฟล์เสียงในชุดฝึก (Train Set)
- 50 ไฟล์เสียงในชุดทดสอบ (Test Set)

แต่ละไฟล์มีการแยกแหล่งเสียงออกเป็น 4 ประเภท: Vocals, Drums, Bass, และ Other

4.1.2 การเตรียมข้อมูลเสียง

ทำการตัดไฟล์เสียงออกเป็น Segment ยาว 10 วินาที เพื่อไม่ให้เกิดภาระกับหน่วยความจำ

ใช้อัตราการสุ่มตัวอย่าง (Sampling Rate) ที่ 44.1kHz

แปลงไฟล์เสียงเป็น Mel-Spectrogram ขนาด (128, 864) ต่อ Segment

ทำการ Normalize ค่าสเกลของ Mel-Spectrogram ให้อยู่ในช่วง 0-1

4.2 ผลการฝึกโมเดล

4.2.1 การตั้งค่าการฝึก

โมเดลที่ใช้: U-Net Architecture ที่ดัดแปลงให้มี Output 4 ช่องทาง

ขนาดอินพุต: (128, 864, 1)

Optimizer: Adam (learning_rate=0.0001)

Loss Function: Mean Squared Error (MSE)

Epochs: 100 รอบ

Batch Size: 8

4.2.2 ผลลัพธ์จากการฝึก (Training Results)

ค่า Training Loss ลดลงจาก ~3000 ⟶ เหลือ ~163

ค่า Validation Loss ลดลงจาก ~500 ⟶ เหลือ ~232

ค่า Mean Absolute Error (MAE) อยู่ในช่วง ~7.9-9.1

โมเดลไม่มีอาการ Overfitting อย่างชัดเจน

มีการบันทึกโมเดลที่ดีที่สุดที่ได้ค่าความสูญเสีย Validation ต่ำที่สุดลงในไฟล์ best_model_unet.h5 กราฟ Loss Curve แสดงการลดลงอย่างต่อเนื่อง สะท้อนถึงการเรียนรู้ที่มีประสิทธิภาพ

4.3 ข้อมูลชุดทดสอบ (Testing Dataset)

ทำการนำเข้าไฟล์เสียงตัวอย่าง Interstellar Journey.mp3 แล้วแปลงเป็น Mel-Spectrogram ตามขนาดที่กำหนด

ตัดแบ่งเพื่อป้อนเข้าสู่โมเดล ทำการแยกแหล่งเสียง 4 ช่อง (Vocals, Drums, Bass, Other) ออกมาเป็นไฟล์ .wav

4.4 ผลการแยกเสียง (Separation Results)

4.4.1 ประเภทเสียงที่แยกได้

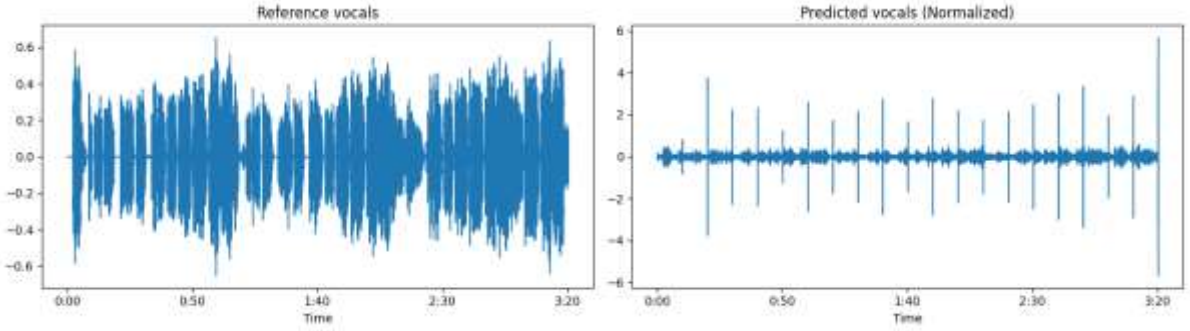
Vocals แยกเสียงร้องได้ แต่ยังมีเสียงเครื่องดนตรีปนเข้ามา เสียงส่วนใหญ่ถูกดึงเข้ามาอยู่ใน Vocals

Drums เสียงกลองแยกออกมาได้อย่างชัดเจน ความเพี้ยนของเสียงต่ำ

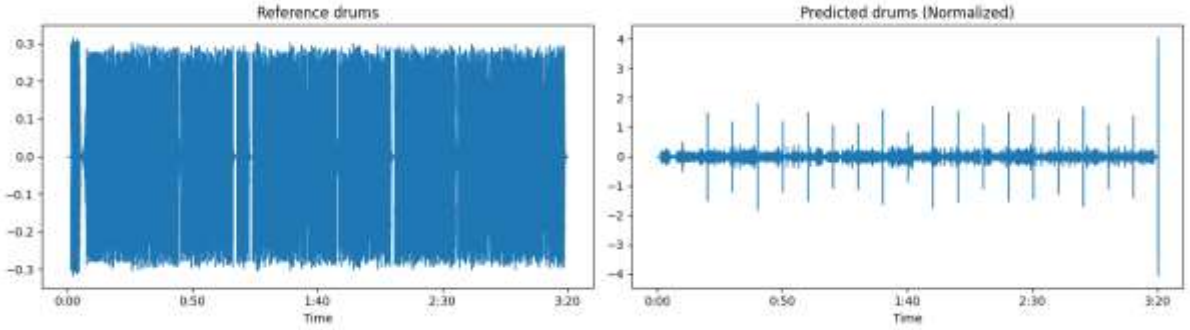
Bass เสียงเบสมีความเบาและหลุดบางช่วง โมเดลยังไม่สามารถจำแนกเบสได้ดีพอ

Other รวมเสียงเครื่องดนตรีอื่นๆ มีเสียงร้องเล็ดลอดมาบ้าง

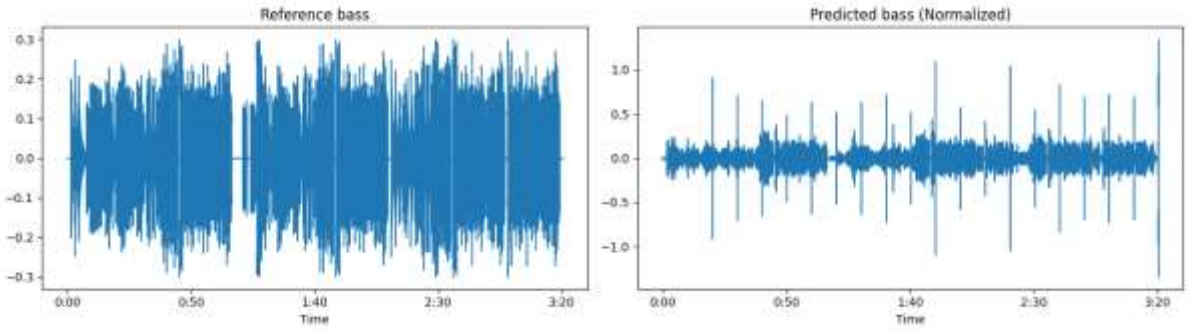
4.4.2 ประเภทเสียงที่แยกได้ด้วยกราฟเสียง



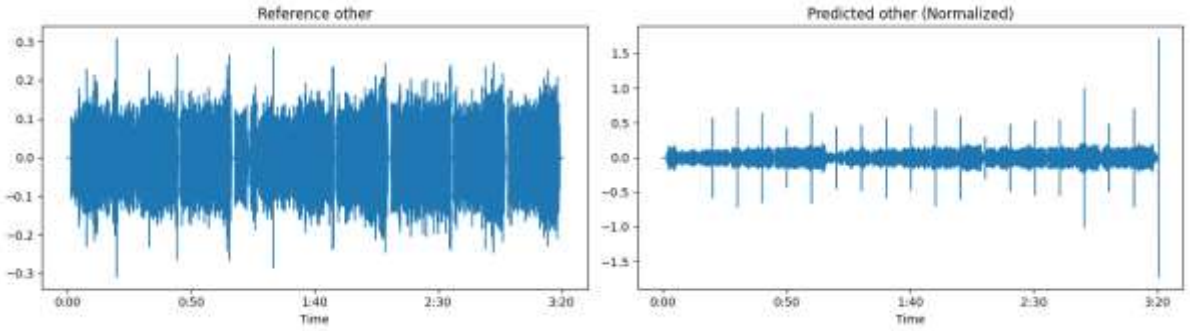
ภาพที่4.1 กราฟเปรียบเทียบความเหมือนของเสียงต้นฉบับและโมเดลของ Vocals



ภาพที่4.2 กราฟเปรียบเทียบความเหมือนของเสียงต้นฉบับและโมเดลของ Drums



ภาพที่4.3 กราฟเปรียบเทียบความเหมือนของเสียงต้นฉบับและโมเดลของ Bass

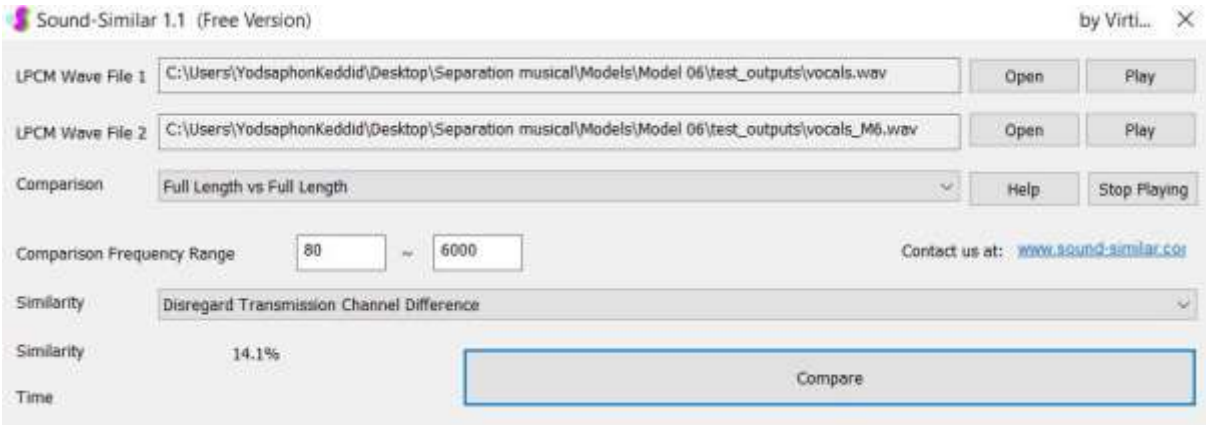


ภาพที่4.4 กราฟเปรียบเทียบความเหมือนของเสียงต้นฉบับและโมเดลของ Other

จากกราฟที่แสดงให้เห็นการเปรียบเทียบระหว่างเสียงจากไฟล์เสียงต้นฉบับฝั่งซ้ายจะเห็นได้ว่า เสียงที่โมเดลสามารถแยกออกมาได้ในฝั่งขวามีพลังงานของเสียงต่ำกว่าเสียงของต้นฉบับ แล้วมีบางช่วงที่เส้นเสียงมีแนวโน้มขยายตัวขึ้น ซึ่งเกิดจากช่วงระหว่างการตัด-ต่อประกอบ Segment ในแต่ละช่วง

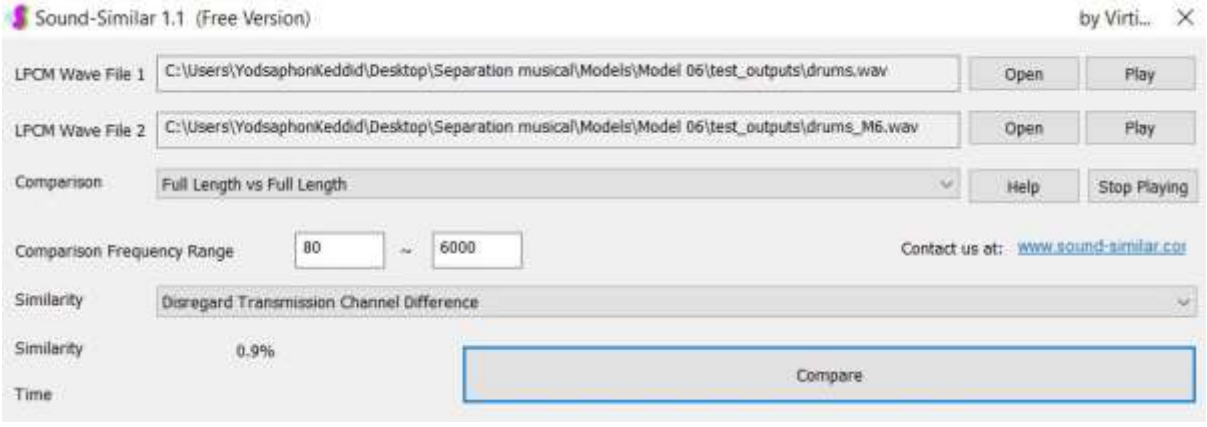
4.4.3 ประเภทเสียงที่แยกได้ด้วยโปรแกรม Sound Similar

การประเมินค่าความเหมือนระหว่างเสียงที่แยกโดยโมเดลกับเสียงต้นฉบับในรายงานนี้ ดำเนินการโดยใช้ซอฟต์แวร์ Sound Similar ซึ่งเป็นโปรแกรมสำหรับเปรียบเทียบความใกล้เคียงของไฟล์เสียงในเชิง waveform โดยโปรแกรมจะประมวลผลระดับความคล้ายคลึงและแสดงผลเป็นค่าเปอร์เซ็นต์ ซึ่งสามารถนำมาใช้เป็นดัชนีชี้วัดเบื้องต้นของประสิทธิภาพการแยกเสียงของโมเดล



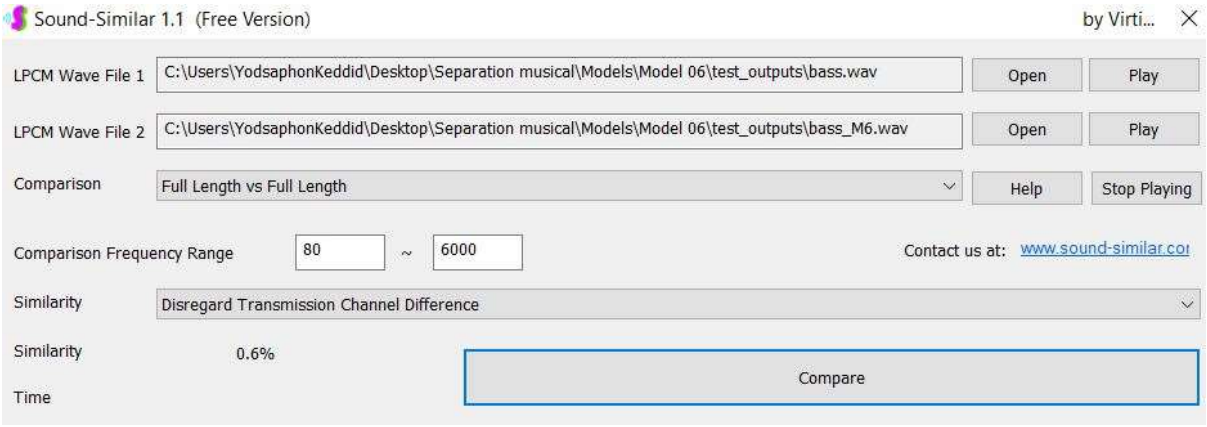
ภาพที่4.5 ประเมินความเหมือนของเสียงต้นฉบับและโมเดลของ Vocals

ผลการประเมินการแยกเสียงร้องจากโมเดลในช่วงความถี่ 80 ถึง 6000 เฮิรตซ์ พบว่ามีค่าความเหมือนอยู่ที่ 14.1% แสดงให้เห็นว่าโมเดลสามารถแยกเสียงร้องได้ในระดับพอใช้ มีการจับรายละเอียดของเสียงร้องบางส่วนได้ แต่ยังมีข้อจำกัดในด้านความแม่นยำ



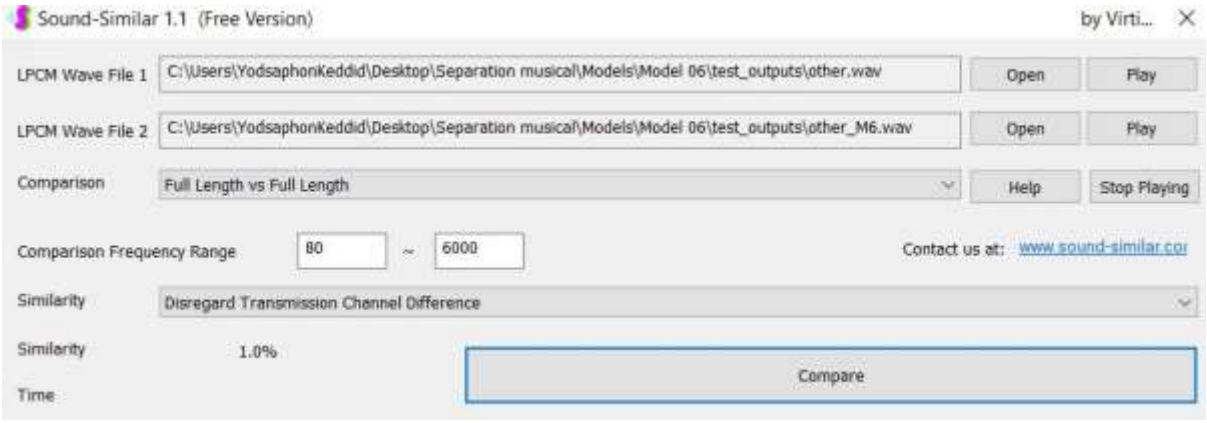
ภาพที่4.6 ประเมินความเหมือนของเสียงต้นฉบับและโมเดลของ Drums

ค่าความเหมือนของเสียงกลองที่ได้จากการประเมินในช่วงความถี่ 80 ถึง 6000 เฮิรตซ์ อยู่ที่ 0.9% ถือว่าอยู่ในระดับต่ำ แสดงว่าโมเดลยังไม่สามารถจับลักษณะเฉพาะของเสียงกลองได้ดีพอในย่านความถี่นี้



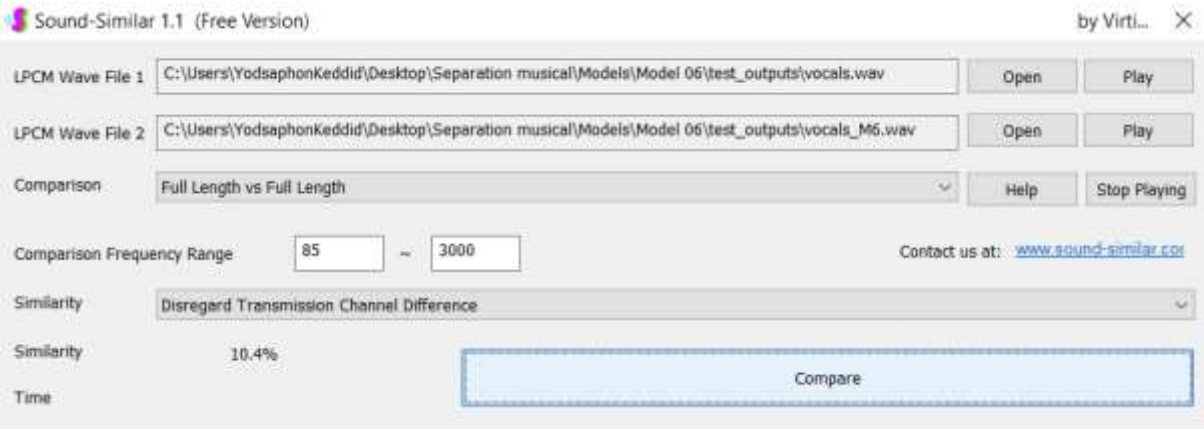
ภาพที่4.7 ประเมินความเหมือนของเสียงต้นฉบับและโมเดลของ Bass

โมเดลมีค่าความเหมือนในการแยกเสียงเบสที่ 0.6% จากการประเมินในช่วงความถี่ 80 ถึง 6000 เฮิรตซ์ ซึ่งถือว่าต่ำมาก ทำให้เห็นว่าโมเดลไม่สามารถแยกเสียงเบสได้อย่างมีประสิทธิภาพภายใต้ช่วงความถี่นี้



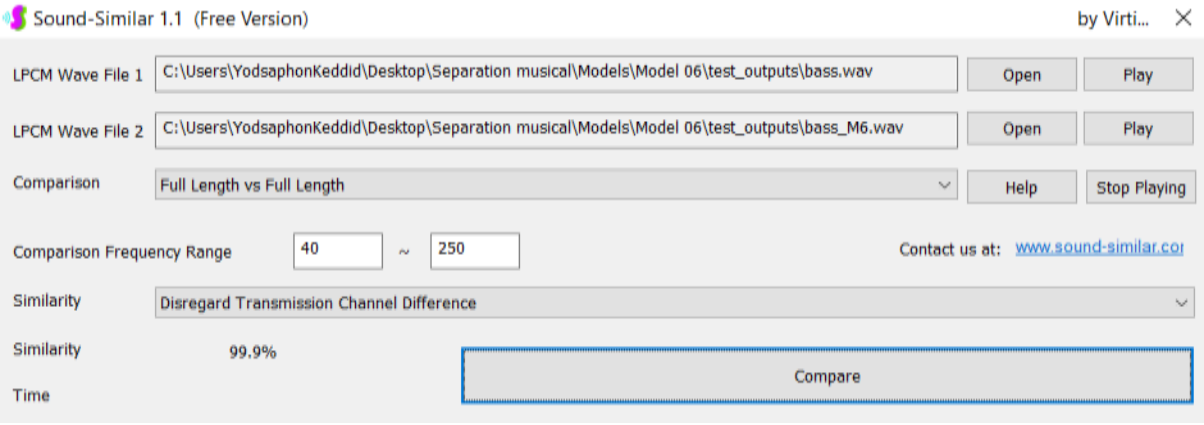
ภาพที่4.8 ประเมินความเหมือนของเสียงต้นฉบับและโมเดลของ Other

ผลการแยกเสียงประเภทอื่นในช่วงความถี่ 80 ถึง 6000 เฮิรตซ์ ให้ค่าความเหมือน 1.0% ซึ่งถือว่าต่ำมาก โมเดลยังไม่สามารถระบุลักษณะของเสียงอื่น ๆ ได้อย่างแม่นยำภายใต้ช่วงความถี่นี้



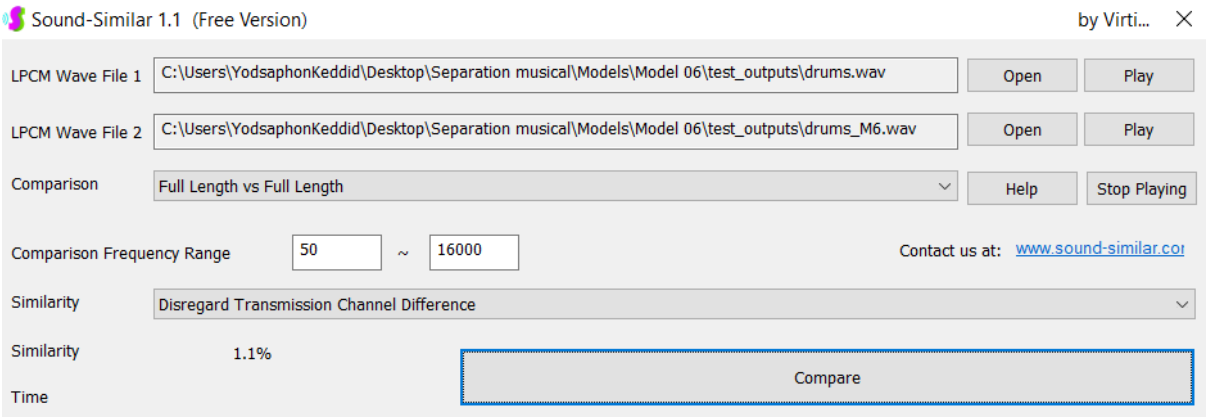
ภาพที่4.9 ประเมินความเหมือนของเสียงต้นฉบับและโมเดลของ Vocals

เมื่อประเมินเสียงร้องในช่วงความถี่ 85 ถึง 3000 เฮิรตซ์ ซึ่งเป็นย่านความถี่เฉพาะของเสียงร้อง พบว่าค่าความเหมือนอยู่ที่ 10.4% จัดอยู่ในระดับปานกลาง แสดงว่าโมเดลสามารถแยกเสียงร้องได้บางส่วน แต่ยังไม่ชัดเจนหรือแม่นยำเท่าที่ควร



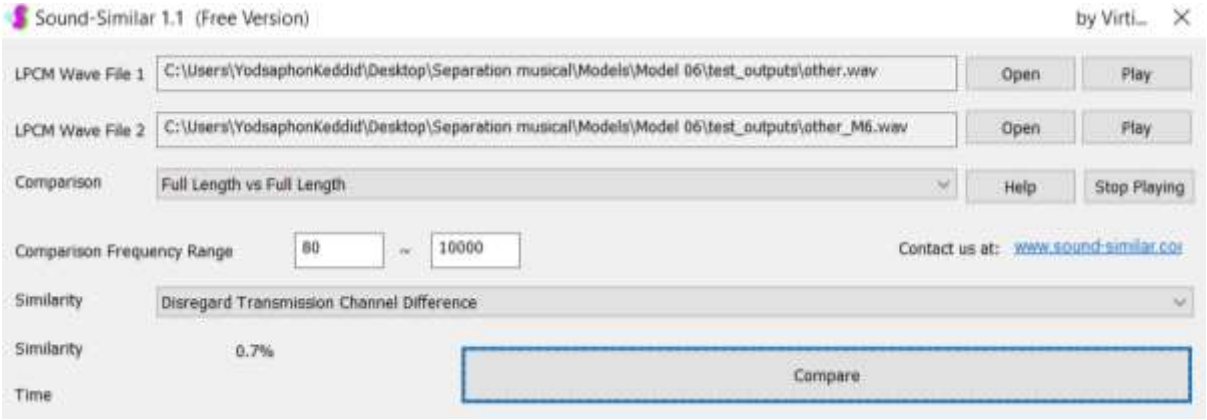
ภาพที่4.10 ประเมินความเหมือนของเสียงต้นฉบับและโมเดลของ Bass

ผลการประเมินในช่วงความถี่ 40 ถึง 250 เฮิรตซ์ แสดงว่าโมเดลมีค่าความเหมือนสูงถึง 99.9% ซึ่งอยู่ในระดับดีเยี่ยม แสดงให้เห็นว่าโมเดลสามารถแยกเสียงเบสได้อย่างแม่นยำและสอดคล้องกับต้นฉบับเป็นอย่างมาก



ภาพที่4.11 ประเมินความเหมือนของเสียงต้นฉบับและโมเดลของ Drums

ค่าความเหมือนที่ได้จากการประเมินเสียงกลองในช่วงความถี่ 50 ถึง 16000 เฮิรตซ์ คือ 1.1% จัดว่าอยู่ในระดับต่ำ สะท้อนว่าโมเดลยังไม่สามารถแยกเสียงกลองออกมาได้อย่างมีประสิทธิภาพในช่วงความถี่นี้



ภาพที่4.12 ประเมินความเหมือนของเสียงต้นฉบับและโมเดลของ Other

การประเมินเสียงอื่นในช่วงความถี่ 80 ถึง 10000 เฮิรตซ์ พบว่าค่าความเหมือนอยู่ที่ 0.7% ซึ่งถือว่าต่ำมาก แสดงว่าโมเดลไม่สามารถแยกเสียงประเภทอื่นได้อย่างแม่นยำในช่วงความถี่นี้

จากผลลัพธ์การประเมินจะเห็นได้ว่า ความเหมือนของคลื่นเสียงระหว่างไฟล์ต้นฉบับ และไฟล์ที่โมเดลแยกออกมาได้นั้นค่อนข้างต่ำ อันเนื่องมาจากพลังงานของเสียงที่ต่ำกว่าต้นฉบับ การพบเสียงรบกวนภายในไฟล์เสียงที่แยกออกมา และยังคงมีการปะปนของเสียงประเภทอื่นภายในไฟล์ที่แยกออกมา

ประเภทของเสียง	ความถี่ 80–6000 Hz (%	ย่านความถี่เฉพาะ (%)	ย่านความถี่เฉพาะที่ใช้
เสียงร้อง	14.1	10.4	85 – 3000 Hz
เบส	0.6	99.9	40 – 250 Hz
กลอง	0.9	1.1	50 – 16000 Hz
เสียงอื่น	1.0	0.7	80 – 10000 Hz

ภาพที่4.13 ตารางเปรียบเทียบเปอร์เซ็นความเหมือน

จากผลการประเมินความเหมือนระหว่างเสียงที่โมเดลทำการแยกออกมากับเสียงต้นฉบับ พบว่าโมเดลมีประสิทธิภาพในการแยกเสียงแต่ละประเภทแตกต่างกัน

โดยเสียงร้องเมื่อประเมินในช่วงความถี่ 80–6000 Hz ให้ค่าความเหมือนที่ 14.1% ถือว่าอยู่ในระดับปานกลาง ส่วนเสียงเบสในช่วงเดียวกันให้ค่าความเหมือนเพียง 0.6% ซึ่งจัดว่าอยู่ในระดับต่ำมาก เช่นเดียวกับเสียงกลองและเสียงอื่นที่มีค่าความเหมือน 0.9% และ 1.0% ตามลำดับ

สำหรับการประเมินด้วยช่วงความถี่เฉพาะของแต่ละเสียง พบว่าเสียงร้องในช่วง 85–3000 Hz มีค่าความเหมือน 10.4% ซึ่งยังคงอยู่ในระดับปานกลาง เสียงเบสในช่วง 40–250 Hz ให้ผลลัพธ์ที่ดีมากโดยมีค่าความเหมือนสูงถึง 99.9% แสดงว่าโมเดลสามารถแยกเสียงเบสได้อย่างแม่นยำเมื่อใช้ย่านความถี่ที่เหมาะสม เสียงกลองที่ประเมินในช่วง 50–16000 Hz ให้ค่าความเหมือน 1.1% ซึ่งยังถือว่าต่ำ

ขณะที่เสียงอื่นที่ใช้ช่วงความถี่ 80–10000 Hz มีค่าความเหมือนเพียง 0.7% แสดงว่าโมเดลยังมีข้อจำกัดในการแยกเสียงประเภทอื่นโดยรวมแล้วผลการประเมินชี้ให้เห็นว่าโมเดลมีความสามารถเฉพาะในการแยกเสียงเบสได้ดีหากใช้ช่วงความถี่ที่ตรงกับลักษณะของเสียง แต่ยังต้องปรับปรุงเพิ่มเติมในส่วนของเสียงร้อง เสียงกลอง และเสียงอื่นเพื่อเพิ่มความแม่นยำในการแยกเสียงโดยรวม

4.5 การวิเคราะห์เชิงลึก

4.5.1 จุดแข็งของโมเดล

การแยกเสียงกลอง (Drums) ทำได้อย่างน่าพอใจ

โมเดลสามารถจับ Pattern ของเสียงในย่านต่ำ (Low frequency) ได้ดี

4.5.2 ข้อจำกัดที่พบ

การแยกเสียงร้อง (Vocals) ยังมีเสียงเครื่องดนตรีปนอยู่มาก

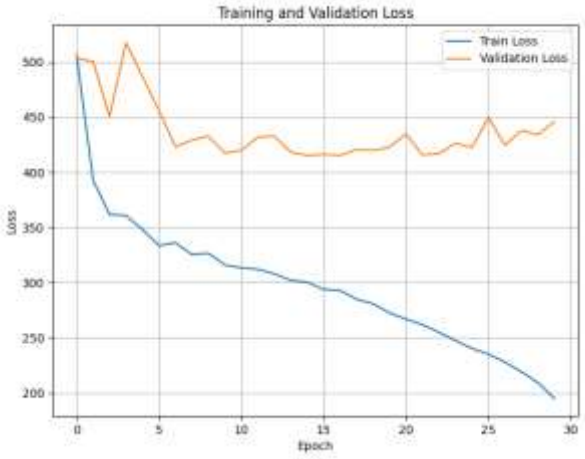
โมเดลมีปัญหาในการจำแนกเสียงเบสที่มีความถี่ต่ำ

ข้อมูล Input มีขนาดจำกัด (10s segment) อาจส่งผลต่อประสิทธิภาพ

4.5.3 ปัจจัยที่มีผลกระทบ

ความละเอียดของ Spectrogram (128 Mel Bands) อาจน้อยเกินไป

ขาดการใช้เทคนิค Post-Processing เช่น Spectrogram Masking หรือ Phase Reconstruction



ภาพที่4.8 กราฟผลการเทรนโมเดล

ซึ่งจากกราฟการประเมินการเรียนรู้และการทำนายของโมเดลจะเห็นได้ว่า แม้โมเดลจะเรียนรู้ข้อมูลได้เป็นอย่างดี ถึงกระนั้นโมเดลก็เกิด Overfitting อย่างรวดเร็ว

4.6 บทสรุปผลการทดลอง

ระบบต้นแบบสามารถแยกเสียงประเภทต่างๆ ออกจากเพลงได้จริง ประสิทธิภาพของโมเดลอยู่ในระดับที่ ต้นแบบ (Prototype Level) มีแนวทางในการต่อยอดเพื่อพัฒนาโมเดลให้แม่นยำและมีคุณภาพเสียงสูงขึ้นได้ในอนาคต

บทที่ 5
สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ

ในการวิจัยครั้งนี้ คณะผู้จัดทำได้พัฒนาระบบการแยกเสียงดนตรีออกจากไฟล์เสียง โดยเน้นไปที่การใช้เทคโนโลยีปัญญาประดิษฐ์ (Artificial Intelligence) และการเรียนรู้ของเครื่อง (Machine Learning) เพื่อเพิ่มความแม่นยำและประสิทธิภาพในการแยกองค์ประกอบของเสียง ได้แก่ เสียงร้อง (Vocals), กลอง (Drums), เบส (Bass) และเสียงอื่นๆ (Other) จากไฟล์เสียงผสม (Mix) ทั้งยังได้ทำการทดสอบการแยกเสียงอย่างละเอียดด้วยกระบวนการเปรียบเทียบผลการแยกเสียงกับข้อมูลต้นฉบับ บทนี้จะสรุปผลการดำเนินงานวิจัย วิเคราะห์ปัญหา และเสนอแนะแนวทางการปรับปรุงและพัฒนาระบบในอนาคต

5.1 สรุปผลการวิจัย

จากการดำเนินงานวิจัย พบว่าโมเดล U-Net ที่ได้รับการฝึกสอนสามารถแยกเสียงจากไฟล์ผสมได้จริง ซึ่งขั้นตอนการดำเนินงานประกอบด้วย

5.1.1 การเตรียมชุดข้อมูลที่มีไฟล์ต้นฉบับ (vocals, drums, bass, other) และไฟล์ผสม (mix)

5.1.2 การตัดแบ่งข้อมูลเสียงออกเป็นช่วงย่อยขนาดที่เหมาะสมเพื่อป้อนเข้าสู่โมเดล

5.1.3 การนำโมเดลที่ฝึกแล้วมาใช้ในการทำนายและแยกองค์ประกอบของเสียงจากไฟล์เสียง

5.1.4 การประกอบไฟล์เสียงที่แยกได้กลับมาเป็นไฟล์เต็ม และการเปรียบเทียบกับไฟล์ต้นฉบับ

5.1.5 จากการประเมินด้วยตัวชี้วัด เช่น Sound Similar และการเปรียบเทียบสัญญาณเสียง

5.2 การอภิปรายผล

5.2.1 ประสิทธิภาพของระบบแยกเสียง

จากการประเมินผลของระบบแยกเสียงด้วยโมเดลที่พัฒนาขึ้น พบว่าเสียงที่ได้จากโมเดลมีความคล้ายคลึงกับเสียงต้นฉบับในระดับต่ำ โดยค่าร้อยละของความเหมือนเฉลี่ยเมื่อเปรียบเทียบกับเสียงต้นฉบับแยกตามประเภทคือ เสียงร้อง เสียงกลอง เสียงเบส และเสียงดนตรีอื่นๆ ซึ่งแสดงให้เห็นว่าระบบยังไม่สามารถแยกองค์ประกอบของเสียงต่าง ๆ ได้อย่างแม่นยำเท่าที่ควร โดยเฉพาะในกรณีของเสียงดนตรีประเภทเบสและกลองที่มีความเหมือนกับต้นฉบับน้อยมาก

5.2.2 ข้อจำกัดของระบบ

ระบบแยกเสียงที่พัฒนาขึ้นมีข้อจำกัดสำคัญหลายประการ ประการแรกคือคุณภาพของเสียงที่ได้มีเสียงรบกวน (noise) ปะปนอยู่ในระดับสูง ส่งผลให้เสียงที่แยกออกมาไม่สะอาดและอาจยากต่อการนำไปใช้งานต่อ ประการที่สองคือมีการแทรกของเสียงจากประเภทอื่นในแต่ละแทร็ก เช่น แทร็กของเสียงร้องอาจมีเสียงดนตรีปะปน และแทร็กของกลองอาจมีเสียงเบสเจืออยู่ด้วย นอกจากนี้ระบบยังมีปัญหาเรื่องพลังงานของสัญญาณเสียง (signal energy) ที่ต่ำกว่าเสียงต้นฉบับอย่างมีนัยสำคัญ ทำให้เสียงที่แยกออกมาเบาและขาดมิติเมื่อเทียบกับเสียงจริง

5.2.3 การวิเคราะห์ปัจจัยที่ส่งผลต่อผลลัพธ์

ปัจจัยที่อาจส่งผลต่อผลลัพธ์ของระบบแยกเสียงในงานวิจัยนี้ประกอบด้วยหลายด้าน ได้แก่:

1. โครงสร้างของโมเดลที่อาจยังไม่เหมาะสมหรือมีความสามารถไม่เพียงพอในการเรียนรู้ลักษณะเฉพาะของแต่ละแหล่งเสียง
2. ขนาดและคุณภาพของชุดข้อมูลฝึกที่อาจไม่เพียงพอสำหรับการเรียนรู้ที่หลากหลาย โดยเฉพาะในกรณีที่เสียงดนตรีมีความซับซ้อนหรือมีการซ้อนทับกันมาก
3. ข้อจำกัดด้านทรัพยากรการประมวลผลที่อาจทำให้ไม่สามารถฝึกโมเดลขนาดใหญ่หรือใช้เทคนิคขั้นสูง

5.3 ข้อเสนอแนะ

เพื่อพัฒนาระบบให้มีประสิทธิภาพมากยิ่งขึ้นในอนาคต ควรพิจารณาดังนี้:

1. เพิ่มขนาดของชุดข้อมูลฝึกให้หลากหลายและครอบคลุมลักษณะของเสียงแต่ละประเภทมากขึ้น
2. ปรับปรุงกระบวนการแปลงข้อมูลให้เหมาะสมกับลักษณะของเสียงที่จะวิเคราะห์
3. ทดสอบใช้ฟังก์ชันค่าความสูญเสีย (loss function) ที่ออกแบบมาเฉพาะสำหรับงานแยกเสียง เช่น SI-SDR Loss

5.4 สรุปผลการวิจัย

งานวิจัยนี้มีวัตถุประสงค์ เพื่อพัฒนาระบบแยกเสียงดนตรีออกจากกันโดยใช้เทคนิคการเรียนรู้ของเครื่องผ่านโครงข่ายประสาทเทียมแบบ U-Net ซึ่งจากการทดลองพบว่า แม้ระบบจะสามารถแยกเสียงร้องออกจากเสียงดนตรีได้ในระดับหนึ่ง แต่ผลลัพธ์โดยรวมยังอยู่ในระดับต่ำ โดยค่าความเหมือนของเสียงที่ได้จากโมเดลเมื่อเปรียบเทียบกับเสียงต้นฉบับเสียงร้อง เสียงดนตรีอื่น เสียงกลอง และเสียงเบส ทั้งนี้เสียงที่แยกได้ยังมีเสียงรบกวนแทรกอยู่ และมีความผิดเพี้ยนจากต้นฉบับในด้านพลังงานเสียงและความคมชัดของแหล่งเสียง

โดยสาเหตุหลักน่าจะมาจากข้อจำกัดของโมเดลที่ใช้ โครงสร้างข้อมูลฝึก และวิธีการประมวลผลเบื้องต้นซึ่งอาจยังไม่เหมาะสมเพียงพอ อย่างไรก็ตาม ผลลัพธ์ที่ได้แสดงให้เห็นถึงศักยภาพของแนวทางในการแยกเสียงด้วยปัญญาประดิษฐ์ และสามารถนำไปต่อยอดในงานวิจัยหรือการพัฒนาระบบแยกเสียงที่มีประสิทธิภาพสูงยิ่งขึ้นในอนาคต

บรรณานุกรม

[1] A. V. Oppenheim and R. W. Schafer, Discrete-Time Signal Processing, 3rd ed., Pearson, 2010.

[2] U. Zölzer, Digital Audio Signal Processing, 3rd ed., Wiley, 2022.

[3] R. G. Lyons, Understanding Digital Signal Processing, 3rd ed., Prentice Hall, 2010.

[4] S. W. Smith, The Scientist and Engineer's Guide to Digital Signal Processing, California Technical Pub., 1999.

[5] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," Adv. Neural Inf. Process. Syst., vol. 13, 2001, pp. 556–562.

[6] P. Comon, "Independent component analysis, a new concept?," Signal Process., vol. 36, no. 3, 1994, pp. 287–314.

[7] E. Vincent, T. Virtanen, and S. Gannot, Eds., Audio Source Separation and Speech Enhancement, Wiley, 2018.

[8] D. P. W. Ellis and G. E. Poliner, Introduction to Audio Analysis: A MATLAB® Approach, Academic Press, 2011.

[9] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," IEEE Trans. Audio, Speech, Lang. Process., vol. 28, no. 4, 1980, pp. 357–366.

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Interv. (MICCAI), 2015, pp. 234–241.

[11] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep U-Net convolutional networks," in Proc. 18th Int. Soc. Music Information Retrieval Conf. (ISMIR), 2017, pp. 746–751.

[12] D. Stoller, S. Ewert, and S. Dixon, "Wave-U-Net: A multi-scale neural network for end-to-end audio source separation," in Proc. 19th Int. Soc. Music Information Retrieval Conf. (ISMIR), 2018.

[13] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," IEEE Trans. Audio, Speech, Lang. Process., vol. 14, no. 4, 2006, pp. 1462–1469.

บรรณานุกรม(ต่อ)

[14] M. Abadi et al., “TensorFlow: Large-scale machine learning on heterogeneous systems,” Tech. Rep., Google, 2015.

[15] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto, “librosa: Audio and music signal analysis in Python,” in Proc. 14th Python in Science Conf. (SciPy), 2015.

[16] F.-R. Stoeter, “stempeg: Python I/O for STEM audio files,” GitHub repository, 2019.

[17] โปรแกรม Sound Similar Free https://www.virtins.com/VT-Sound-Recognition.html

ประวัติผู้วิจัย

ชื่อ – นามสกุล	นายยศพล เกตุดิษฐ์
ประวัติการศึกษา	ระดับปริญญาตรี พ.ศ. 2566 วิศวกรรมศาสตร์บัณฑิต วิศวกรรมศาสตร์และเทคโนโลยี สาขาวิศวกรรมคอมพิวเตอร์และปัญญาประดิษฐ์
ตำแหน่งปัจจุบัน	นักศึกษา
สถานศึกษา	สถาบันการจัดการปัญญาภิวัฒน์ วิทยาเขตแจ้งวัฒนะ
อีเมลล์	junkeddid@gmail.com
เบอร์โทรศัพท์	088 - 244 - 6444