การขยายข้อมูลในทางเวลาสำหรับการเรียนรู้เชิงลึกเพื่อใช้ในการตรวจจับความผิดปกติทางอุตสาหกรรม

กฤชภัทร เบญจกรัณย์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
บัณฑิตศึกษา สาขาเทคโนโลยีวิศวกรรม
สถาบันเทคโนโลยีไทย-ญี่ปุ่น
ปีการศึกษา 2566

TIME-SERIES AUGMENTATION-BASED DEEP LEARNING FOR INDUSTRIAL ANORMALY DETECTION

Kritchapat Benjakaran

A Thesis Submitted in Partial Fulfillment of Requirements

for The Degree of Master of Engineering Program in Engineering Technology

Graduate Studies

Thai-Nichi Institute of Technology

Academic Year 2023

หัวข้อวิทยานิพนธ์	การขยายข้อมูลในทางเวลาสำหรับการเรียนรู้เชิงลึกเพื่อใช้ในการตรวจจับความผิดปกติทางอุตสาหกรรม
โดย	กฤชภัทร เบญจกรัณย์
สาขาวิชา	เทคโนโลยีวิศวกรรม
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รองศาสตราจารย์ ดร.วรากร ศรีเชวงทรัพย์

บัณฑิตศึกษา สถาบันเทคโนโลยีไทย-ญี่ปุ่น อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... รองอธิการบดีฝ่ายวิชาการ
(รองศาสตราจารย์ ดร.วรากร ศรีเชวงทรัพย์)
วันที่.....เดือน.....พ.ศ.....

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ดร.ณัฐกิตติ์ จิตรเอื้อตระกูล)

..... กรรมการ
(ดร.ชัชไชย วรรณบูรณ์)

..... กรรมการ
(ดร.อัดนา เซนโต๊ะ)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์
(รองศาสตราจารย์ ดร.วรากร ศรีเชวงทรัพย์)

กฤชภัทร เบญจกรัณย์ : การขยายข้อมูลในทางเวลาสำหรับการเรียนรู้เชิงลึกเพื่อใช้ในการตรวจจับความผิดปกติทางอุตสาหกรรม. อาจารย์ที่ปรึกษา : รองศาสตราจารย์ ดร. วรากร ศรีเชวงทรัพย์, 50 หน้า.

การตรวจจับความผิดปกติในกระบวนการทางอุตสาหกรรมอย่างทันท่วงทีถือเป็นสิ่งสำคัญยิ่ง เนื่องจากสามารถป้องกันความเสียหาย ลดต้นทุน และเพิ่มความปลอดภัย ในปัจจุบันการเก็บข้อมูลการทำงานของเครื่องจักรและกระบวนการต่างๆ อยู่ในรูปแบบชุดข้อมูล time series เพื่อใช้ในการวิเคราะห์ ซึ่งวิธีการตรวจจับอาศัยกฎเกณฑ์ที่ตั้งไว้ล่วงหน้า (Rule-based) วิธีการเหล่านี้มีข้อจำกัดเนื่องจากไม่สามารถปรับตัวกับความผิดปกติใหม่ๆได้ อีกทั้งยังต้องใช้ความเชี่ยวชาญเฉพาะด้านในการกำหนดกฎเกณฑ์ ในขณะที่วิธีการทางสถิติ (Statistical) เช่น การคำนวณค่าเบี่ยงเบนมาตรฐาน (Standard deviation) สามารถปรับตัวกับความผิดปกติได้ดีกว่า แต่ก็ยังมีข้อจำกัดในกรณีที่ความผิดปกติมีรูปแบบที่ซับซ้อน จึงนำวิธีการเรียนรู้เชิงลึก (Deep learning) ซึ่งเป็นเทคนิคการเรียนรู้ของเครื่องจักร (Machine learning) ที่ได้รับความนิยมอย่างมากในปัจจุบัน โดยมีสามารถเรียนรู้รูปแบบที่ซับซ้อนจากข้อมูลจำนวนมากและนำไปใช้ในการแก้ปัญหาต่างๆได้อย่างมีประสิทธิภาพ อย่างไรก็ตาม หนึ่งในข้อจำกัดสำคัญของการเรียนรู้เชิงลึกคือการที่ต้องการข้อมูลจำนวนมากในการฝึกโมเดล ในกรณีของข้อมูลอุตสาหกรรม ข้อมูลความผิดปกติมักมีอยู่น้อยกว่าข้อมูลปกติ ทำให้โมเดลที่ฝึกมาจากข้อมูลเหล่านี้มีประสิทธิภาพในการตรวจจับความผิดปกติต่ำ ดังนั้นงานวิจัยชิ้นนี้จึงได้หยิบยกเทคนิคการขยายข้อมูลในทางเวลา (Time series data augmentation) ซึ่งเป็นเทคนิคที่ช่วยเพิ่มขนาดและความหลากหลายของข้อมูล time series โดยไม่ต้องเก็บข้อมูลใหม่ เพื่อแก้ปัญหาข้อมูลไม่เพียงพอในการฝึกโมเดลการเรียนรู้เชิงลึก ด้วยเทคนิคต่างๆ อาทิ Oversampling : เพิ่มจำนวนตัวอย่างของข้อมูลความผิดปกติ, Under sampling : ลดจำนวนตัวอย่างของข้อมูลปกติ เป็นต้น

งานวิจัยชิ้นนี้นำเสนอโดยใช้เทคนิคการขยายข้อมูลในทางเวลาและโมเดลการเรียนรู้เชิงลึกจากกรณีศึกษาของข้อมูลการสั่นสะเทือนด้วยวัตถุจากเครื่องพิมพ์ 3 มิติ ด้วยขั้นตอนการศึกษาคือ 1) การเก็บข้อมูล 2) การแบ่งข้อมูล 3) การขยายข้อมูล 4) การฝึกโมเดล 5) การประเมินผล ซึ่งผลลัพท์สามารถลดจำนวนข้อมูลจริงที่ต้องวัดผลจากเครื่องจักรได้ถึง 40% แสดงให้เห็นถึงการใช้งานในสภาพแวดล้อมที่ข้อมูลมีจำกัดได้เป็นอย่างดี

บัณฑิตศึกษา ลายมือชื่อนักศึกษา .....

สาขาวิชา เทคโนโลยีวิศวกรรม ลายมือชื่ออาจารย์ที่ปรึกษา .....

ปีการศึกษา 2566

KRITCHAPAT BENJAKARAN : TIME-SERIES AUGMENTATION-BASED DEEP LEARNING FOR INDUSTRIAL ANOMALY DETECTION. ADVISOR : ASSOCIATE PROFESSOR DR. WARAKORN SRICHAVENGSUP, 50 PP.

Timely anomaly detection in industrial processes is crucial for preventing damage, reducing costs, and improving safety. Currently, machine and process operation data are collected in time series format for analysis. Detection methods based on predefined rules (Rule-based) have limitations as they cannot adapt to new anomalies and require specialized expertise to define the rules. Statistical methods, such as standard deviation calculation, can adapt better to anomalies but still have limitations when anomalies have complex patterns. Deep learning is a popular machine learning technique that can learn complex patterns from large amounts of data and apply them to solve various problems effectively. However, one major limitation of deep learning is the need for large amounts of data to train models. In the case of industrial data, anomaly data is often less than normal data, making models trained on this data less effective at detecting anomalies. This research addresses this issue by using time series data augmentation, a technique that increases the size and diversity of time series data without collecting new data. Techniques used include Oversampling: increasing the number of anomaly data samples, Under sampling: reducing the number of normal data samples, etc.

This research is presented using temporal data expansion techniques and deep learning models. From the case study of vibration data from objects from 3D printers, the study steps are 1) data collection, 2) data division, 3) data expansion, 4) model training, 5) evaluation, which results It can reduce the amount of actual data that must be measured by machines by up to 40%, demonstrating its utility in data-constrained environments.

Graduate Studies Student's Signature .....

Field of Engineering of Technology Advisor's Signature .....

Academic Year 2023.

กิตติกรรมประกาศ

งานวิทยานิพนธ์เรื่อง “การขยายข้อมูลในทางเวลาสำหรับการเรียนรู้เชิงลึกเพื่อใช้ในการตรวจจับความผิดปกติทางอุตสาหกรรม” ฉบับนี้ สำเร็จลุล่วงไปได้ด้วยความกรุณาอย่างยิ่งจากรองศาสตราจารย์ ดร.วรากร ศรีเชวงทรัพย์ อาจารย์ที่ปรึกษาและดร. ชัชไชย วรรณบูรณ์ ซึ่งได้ให้ความรู้ คำปรึกษา แนะนำแนวทาง เพื่อนำไปปรับปรุง และแก้ไขข้อบกพร่องต่างๆ เพื่อให้งานสมบูรณ์ยิ่งขึ้น พร้อมทั้งยังช่วยพัฒนาการทำงานของผู้วิทยานิพนธ์ให้เป็นไปอย่างมีคุณภาพ และสนับสนุนเครื่องมือที่ใช้ในการทำงานวิจัยนี้ อีกทั้งยังดูและเอาใจใส่ดีตลอดระยะเวลาในการทำวิทยานิพนธ์

ขอขอบพระคุณคณะกรรมการทุกท่านที่ได้ให้ความรู้ คำแนะนำ และข้อคิดที่เป็นประโยชน์ต่อการทำวิทยานิพนธ์ฉบับนี้ พร้อมทั้งให้คำแนะนำแนวทาง ปรับปรุงแก้ไข จนสามารถทำให้วิทยานิพนธ์ฉบับนี้เสร็จสิ้นสมบูรณ์ตามขอบเขตที่กำหนดไว้

นอกจากนี้ทางผู้วิจัยขอขอบคุณแรงสนับสนุนจากครอบครัว และคนรอบข้างที่คอยให้กำลังใจ ให้คำปรึกษาตลอดระยะเวลาการศึกษา จนจัดทำวิทยานิพนธ์ฉบับนี้เสร็จสิ้น

ผู้วิจัยหวังเป็นอย่างยิ่งว่า วิทยานิพนธ์ฉบับนี้จะก่อให้เกิดประโยชน์แก่ผู้ที่สนใจ และสามารถนำไปต่อยอดกับองค์ความรู้ที่เกี่ยวข้องได้ อีกทั้งสามารถก่อให้เกิดแนวคิดเพื่อใช้ในการศึกษา หรือเป็นแนวทางการทำงานวิจัยที่เกี่ยวข้องในอนาคต

กฤชภัทร เบญจกรัณย์

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

การตรวจจับความผิดปกติทางอุตสาหกรรมเป็นสิ่งสำคัญสำหรับการป้องกันความเสียหายต่ออุปกรณ์ เพิ่มประสิทธิภาพการผลิต และรักษาความปลอดภัยของพนักงาน วิธีการแบบดั้งเดิมในการตรวจจับความผิดปกติทางอุตสาหกรรมสามารถแบ่งออกเป็นสองประเภทหลัก ได้แก่ 1.) วิธีการเชิงสถิติ ซึ่งเป็นการเปรียบเทียบค่าเซ็นเซอร์กับค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐานที่คำนวณจากข้อมูลในอดีต ค่าที่อยู่นอกช่วงที่กำหนดจะถูกพิจารณาว่าผิดปกติ 2.) วิธีการเชิงกฎ เป็นการใช้กฎที่กำหนดไว้ล่วงหน้าเพื่อระบุความผิดปกติ มักสร้างขึ้นโดยประสบการณ์และความรู้จากผู้เชี่ยวชาญ ตัวอย่างเช่น รูปแบบการสั่นสะเทือนที่ผิดปกติอาจบ่งบอกถึงปัญหาเกี่ยวกับมอเตอร์ วิธีการแบบดั้งเดิมเหล่านี้มีข้อจำกัดหลายประการ เช่น มักพึ่งพาผู้เชี่ยวชาญในการกำหนดค่าและสร้างกฎ ซึ่งอาจใช้เวลานานและเกิดข้อผิดพลาดได้ และไม่ยืดหยุ่นต่อการเปลี่ยนแปลงของข้อมูลหรือสภาพการทำงานรวมทั้งมีประสิทธิภาพต่ำในการตรวจจับความผิดปกติที่ซับซ้อน

การเรียนรู้เชิงลึก (Deep Learning) เป็นเทคโนโลยีที่ได้รับความนิยมมากขึ้นสำหรับการตรวจจับความผิดปกติทางอุตสาหกรรม โมเดล การเรียนรู้เชิงลึกสามารถเรียนรู้รูปแบบที่ซับซ้อนจากข้อมูลจำนวนมาก และสามารถนำไปใช้เพื่อตรวจจับความผิดปกติในข้อมูลใหม่ งานวิจัยมากมายได้แสดงให้เห็นว่าโมเดลการเรียนรู้เชิงลึกมีสามารถใช้เพื่อวิเคราะห์ข้อมูลจำนวนมหาศาล และเรียนรู้รูปแบบที่ซับซ้อน ซึ่งช่วยให้สามารถตรวจจับความผิดปกติในอุตสาหกรรมได้อย่างมีประสิทธิภาพ เช่น การใช้โมเดล Convolutional Neural Network (CNN) สำหรับการตรวจจับความผิดปกติของมอเตอร์[1-2] ซึ่งสามารถเรียนรู้รูปแบบการสั่นสะเทือนที่บ่งบอกถึงความผิดปกติของมอเตอร์ได้ [3-4] นำเสนอโมเดล Deep Autoencoder สำหรับการตรวจจับความผิดปกติของเครื่องจักร ซึ่งสามารถเรียนรู้รูปแบบการทำงานปกติของเครื่องจักร และสามารถตรวจจับความผิดปกติจากรูปแบบที่ผิดปกติ และนอกจากนี้ โมเดล Long Short-Term Memory (LSTM) [5] เพื่อการตรวจจับความผิดปกติของสัญญาณจากเครื่องจักร โดยโมเดลจะเรียนรู้ที่จะสร้างพฤติกรรมแบบปกติของข้อมูลเชิงเวลา และหลังจากนั้นจะใช้ความคลาดเคลื่อนจากการสร้างข้อมูลขึ้นใหม่ เพื่อหาความผิดปกติ ผลการทดลองของทุกงานวิจัยแสดงให้เห็นว่าโมเดลการเรียนรู้เชิงลึก สามารถตรวจจับความผิดปกติได้จากข้อมูลหลายรูปแบบทั้งคาดการณ์ได้ คาดการณ์ไม่ได้ มีรอบคาบหรือไม่มีรอบคาบ อีกทั้งยังสามารถแยกแยะความผิดปกติขากข้อมูลที่ซับซ้อนได้

อย่างไรก็ตาม โมเดลการเรียนรู้เชิงลึกสำหรับการตรวจจับความผิดปกติในเครื่องจักรมักต้องการข้อมูลจำนวนมากสำหรับการฝึกฝน โดยทั่วไปจะมีการติดตั้งเซ็นเซอร์จำนวนมาก เพื่อเก็บข้อมูลพฤติกรรมและสภาวะสุขภาพของเครื่องจักรเหล่านั้น การรวบรวมข้อมูลจำนวนมากในลักษณะนี้ในอุตสาหกรรมอาจเป็นเรื่องยากและมีค่าใช้จ่ายสูง อีกทั้งยังสิ้นเปลืองเวลาอย่างมากในการเก็บข้อมูล การขยายข้อมูลในทางเวลา (Time-Series Augmentation) เป็นวิธีหนึ่งที่สามารถช่วยแก้ปัญหานี้ได้ โดยการเพิ่มปริมาณข้อมูลที่มีอยู่โดยไม่ลดคุณภาพของข้อมูล ด้วยการใช้เทคนิคต่างๆ เช่น การสุ่มตัวอย่าง (Sampling), การผสมผสานข้อมูล (Combining), และการเปลี่ยนแปลงสัญญาณ (Transformation) วิธีนี้ช่วยให้โมเดลการเรียนรู้เชิงลึกสามารถเข้าถึงข้อมูลที่หลากหลายมากขึ้นโดยไม่จำเป็นต้องมีการเก็บรวบรวมข้อมูลเพิ่มเติม

ด้วยเหตุนี้งานวิจัยนี้มุ่งเน้นไปที่การพัฒนาวิธีการตรวจจับความผิดปกติทางอุตสาหกรรมโดยใช้โมเดลการเรียนรู้เชิงลึก ผสมผสานกับเทคนิคการขยายข้อมูลเชิงเวลา เทคนิคนี้ช่วยเพิ่มขนาดข้อมูลที่ใช้ฝึกสอนโมเดลปัญญาประดิษฐ์ โดยการสร้างข้อมูลจำลองใหม่จากข้อมูลที่มีอยู่ งานวิจัยนี้ใช้ชุดข้อมูลจากเซ็นเซอร์วัดการสั่นสะเทือนที่ติดตั้งบนมอเตอร์ ผลลัพธ์ที่ได้แสดงให้เห็นว่าการใช้เทคนิคการขยายขนาดทางเวลาช่วยลดจำนวนข้อมูลที่ใช้ในการฝึกโมเดลการเรียนรู้เชิงลึก ลงอย่างมีนัยสำคัญ โมเดลที่ผ่านการฝึกอบรมด้วยชุดข้อมูลที่ขยายขนาดทางเวลาแล้ว สามารถตรวจจับความผิดปกติของมอเตอร์ได้อย่างแม่นยำ ผลงานวิจัยนี้แสดงให้เห็นว่าการใช้เทคนิคขยายขนาดทางเวลาร่วมกับโมเดลการเรียนรู้เชิงลึก เป็นแนวทางที่มีประสิทธิภาพสำหรับการตรวจจับความผิดปกติทางอุตสาหกรรม โดยใช้ข้อมูลจำนวนน้อยลง ซึ่งสามารถลดค่าใช้จ่ายและเวลาในการเก็บรวบรวมข้อมูลจากเครื่องจักรได้

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษาเทคนิคการขยายขนาดข้อมูลในทางเวลา

1.2.2 เพื่อศึกษาการใช้โมเดลการเรียนรู้เชิงลึกในการตรวจจับความผิดปกติในข้อมูลเชิงเวลา

1.2.3 เพื่อศึกษาการใช้ข้อมูลจากการขยายขนาดข้อมูลในทางเวลาในการฝึกสอนโมเดลการเรียนรู้เชิงลึก

1.3 ขอบเขตงานวิจัย

1.3.1 พัฒนาวิธีการตรวจจับความผิดปกติทางอุตสาหกรรมโดยใช้โมเดลการเรียนรู้เชิงลึก (Deep Learning) ผสมผสานกับเทคนิคการขยายข้อมูลเชิงเวลา (Time Series Data Augmentation)

1.3.2 ประเมินประสิทธิภาพของโมเดลที่พัฒนาขึ้น โดยเปรียบเทียบกับวิธีการแบบดั้งเดิมและโมเดล Deep Learning ที่ไม่มีการขยายข้อมูล

1.3.3 วิเคราะห์ผลลัพธ์และสรุปผลการวิจัย นำเสนอข้อดีข้อเสีย แนวทางการพัฒนาต่อยอด และประโยชน์ของงานวิจัย

1.4 ผลที่คาดว่าจะได้รับ

1.4.1 สร้างโมเดล Deep learning ที่สามารถตรวจจับความผิดปกติในกระบวนการทางอุตสาหกรรมได้ด้วยเทคนิคการขยายข้อมูลเชิงเวลาอย่างมีประสิทธิภาพ

1.4.2 สร้างเทคนิคที่ช่วยเพิ่มขนาดและความหลากหลายของข้อมูล โดยมีการจัดเก็บข้อมูลจริงที่ลดลงจากเดิม

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะนำเสนอทฤษฎีที่เกี่ยวข้องกับการขยายข้อมูลในทางเวลาสำหรับการเรียนรู้เชิงลึกเพื่อใช้ในการตรวจจับความผิดปกติทางอุตสาหกรรม และการทบทวนวรรณกรรม ผู้วิจัยได้ทำการศึกษาเอกสาร ทฤษฎี และงานวิจัยต่างๆ ที่เกี่ยวข้อง โดยทฤษฎีที่เกี่ยวข้องมีดังต่อไปนี้

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 การเรียนรู้เชิงลึก (Deep learning)

Deep learning หรือ การเรียนรู้เชิงลึก เป็นสาขาย่อยของ Machine learning หรือการเรียนรู้ของเครื่อง ที่ได้รับแรงบันดาลใจจากการทำงานของสมองมนุษย์ โดยใช้โครงข่ายประสาทเทียม (Neural network) ที่มีโครงสร้างหลายชั้น (Deep) ในการเรียนรู้จากข้อมูล

หลักการทำงานของ Deep learning :

- โครงข่ายประสาทเทียม : ประกอบด้วย หน่วยประมวลผล (neuron) เชื่อมต่อกันเป็นชั้นๆ ทำหน้าที่คล้ายเซลล์ประสาทในสมอง
- การเรียนรู้ : โมเดล Deep learning เรียนรู้จากข้อมูลโดยปรับค่า พารามิเตอร์ (parameter) ของโครงข่ายประสาทเทียม ผ่านกระบวนการ Backpropagation
- การฝึกโมเดล : โมเดล Deep learning จำเป็นต้องได้รับการฝึกฝน (training) ด้วยข้อมูลจำนวนมาก เพื่อเรียนรู้รูปแบบ (pattern) ของข้อมูล
- การคาดการณ์: โมเดล Deep learning ที่ผ่านการฝึกฝนแล้ว สามารถนำไปใช้คาดการณ์ผลลัพธ์ (prediction) ของข้อมูลใหม่

ข้อดีของ Deep learning :

- มีประสิทธิภาพสูง
- เรียนรู้รูปแบบที่ซับซ้อน
- ปรับตัวให้เข้ากับข้อมูลใหม่

ข้อจำกัดของ Deep learning :

- ต้องการข้อมูลจำนวนมากในการฝึกโมเดล
- อาจมีอคติ (bias) ในการคาดการณ์

2.1.2 Convolutional Neural Network (CNN)

CNN ย่อมาจาก Convolutional Neural Network เป็นโมเดล Deep learning ประเภทหนึ่งที่เหมาะสำหรับงานที่เกี่ยวข้องกับภาพ (image) วิดีโอ (video) และข้อมูลเชิงเวลา (time series)

โครงสร้างของ CNN :

• Convolutional layer : ทำหน้าที่ดึงคุณสมบัติ (feature) ของข้อมูลประกอบด้วย filter หลายตัว แต่ละ filter เรียนรู้คุณสมบัติ (feature) เฉพาะ filter เลื่อนผ่านข้อมูล (input) คล้ายการกรอง (convolution) ผลลัพธ์ของ convolutional layer เรียกว่า feature map

• Pooling layer : ทำหน้าที่ลดขนาดของ feature map มีหลายวิธี เช่น Max pooling, Average pooling ช่วยลดจำนวนพารามิเตอร์ (parameter) ของโมเดล

• Fully connected layer : ทำหน้าที่จำแนกประเภท (classification) หรือคาดการณ์ผลลัพธ์ (regression) คล้ายกับโมเดล Artificial Neural Network (ANN) ทั่วไป

การทำงานของ CNN :

• ข้อมูล (input) ป้อนเข้าสู่ convolutional layer

• convolutional layer ดึงคุณสมบัติ (feature) ของข้อมูล

• pooling layer ลดขนาดของ feature map

• กระบวนการ convolutional layer และ pooling layer ทำซ้ำหลายชั้น

• fully connected layer ทำหน้าที่จำแนกประเภท (classification) หรือคาดการณ์ผลลัพธ์ (regression)

ประเภทของ CNN :

• LeNet : โมเดล CNN รุ่นแรก ใช้สำหรับการจำแนกประเภทตัวเลข

• AlexNet : โมเดล CNN ที่มีประสิทธิภาพสูง ใช้สำหรับการจำแนกประเภทภาพ

• VGGNet : โมเดล CNN ที่ใช้ convolutional layer หลายชั้น

• ResNet : โมเดล CNN ที่ใช้ skip connection ช่วยให้โมเดลเรียนรู้ได้ลึกขึ้น

• InceptionNet : โมเดล CNN ที่ใช้ parallel convolution ช่วยให้โมเดลเรียนรู้คุณสมบัติ (feature) หลายแบบ

ข้อดีของ CNN:

• มีประสิทธิภาพสูง

• เรียนรู้รูปแบบที่ซับซ้อน

• ทนทานต่อสัญญาณรบกวน

ข้อจำกัดของ CNN :

• ต้องการข้อมูลจำนวนมากในการฝึกโมเดล

• อาจมีอคติ (bias) ในการคาดการณ์

• ใช้ทรัพยากรในการประมวลผลสูง

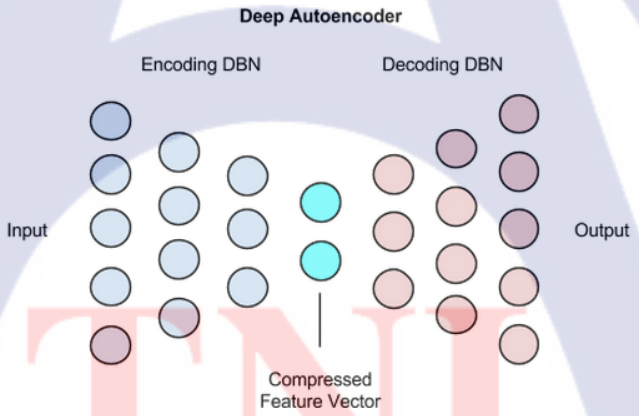
2.1.3 Deep Autoencoder

Deep Autoencoder เป็นโมเดล Deep learning ประเภทหนึ่งที่ใช้สำหรับ การลดมิติ (dimensionality reduction) การสร้างข้อมูล (data generation) และ การดึงคุณสมบัติ (feature extraction)

โครงสร้างของ Deep Autoencoder : ประกอบด้วย Encoder และ Decoder

• Encoder : แปลงข้อมูล (input) ไปยัง latent space เป็นมิติที่มีขนาดต่ำกว่าข้อมูล (input) ประกอบด้วย convolutional layer หรือ fully connected layer

• Decoder : แปลงข้อมูลจาก latent space กลับไปยังข้อมูล (output) ประกอบด้วย convolutional layer หรือ fully connected layer



รูปที่ 2.1 แสดงโครงสร้างพื้นฐานของ Deep Autoencoder ซึ่งประกอบด้วย: Encoder, Latent space, Decoder

การทำงานของ Deep Autoencoder :

• ข้อมูล (input) ป้อนเข้าสู่ encoder

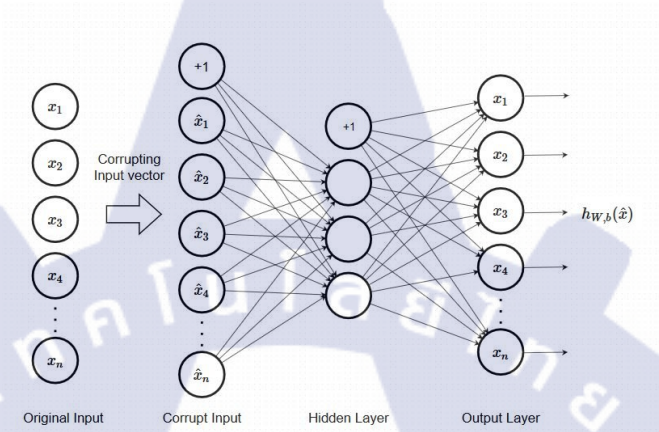
• encoder แปลงข้อมูล (input) ไปยัง latent space

• decoder แปลงข้อมูลจาก latent space กลับไปยังข้อมูล (output)

• โมเดล Deep Autoencoder เรียนรู้จาก loss function ช่วยให้โมเดลลดข้อผิดพลาดระหว่างข้อมูล (input) และข้อมูล (output)

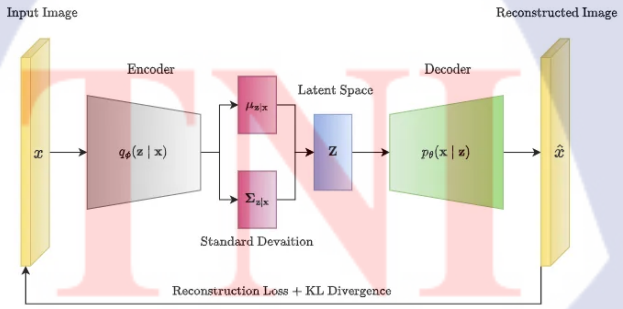
ประเภทของ Deep Autoencoder :

• Denoising Autoencoder: เพิ่ม noise ให้กับข้อมูล (input)



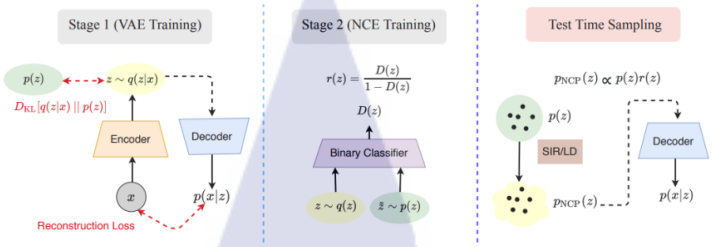
รูปที่ 2.2 Denoising Autoencoder

• Variational Autoencoder: ใช้ probability distribution ในการแปลงข้อมูล



รูปที่ 2.3 Variational Autoencoder

• Contrastive Autoencoder: เปรียบเทียบข้อมูล (input) กับข้อมูลอื่น



รูปที่ 2.4 Contrastive Autoencoder

ข้อดีของ Deep Autoencoder :

- เรียนรู้รูปแบบที่ซับซ้อน
- ทนทานต่อสัญญาณรบกวน
- ใช้ทรัพยากรในการประมวลผลน้อย

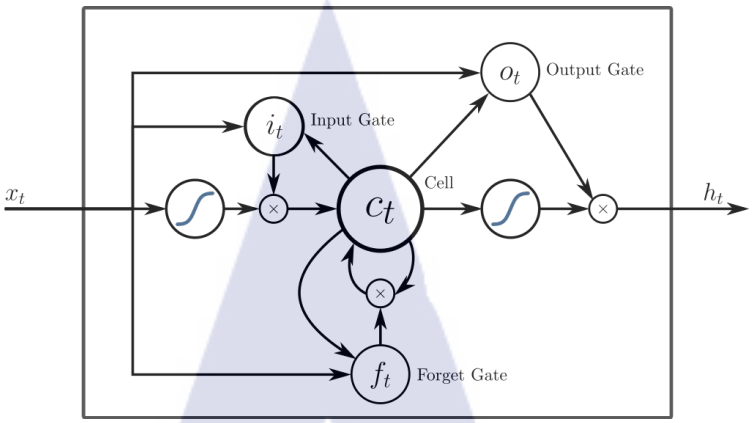
ข้อจำกัดของ Deep Autoencoder :

- ต้องการข้อมูลจำนวนมากในการฝึกโมเดล
- อาจมีอคติ (bias) ในการคาดการณ์

2.1.4 Long Short-Term Memory (LSTM)

LSTM ย่อมาจาก Long Short-Term Memory เป็นโมเดล Deep learning ประเภทหนึ่งที่ใช้สำหรับงานที่เกี่ยวข้องกับ ลำดับ (sequence) ของข้อมูล เช่น ข้อความ (text) เสียงพูด (speech) และข้อมูลเชิงเวลา (time series) ซึ่งมีจุดเด่นสามารถจดจำข้อมูลในอดีตได้นานกว่า Recurrent Neural Network (RNN) ทั่วไป เหมาะสำหรับงานที่ข้อมูลมีความสัมพันธ์กันยาวนานโดยโครงสร้างนั้นจะประกอบด้วย cell หลายตัว และแต่ละ cell จะเก็บ state ของข้อมูลที่ประกอบด้วย hidden state และ cell state ซึ่ง hidden state เก็บข้อมูลสรุปของข้อมูลในอดีต ส่วน cell state เก็บข้อมูลดิบของข้อมูลในอดีต แต่ละ cell มี gate 3 ตัว ควบคุมการไหลของข้อมูลดังนี้

- forget gate: ควบคุมการลืมข้อมูล
- input gate: ควบคุมการรับข้อมูลใหม่
- output gate: ควบคุมการส่งออกข้อมูล



รูปที่ 2.5 แสดงโครงสร้างพื้นฐานของ LSTM

โดยขั้นตอนการทำงานจะประกอบไปด้วย

• ข้อมูล (input) ถูกป้อนเข้าสู่ LSTM

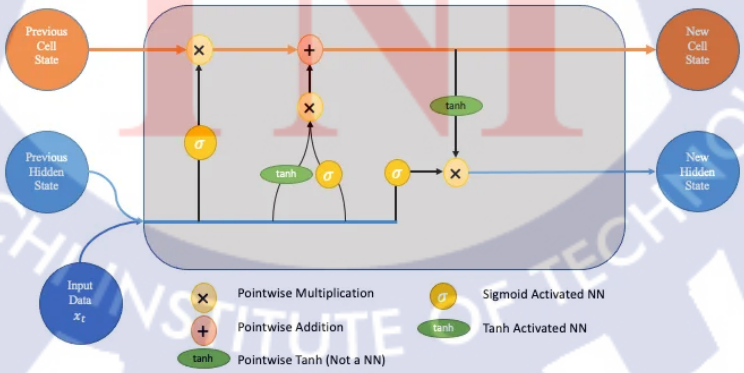
• LSTM ทำการประมวลผลข้อมูล (input) ผ่าน cell แต่ละตัว

• forget gate ทำหน้าที่ควบคุมการลืมข้อมูล ด้วยฟังก์ชัน sigmoid มีค่า output อยู่ระหว่าง 0 ถึง 1 โดยค่า 0 หมายถึง ลืมข้อมูลทั้งหมดและ 1 หมายถึง เก็บข้อมูลทั้งหมด

• input gate ควบคุมการรับข้อมูลใหม่ประกอบด้วย 2 ส่วน คือ sigmoid layer จะควบคุมว่าจะรับข้อมูลใหม่หรือไม่ และ tanh layer ทำหน้าที่ควบคุมค่าข้อมูลใหม่ที่จะเก็บ

• output gate ควบคุมการส่งออกข้อมูล ด้วยฟังก์ชัน sigmoid มีค่า output อยู่ระหว่าง 0 ถึง 1 โดยค่า 0 หมายถึง ไม่ออกข้อมูล และ 1 หมายถึง ส่งออกข้อมูลทั้งหมด

• การเรียนรู้จาก loss function ช่วยให้โมเดลลดข้อผิดพลาด



รูปที่ 2.6 แสดงการไหลของข้อมูลและการทำงานของ LSTM

2.1.5 Time Series Augmentation (Tsaug)

เทคนิค Tsaug เป็นวิธีการสร้างข้อมูลจำลองแบบใหม่เพิ่มเติมสำหรับงาน Time Series โดยมีวัตถุประสงค์เพื่อ เพิ่มขนาดชุดข้อมูล (data augmentation), การป้องกัน overfitting และเพิ่มประสิทธิภาพของโมเดล Time Series โดยแบ่งเป็นเทคนิคหลัก และเทคนิคเฉพาะดังนี้

- เทคนิค Augmentation หลักประกอบด้วย :
  - การเพิ่มสัญญาณรบกวน (Noise Augmentation) :
    - เพิ่มสัญญาณรบกวนแบบ Gaussian noise ให้กับข้อมูล
  - การแปลงข้อมูล (Data Transformation):
    - หาค่าเฉลี่ยเคลื่อนที่ (moving average)
    - หาความแตกต่าง (differencing)
    - หาค่าเฉลี่ยสะสม (cumulative sum)
  - การสุ่มตัวอย่าง (Sampling):
    - สุ่มตัวอย่างแบบสุ่ม (random sampling)
    - สุ่มตัวอย่างแบบมีชั้น (stratified sampling)
- เทคนิค Augmentation เฉพาะสำหรับงาน Time Series ประกอบด้วย:
  - การขยายข้อมูล (Data Stretching): ขยายหรือหดช่วงเวลาของข้อมูล
  - การบิดเบือนข้อมูล (Data Warping): บิดเบือนรูปร่างของข้อมูล
  - การผสมผสานข้อมูล (Data Mixing): ผสมผสานข้อมูลจากชุดข้อมูลเดียวกัน (intra-dataset mixing)

2.2 การทบทวนวรรณกรรม

ผู้วิจัยได้ทำการทบทวนงางวิจัยที่เกี่ยวข้องโดยการหาข้อมูลจากฐานข้อมูลมาตรฐานต่างๆ เช่น จากเว็บไซต์ www.sciencedirect.com และทำการหาบทความต่างๆ เพิ่มเติมที่เกี่ยวข้องกับหัวข้องานวิจัย ซึ่งได้พบงานวิจัยที่เกี่ยวข้อง สรุปได้ตามตารางที่ 2.1 ดังต่อไปนี้

ตารางที่ 2.1 สรุปงานวิจัยที่เกี่ยวข้อง

ลำดับ	ผู้แต่ง	ปีที่พิมพ์	ชื่อเรื่อง
1	Y. Jiang et al. [1]	2019	A Machine Vision-based Realtime Anomaly Detection Method for Industrial Products Using Deep Learning
2	M. Ghazali et al. [2]	2021	Vibration analysis for machine monitoring and diagnosis: a systematic review. Shock and Vibration
3	X. Xu et al. [5]	2020	LSTM-GAN-XGBOOST Based Anomaly Detection Algorithm for Time Series Data
4	O. Mey et al. [6]	2020	Machine Learning-Based Unbalance Detection of a Rotating Shaft Using Vibration Data

2 . 2.1 A Machine Vision-based Realtime Anomaly Detection Method for Industrial Products Using Deep Learning

บทความนี้เจาะลึกวิธีการตรวจจับความผิดปกติในผลิตภัณฑ์อุตสาหกรรมโดยใช้เทคโนโลยี Deep Learning นำเสนอโมเดลตรวจจับความผิดปกติแบบมีผู้สอน (Supervised) ที่ใช้ YOLOv3 โมเดลนี้มีตัวจำแนก ROI (Region of Interest) ที่ออกแบบมาเพื่อตรวจจับประเภทความผิดปกติโดยเฉพาะ สำหรับข้อมูลภาพที่มีความสมดุล และ นำเสนอโมเดลตรวจจับความผิดปกติแบบกึ่งมีผู้สอน (Semi-supervised) ที่ใช้ Fast-AnoGAN โดยโมเดลนี้จะถูกฝึกจากภาพปกติเท่านั้น โมเดลจะใช้ WGAN-GP ที่ได้รับการฝึกแล้วเพื่อสร้างภาพจำลอง และทำการตรวจจับความผิดปกติโดยการคำนวณ "คะแนนความผิดปกติ" (Anomaly Score) ซึ่งเกิดจากความแตกต่างระหว่างภาพจำลองและภาพที่ใช้ในการทดสอบ สำหรับข้อมูลภาพที่มีความสมดุล ซึ่งได้ทำการทดสอบกับชุดข้อมูลภาพที่มีทั้งความสมดุลและไม่สมดุล ภายใต้สถานการณ์การผลิตอุตสาหกรรมจริง ผลการประเมินแสดงให้เห็นว่าทั้งสองโมเดลที่ใช้ Deep Learning สามารถตอบสนองความต้องการด้านความแม่นยำและการตรวจจับแบบเรียลไทม์ (real-time) สำหรับการตรวจจับความผิดปกติในการผลิตที่ต้องใช้ความเร็วสูง

2.2.2 Vibration analysis for machine monitoring and diagnosis: a systematic review. Shock and Vibration

บทความนี้ นำเสนอการวิเคราะห์การสั่นสะเทือน (Vibration analysis) เพื่อใช้ในการตรวจสอบและวินิจฉัยสภาพเครื่องจักร (machine monitoring and diagnosis) โดยใช้แนวทางการทบทวนวรรณกรรมอย่างเป็นระบบ โดยมุ่งเน้นไปที่ประเด็นสำคัญดังนี้

• ความสำคัญของการวิเคราะห์การสั่นสะเทือน: การสั่นสะเทือนเป็นสัญญาณเตือนเบื้องต้นที่สำคัญของความผิดปกติในเครื่องจักร การวิเคราะห์การสั่นสะเทือนช่วยในการตรวจจับปัญหาตั้งแต่เนิ่นๆ ป้องกันความเสียหายร้ายแรง ลดต้นทุนการซ่อมบำรุง และยืดอายุการใช้งานของเครื่องจักร

• การเก็บข้อมูล (data acquisition) ประกอบด้วย เครื่องมือวัด (sensor) และเครื่องวิเคราะห์ (analyzer) , การประมวลผลสัญญาณ (signal processing) เพื่อแยกแยะสัญญาณที่เกี่ยวข้องกับความผิดปกติ , การวินิจฉัยความผิดปกติ (fault recognition) โดยใช้เทคนิคต่างๆ เช่น การวิเคราะห์ความถี่ (frequency analysis) และการวิเคราะห์เวลา (time domain analysis)

• เทคนิคการวิเคราะห์สัญญาณ: บทความกล่าวถึงเทคนิคต่างๆ ที่ใช้ในการประมวลผลสัญญาณการสั่นสะเทือน เช่น การวิเคราะห์ความถี่ (Fast Fourier Transform - FFT) และการวิเคราะห์สถิติ (Statistical analysis)

• ปัญญาประดิษฐ์ (AI) ในการวิเคราะห์การสั่นสะเทือน: บทความเน้นย้ำถึงบทบาทที่เพิ่มขึ้นของปัญญาประดิษฐ์ (AI) โดยเฉพาะอย่างยิ่ง การเรียนรู้เชิงเครื่อง (Machine Learning) ในการวินิจฉัยความผิดปกติของเครื่องจักร

2.2.3 LSTM-GAN-XGBOOST Based Anomaly Detection Algorithm for Time Series Data

บทความนี้เสนอวิธีการตรวจจับความผิดปกติของข้อมูลอนุกรมเวลา ซึ่งเป็นงานพื้นฐานที่สำคัญสำหรับระบบพยากรณ์และการจัดการสุขภาพ (PHM) วิธีการดั้งเดิม มักประสบปัญหาในการตรวจจับคุณสมบัติเชิงลึกของข้อมูลอนุกรมเวลาขนาดใหญ่ งานวิจัยนี้เสนอ อัลกอริทึมผสมผสานแบบ LSTM-GAN-XGBOOST ที่อาศัยข้อดีของเครือข่ายประสาทเทียม (ANNs) และการเรียนรู้แบบผสม (EL) ซึ่งประกอบไปด้วย

• LSTM: สำหรับดึงคุณสมบัติตามมิติเวลาของข้อมูล

• GAN: สำหรับดึงคุณสมบัติเชิงลึกของข้อมูลปกติอย่างมีประสิทธิภาพ

• XGBOOST: สำหรับจำแนกคุณสมบัติที่ดึงออกมาและส่งออกคะแนนความผิดปกติ

ซึ่งผลการทดลองยืนยันประสิทธิภาพของวิธีการที่เสนอ โดยสามารถตรวจจับความผิดปกติของข้อมูลชุดแบริ่งลูกปืนได้ดีกว่า สามารถตรวจจับความผิดปกติได้แม่นยำ (AUC 99.1%)และมีประสิทธิภาพเหนือกว่าวิธีการดั้งเดิม

2.2.4 Machine Learning-Based Unbalance Detection of a Rotating Shaft Using Vibration Data

บทความนี้ได้ศึกษาเกี่ยวกับการวิเคราะห์การสั่นสะเทือนของเครื่องจักรหมุนด้วยเซนเซอร์ตรวจจับแรงสั่นสะเทือนเป็นวิธีที่มีประสิทธิภาพในการตรวจจับความผิดปกติหรือความเสียหายของส่วนประกอบเครื่องจักรแต่เนิ่นๆ ช่วยหลีกเลี่ยงการหยุดทำงานของเครื่องจักร ในการวิจัยนี้เราขอนำเสนอชุดข้อมูลที่สามารถใช้สำหรับพัฒนาและประเมินวิธีการตรวจจับความผิดปกติที่เกิดจากการไม่สมดุล (unbalance) ในเครื่องจักรโดยทดลองวิจัยสร้างความไม่สมดุลในระดับต่างๆ บนเพลาหมุนโดยใช้ชิ้นส่วนพิมพ์ 3 มิติและทำการบันทึกข้อมูลการสั่นสะเทือนผ่านเซนเซอร์ 3 ตัว ด้วยอัตราตัวอย่าง (sampling rate) 4096 ค่าต่อวินาที ขณะที่ความเร็วของเพลาหมุนอยู่ระหว่าง 630 - 2330 รอบต่อนาทีได้จัดเตรียมชุดข้อมูลไว้ทั้งส่วนที่เป็นข้อมูลสำหรับพัฒนาอัลกอริทึม และข้อมูลสำหรับการประเมินประสิทธิภาพโดยการวิเคราะห์ด้วย

- โครงข่ายประสาทเทียมเชื่อมต่อเต็มชั้น (fully connected neural networks)
- โครงข่ายประสาทเทียมคอนโวลูชั่น (convolutional neural networks)
- โมเดลซ่อนมาร์คอฟ (Hidden Markov Models)
- การจำแนกประเภทแบบสุ่มป่า (Random Forest Classifications)

ผลลัพธ์ปรากฏว่าโมเดลโครงข่ายประสาทเทียมเชื่อมต่อเต็มชั้น (fully-connected neural network) โดยใช้วิธีปรับสเกลและทำ FFT transform กับข้อมูลการสั่นสะเทือน ให้ความแม่นยำในการคาดการณ์ถึง 98.6% บนชุดข้อมูลที่ใช้ในการประเมิน

บทที่ 3
วิธีการวิจัย

ในการศึกษาวิจัยนี้มุ่งเน้นไปที่การพัฒนาเทคนิคการขยายข้อมูลในทางเวลา (Temporal Data Augmentation) สำหรับการเรียนรู้เชิงลึก เพื่อเพิ่มประสิทธิภาพการตรวจจับความผิดปกติในเครื่องจักรอุตสาหกรรม เทคนิคนี้จะช่วยสร้างข้อมูลจำลองเพิ่มเติมจากข้อมูลที่มีอยู่ ช่วยให้โมเดล Deep Learning เรียนรู้รูปแบบ temporal patterns ได้ดีขึ้น และสามารถตรวจจับความผิดปกติที่หายากได้แม่นยำยิ่งขึ้น

เป้าหมายของงานวิจัยนี้พัฒนาเทคนิคการขยายข้อมูลในทางเวลาที่เหมาะสมสำหรับการตรวจจับความผิดปกติในเครื่องจักรอุตสาหกรรมเปรียบเทียบประสิทธิภาพของเทคนิคการขยายข้อมูลกับโมเดล Deep Learning พัฒนาระบบตรวจจับความผิดปกติแบบเรียลไทม์โดยใช้เทคนิคที่เสนอโดยประโยชน์ที่คาดหวังเพื่อเพิ่มประสิทธิภาพการตรวจจับความผิดปกติในเครื่องจักรอุตสาหกรรมลดความเสี่ยงของการเกิดอุบัติเหตุและ downtime เพิ่มประสิทธิภาพการทำงานและ productivity ของระบบอุตสาหกรรมโดยขอบเขตของงานวิจัยนี้มุ่งเน้นไปที่การพัฒนาเทคนิคการขยายข้อมูลในทางเวลาสำหรับโมเดล Deep Learning ประเภท CNNs และ RNN โดยเนื้อหาบทนี้จะกล่าวถึงวิธีการและขั้นตอนการดำเนินงานวิจัย ดังต่อไปนี้

3.1 แผนการดำเนินงานวิจัย

ตารางที่ 3.1 แผนงานและระยะเวลาในการดำเนินงาน

ขั้นตอนวิธีการดำเนินงาน	2565					2566										2567		
	8	9	10	11	12	1	2	3	6	7	8	9	10	11	12	1	2	3
1. ศึกษาปัญหา และเทคโนโลยีที่จะนำมาใช้	■																	
2. ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง	■	■	■															
3. สอบโครงร่างวิทยานิพนธ์			■															
4. ออกแบบระบบการทำงาน			■	■														
5. พัฒนาระบบการทำงาน					■	■	■	■	■									
6. ทดสอบการทำงานของระบบ										■	■	■	■					
7. สอบความก้าวหน้า													■					
8. เก็บข้อมูลการติดตาม														■	■			
9. ส่งบทความทางวิชาการ															■			
10. วิเคราะห์ผลข้อมูล															■	■	■	
11. สรุปผลงานวิจัยและจัดทำเล่มวิทยานิพนธ์																	■	■
12. สอบป้องกันวิทยานิพนธ์																		■

3.2 ขั้นตอนการดำเนินงาน

ผู้วิจัยได้ทำการศึกษา และหาข้อมูลของงานวิจัย โดยมีขั้นตอนการดำเนินงานดังนี้

ศึกษาข้อมูลปัญหา
และเทคโนโลยีที่จะนำมาใช้

ศึกษาทฤษฎี
และงานวิจัยที่เกี่ยวข้อง

ออกแบบระบบการขยายข้อมูล

ทดสอบประสิทธิภาพ

ไม่ผ่าน

วิเคราะห์ข้อมูล

ผ่าน

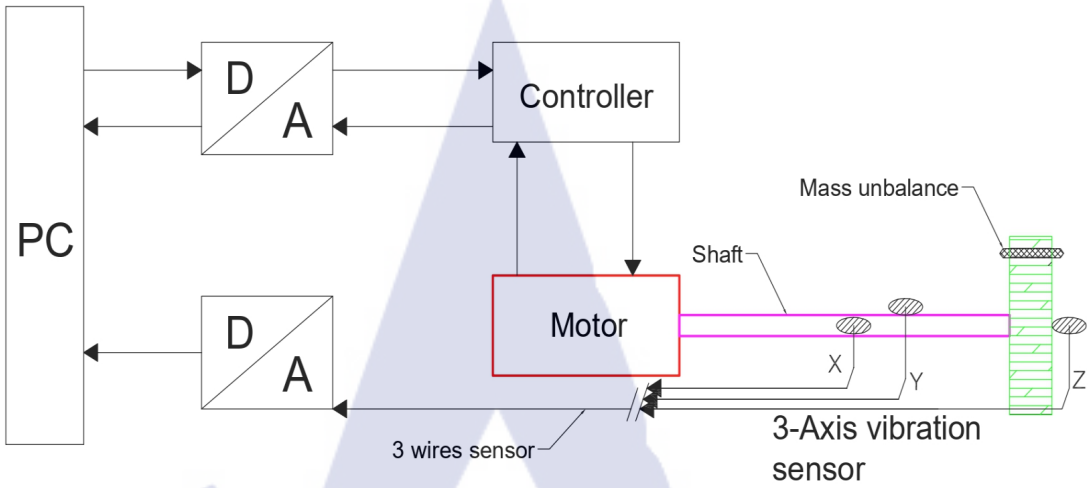
สรุปผลงานวิจัย

รูปที่ 3.1 แผนผังขั้นตอนการดำเนินงาน

3.3 การออกแบบการทดลอง

3.3.1 การวิเคราะห์ชุดข้อมูลที่ใช้ในงานวิจัย

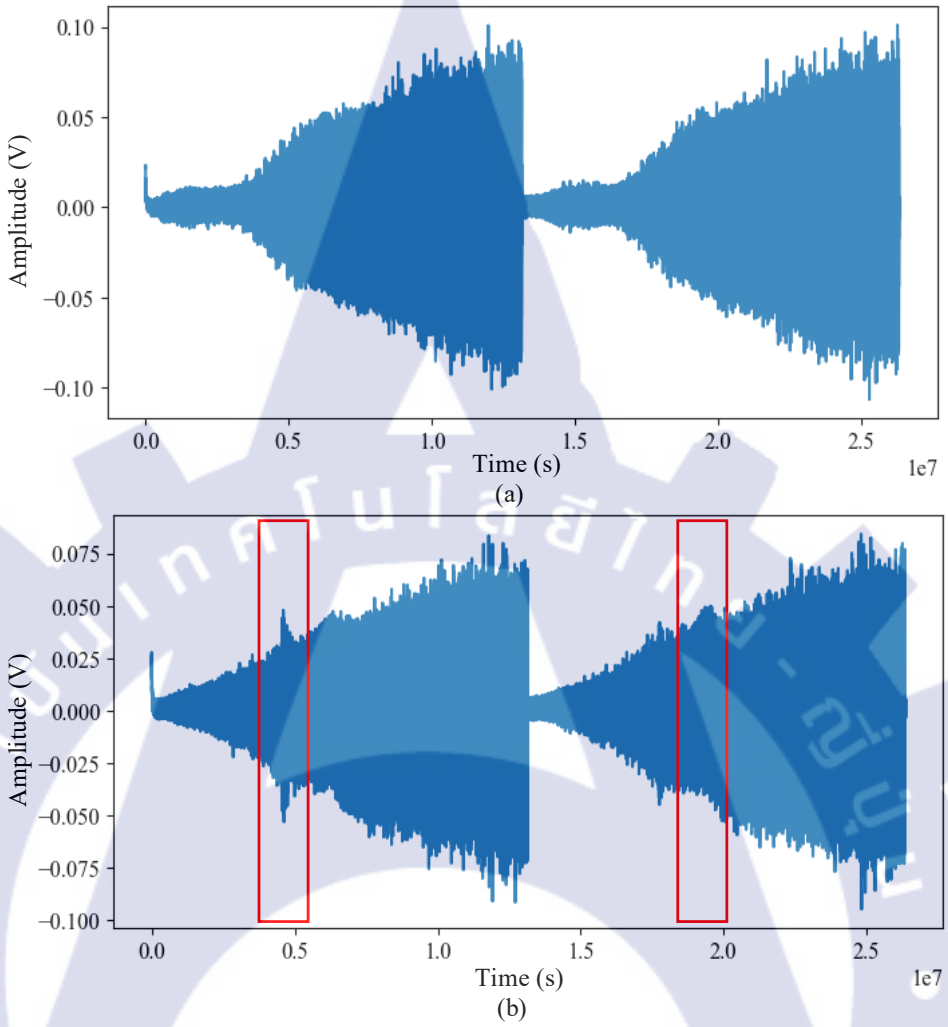
ชุดข้อมูล (Dataset) ที่ใช้ในงานวิจัยประกอบด้วยข้อมูลการสั่นสะเทือนจากเซ็นเซอร์ (PCB Synotech GmbH, type PCB-M607A11 / M001AC) ที่บันทึกไว้บนระบบขับเคลื่อนแบบหมุนของมอเตอร์กระแสตรงแบบเปลี่ยนขั้วอิเล็กทรอนิกส์ (WEG GmbH, type UE 511 T) มอเตอร์ทำหน้าที่ขับเคลื่อนเพลาที่มีเส้นผ่านศูนย์กลาง 12 มม. เพลานี้จะถูกสอดผ่านตลับลูกปืนลูกกลิ้งที่ยึดไว้ในบล็อกลูกปืน ที่จับตัวถ่วงไม่สมดุล (unbalance holder) จะติดตั้งอยู่ด้านหลังของบล็อกลูกปืนโดยตรง ชิ้นส่วนที่จับตัวถ่วงไม่สมดุลนี้ผลิตด้วยเครื่องพิมพ์ 3 มิติ ประกอบด้วยแผ่นดิสก์ (เส้นผ่านศูนย์กลาง: 52 มม.) ที่มีช่องเว้าสมมาตรตามแนวแกน สามารถใส่ตัวถ่วงเข้าไปในช่องเหล่านี้เพื่อจำลองสภาวะการไม่สมดุลได้ รูปที่ 1 แสดงภาพรวมการเก็บข้อมูลความผิดปกติจากระบบขับเคลื่อนมอเตอร์ ชุดข้อมูลนี้สามารถนำไปใช้ในการพัฒนาและทดสอบอัลกอริทึมสำหรับการตรวจจับการไม่สมดุลของระบบขับเคลื่อนโดยอัตโนมัติ โดยจะมีการบันทึกชุดข้อมูลสำหรับรูปแบบการไม่สมดุลของเพลา (Unbalance) ที่มีขนาดต่างกัน 4 แบบและสำหรับกรณีที่เพลาสมดุล (Balance) ข้อมูลการสั่นสะเทือนถูกบันทึกด้วยความถี่ในการสุ่มตัวอย่าง (sampling rate) 4096 ค่าต่อวินาที ตารางที่ 3.2 แสดงชุดข้อมูลสำหรับการฝึกสอนโมเดล (ID: D[0-4]) และชุดข้อมูลสำหรับการประเมิน (ID: E[0-4]) ที่ใช้สำหรับแต่ละระดับของความไม่สมดุล สำหรับการวัดแต่ละครั้งของชุดข้อมูลการฝึกสอนโมเดลจะมีข้อมูลการวัดอย่างต่อเนื่องประมาณ 107 นาที ในขณะที่ข้อมูลการวัดแต่ละครั้งของชุดข้อมูลการประเมินจะมีความยาว 28 นาที รูปที่ 2 แสดงการเปรียบเทียบสัญญาณในทางเวลาจากเซ็นเซอร์วัดการสั่นสะเทือนที่ติดกับเพลาของมอเตอร์ รูป 2(a) คือ สัญญาณในกรณีที่ไม่มีการถ่วงน้ำหนักที่เพลามอเตอร์ (0D) และ 2(b) คือสัญญาณในกรณีที่มีการถ่วงน้ำหนักที่เพลามอเตอร์ (1D) จะเห็นได้ว่าสัญญาณทางเวลาในกรณีที่มีการถ่วงน้ำหนักที่เพลามอเตอร์จะมีแอมพลิจูดที่ผิดปกติเกิดขึ้นอย่างเห็นได้ชัด (กรอบสีแดง) แต่ในส่วนอื่นของสัญญาณไม่มีความแตกต่างมากนัก ซึ่งในความต่างของสัญญาณระดับนี้สามารถใช้เทคนิคการขยายขนาดข้อมูลทางเวลาในการจำลองสัญญาณเพื่อใช้ในการฝึกฝนโมเดลการเรียนรู้เชิงลึกได้



รูปที่ 3.2 ภาพรวมการเก็บข้อมูลความผิดปกติจากระบบขับเคลื่อนมอเตอร์

ตารางที่ 3.2 รายละเอียดชุดข้อมูลที่ใช้ในงานวิจัย

ID	Mass (g)	Number of Samples		Labels
		Train	Test	
0D/0E	0	6438	1670	balance
1D/1E	3.281	6434	1673	unbalance
2D/2E	3.281	6434	1669	unbalance
3D/3E	3.281	6430	1672	unbalance
4D/4E	6.614	6430	1675	unbalance



รูปที่ 3.3 การเปรียบเทียบสัญญาณในทางเวลาจากเซ็นเซอร์วัดการสั่นสะเทือนที่ติดกับเพลาของมอเตอร์ 2(a) สัญญาณในกรณีที่ไม่มีการถ่วงน้ำหนักที่เพลามอเตอร์ (0D) และ 2(b) สัญญาณในกรณีที่มีการถ่วงน้ำหนักที่เพลามอเตอร์ (1D)

3.3.2 การขยายขนาดข้อมูลในเชิงเวลา (Time-Series Data Augmentation)

การขยายข้อมูลเชิงเวลา (Time-Series Data Augmentation) สำหรับการเรียนรู้เชิงลึก ถือเป็นเทคนิคสำคัญที่ออกแบบมาเพื่อเพิ่มประสิทธิภาพของโมเดล โดยเฉพาะอย่างยิ่งในสถานการณ์ที่มีข้อมูลในจำนวนจำกัดหรือมีข้อจำกัดในการเก็บข้อมูล วิธีการนี้เป็นการเพิ่มขนาดและความหลากหลายของชุดข้อมูลเชิงเวลาอย่างไม่เป็นธรรมชาติ เพื่อปรับปรุงความทนทานและความสามารถในการทำให้โมเดลการเรียนรู้เชิงลึกสามารถใช้ได้ในสถานการณ์ทั่วไปมากขึ้น เทคนิคนี้สามารถสังเคราะห์ตัวอย่างข้อมูลสำหรับการฝึกอบรมใหม่ ๆ จากชุดข้อมูลต้นฉบับ โดยใช้การแปลงข้อมูล (transformations) หลายชุดที่ยังคงรักษาโครงสร้างและลักษณะพื้นฐานของข้อมูลเอาไว้ ในกรณีที่เป็นข้อมูลเชิงเวลา ซึ่งเป็นจุดข้อมูลที่ถูกรวบรวมหรือบันทึกไว้ในช่วงเวลาต่อเนื่องกัน การขยายข้อมูลจะต้องคงไว้ซึ่งพลวัตชั่วคราว (temporal dynamics) และความสัมพันธ์ที่อยู่ภายในลำดับข้อมูลนั้น เทคนิคการขยายขนาดข้อมูลเชิงเวลาที่งานวิจัยนี้ใช้มีดังนี้

• การเบี่ยงเบนทางเวลา (Drift) เป็นดึงข้อมูลดั้งเดิมในชุดเวลา ให้เบี่ยงเบนออกไปแบบสุ่มและราบเรียบ ระดับความเบี่ยงเบนนี้ ถูกควบคุมด้วยขนาดการเบี่ยงเบนสูงสุด (maximal drift) และจำนวนจุดเบี่ยงเบน (number of drift points)

• การบิดเบี้ยวทางเวลา (Time Warping) เป็นการปรับเปลี่ยนความเร็วของช่วงเวลาแบบสุ่ม โดยการบิดเบี้ยวเวลา ซึ่งสามารถควบคุมได้ด้วยจำนวนการปรับความเร็ว (number of speed changes) และอัตราส่วนระหว่างความเร็วสูงสุดกับต่ำสุด (maximal ratio of max/min speed)

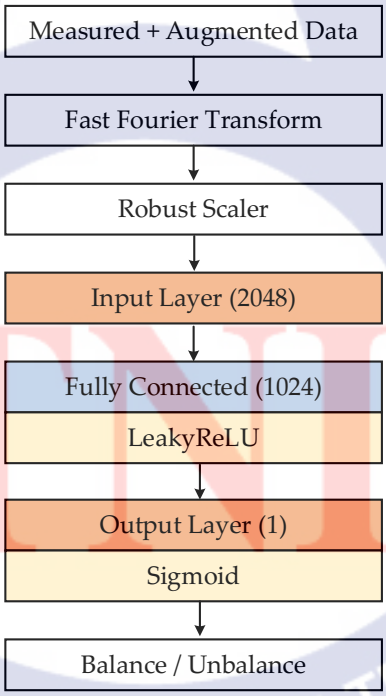
• สร้างสัญญาณรบกวน (Add Noise) เป็นการเพิ่มสัญญาณรบกวนแบบสุ่มให้กับชุดข้อมูลเชิงเวลา สัญญาณรบกวนที่ถูกเพิ่มในแต่ละจุดของข้อมูลนั้น จะเป็นอิสระ (independent) และมีการกระจายตัวทางสถิติที่เหมือนกัน (identically distributed)

• การลดความละเอียดของเวลา (Pool) วิธีนี้คล้ายกับการลดขนาด ของภาพ โดยรวมจุดข้อมูลหลายจุดเข้าด้วยกัน ทำให้ชุดข้อมูลมีจำนวนจุดน้อยลง แต่ครอบคลุมช่วงเวลาเดิม

ด้วยการผสมผสานเทคนิคที่กล่าวมาข้างต้นทำให้สามารถสังเคราะห์ข้อมูลทางเวลาจากข้อมูลดั้งเดิมให้มีความแตกต่างโดยที่ยังคงไว้ซึ่งโครงสร้างของข้อมูลต้นทางไว้ได้

3.3.3 โมเดลการเรียนรู้เชิงลึก

เนื่องจากการวิเคราะห์ในเชิงเวลาเพียงอย่างเดียวอาจไม่สามารถทำให้โมเดลการเรียนรู้เชิงลึกวิเคราะห์สิ่งผิดปกติที่ซ่อนอยู่ในข้อมูลได้ งานวิจัยนี้จึงเลือกใช้การแปลงฟูเรียร์เร็ว (FFT - Fast Fourier Transform) สำหรับแต่ละหน้าต่างความยาวหนึ่งวินาที หรือ 4096 ค่า จากชุดข้อมูลสัญญาณการสั่นสะเทือน ตามทฤษฎีสุ่มตัวอย่างแบบแชนนอน-ไนควิสต์ (Shannon-Nyquist Sampling Theorem) ขั้นตอนนี้จะได้สัมประสิทธิ์ฟูเรียร์ (Fourier coefficients) จำนวน 2048 ค่า สำหรับแต่ละหน้าต่างซึ่งเป็นข้อมูลที่มีความหมายทางกายภาพ โดยการแปลงฟูเรียร์อย่างเร็วจะเผยให้เห็นถึงรูปแบบข้อมูลที่ไม่สามารถสังเกตได้ง่าย ในโดเมนเวลา ตัวอย่างเช่น ลักษณะการเกิดซ้ำ (periodicity) และความถี่หลักภายในสัญญาณจะเป็นสิ่งที่ชัดเจนขึ้น กระบวนการนี้ยังทำให้การระบุและกรองเสียงรบกวน (Noise) หรือส่วนประกอบความถี่ที่ไม่ต้องการออกจากข้อมูลทำได้ง่ายขึ้น และช่วยยกระดับอัตราส่วนสัญญาณต่อเสียงรบกวน (signal-to-noise ratio) ซึ่งสามารถนำไปใช้ในการจำแนกประเภทข้อมูลได้ ทำให้รูปแบบข้อมูลที่แฝงอยู่ชัดเจนมากขึ้น และง่ายต่อการตรวจจับและเรียนรู้สำหรับโมเดลการเรียนรู้เชิงลึก



รูปที่ 3.4 ขั้นตอนการตรวจจับสัญญาณความผิดปกติโดยใช้โครงข่ายประสาทเทียมแบบ Fully Connected

รูปที่ 3.4 แสดงขั้นตอนการตรวจจับสัญญาณความผิดปกติโดยใช้โครงข่ายประสาทเทียมแบบ Fully Connected ชุดข้อมูลสำหรับพัฒนาที่ถูกแปลงผ่านวิธีการนี้ จะถูกแบ่งออกแบบสุ่มเป็น 80% สำหรับการฝึกโมเดล และ 20% สำหรับการทดสอบโมเดล จากนั้นเครือข่ายประสาทเทียมแบบเชื่อมต่อทั้งหมด (Fully Connected Neural Networks) ได้ถูกนำมาใช้ในการตรวจจับความผิดปกติในข้อมูล ภาพประกอบของสถาปัตยกรรมเครือข่ายที่ใช้แสดงไว้ในรูปที่ 3 โดยที่อินพุตประกอบด้วยสัมประสิทธิ์ฟูเรียร์ 2048 ตัวในแต่ละตัวอย่าง และตามด้วยชั้นที่ซ่อนอยู่ (hidden layers) จำนวน 4 ชั้น ซึ่งเป็นชั้นแบบเชื่อมต่อทั้งหมด (fully connected layers) ที่มีฟังก์ชั่นกระตุ้น (activation function) แบบ LeakyReLU และปิดท้ายด้วยชั้นเอาต์พุต (output layer) ที่มีฟังก์ชั่นกระตุ้นแบบ sigmoid

3.3.4 ลำดับขั้นตอนการทดลองด้วยโปรแกรม

ในการวิจัยนี้ได้ทำการทดลองโดยใช้โปรแกรม Python ในการศึกษาโดยแบ่งลำดับขั้นตอนการทำงานไว้เป็นส่วนดังนี้

1) ขั้นตอนการโหลดและจัดการข้อมูล มีดังนี้ :

a. การ Import Library :
   i. Pandas : สำหรับจัดการและวิเคราะห์ข้อมูล
   ii. matplotlib.pyplot : สำหรับสร้างกราฟ
   iii. numpy : สำหรับคำนวณทางตัวเลข
   iv. tensorflow : สำหรับสร้างและฝึกโมเดลเครือข่ายประสาทเทียม
   v. zipfile : สำหรับอ่านและเขียนไฟล์ zip
   vi. sklearn.preprocessing.RobustScaler : สำหรับปรับขนาดข้อมูลให้ทนต่อค่าผิดปกติ

b. กำหนดตัวแปร :
   i. url : เส้นทางไปยังไฟล์ข้อมูล zip
   ii. use_reference_models : ใช้โมเดลอ้างอิงหรือไม่ (True/False)
   iii. model_path : เส้นทางสำหรับเก็บโมเดลที่ฝึกเสร็จ
   iv. labels : พจนานุกรมแปลงชื่อคลาสเป็นตัวเลข
   v. sensor : ชื่อเซ็นเซอร์ที่เก็บข้อมูลการสั่นสะเทือน
   vi. samples_per_second : จำนวนตัวอย่างต่อวินาทีในข้อมูล
   vii. seconds_per_analysis : ระยะเวลาของแต่ละหน้าต่างวิเคราะห์ (วินาที)

viii. window : ขนาดของหน้าต่างวิเคราะห์ (จำนวนตัวอย่าง)

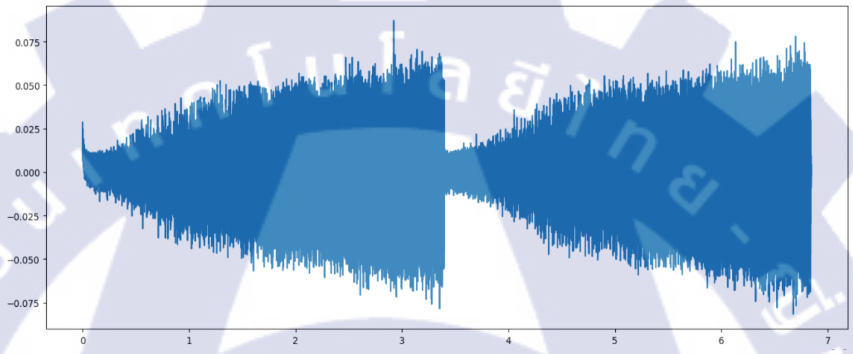
c. โหลดและแยกข้อมูล : X0D, X1D, X3D, X0E, X1E, X2E, X3E, X4E: อาเรย์ข้อมูลการสั่นสะเทือนจากแหล่งต่างๆ (data0D, data1D, etc.) อาจมาจากเซ็นเซอร์ต่างกันหรือช่วงเวลาต่างกัน ทำการตัดข้อมูล[50000:len(data0D[sensor])] ชี้ว่าดึงข้อมูลเฉพาะส่วนมาวิเคราะห์ ตั้งแต่ตำแหน่งที่ 50000 จนถึงปลายข้อมูล

d. แสดงข้อมูลตัวอย่าง : โค้ดวาดกราฟข้อมูลสองชุด (X0E และ X1D) ด้วย matplotlib.pyplotplt.figure(figsize=(16, 6)): ตั้งขนาดกราฟเป็น 16x6 นิ้วplt.plot(): สั่งวาดกราฟข้อมูล

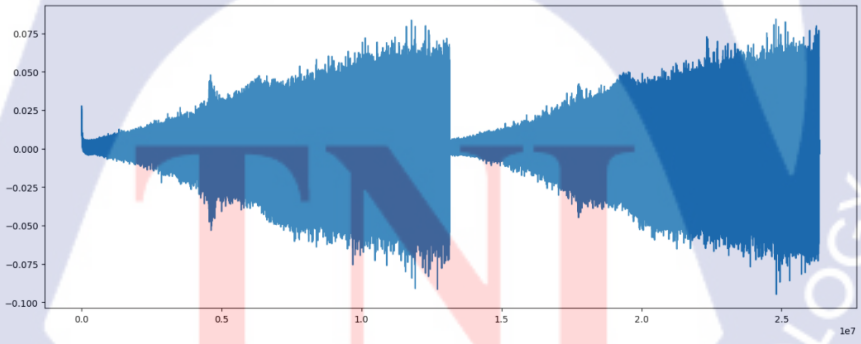
```
[1]: import pandas as pd
     from matplotlib import pyplot as plt
     import numpy as np
     import tensorflow as tf
     import zipfile
     from sklearn.preprocessing import RobustScaler

     url = './fraunhofer_eas_dataset_for_unbalance_detection_v1.zip'

     use_reference_models = False

     model_path = './models'

[2]: with zipfile.ZipFile(url, 'r') as f:
         with f.open('0D.csv', 'r') as c:
             data0D = pd.read_csv(c)
         with f.open('0E.csv', 'r') as c:
             data0E = pd.read_csv(c)
         with f.open('1D.csv', 'r') as c:
             data1D = pd.read_csv(c)
         with f.open('1E.csv', 'r') as c:
             data1E = pd.read_csv(c)

         with f.open('2E.csv', 'r') as c:
             data2E = pd.read_csv(c)

         with f.open('3D.csv', 'r') as c:
             data3D = pd.read_csv(c)
         with f.open('3E.csv', 'r') as c:
             data3E = pd.read_csv(c)

         with f.open('4E.csv', 'r') as c:
             data4E = pd.read_csv(c)

[3]: labels = {'no_unbalance':0, 'unbalance':1}
     sensor = 'Vibration_1'
     samples_per_second = 4096
     seconds_per_analysis = 1.0
     window = int(samples_per_second*seconds_per_analysis)

[4]: def get_features(data, label):
         n = int(np.floor(len(data)/window))
         data = data[:int(n)*window]
         X = data.reshape((n, window))
         y = np.ones(n)*labels[label]
         return X,y
```

รูปที่ 3.5 โค้ดการดึงและจัดการข้อมูล

```
[5]: X0D = np.array(data0D[sensor][50000:len(data0D[sensor])])
     X1D = np.array(data1D[sensor][50000:len(data1D[sensor])])
     X3D = np.array(data3D[sensor][50000:len(data3D[sensor])])

     X0E = np.array(data0E[sensor][50000:len(data0E[sensor])])
     X1E = np.array(data1E[sensor][50000:len(data1E[sensor])])
     X2E = np.array(data2E[sensor][50000:len(data2E[sensor])])
     X3E = np.array(data3E[sensor][50000:len(data3E[sensor])])
     X4E = np.array(data4E[sensor][50000:len(data4E[sensor])])

[6]: print(X0D.shape, X0E.shape)

     (26373295,) (6843567,)

[7]: plt.figure(figsize=(16, 6))
     plt.plot(X0E)  # Plotting the first column of X0
     plt.show()
```




```
[8]: plt.figure(figsize=(16, 6))
     plt.plot(X1D)  # Plotting the first column of X0
     plt.show()
```



รูปที่ 3.5 โค้ดการดึงและจัดการข้อมูล (ต่อ)

2) ส่วนการนำข้อมูลมาขยายสัญญาณ :

a. การ Import Library :

i. Tsaug : สำหรับการเพิ่มความหลากหลายข้อมูล Time Series

ii. tsaug.visualization.plot: สำหรับวาดกราฟข้อมูล

iii. TimeWarp, Crop, Quantize, Drift, Reverse, Pool, AddNoise : ฟังก์ชั่นสำหรับการแปลงข้อมูลแบบต่างๆ

b. สร้างฟังก์ชั่นเพิ่มความหลากหลาย (Augmenter) : แต่ละฟังก์ชั่นมีจุดประสงค์ดังนี้

i. Drift : สร้างการเบี่ยงเบนของสัญญาณแบบสุ่ม (random drift)

ii. TimeWarp : ปรับความเร็วของสัญญาณ (คล้ายการเล่นเร็วหรือช้า)

iii. Pool : ลดความละเอียดของข้อมูล (downsampling)

iv. AddNoise: เพิ่มสัญญาณรบกวน

c. สร้างและแสดงข้อมูลเพิ่มความหลากหลายจากข้อมูลที่มีอยู่:

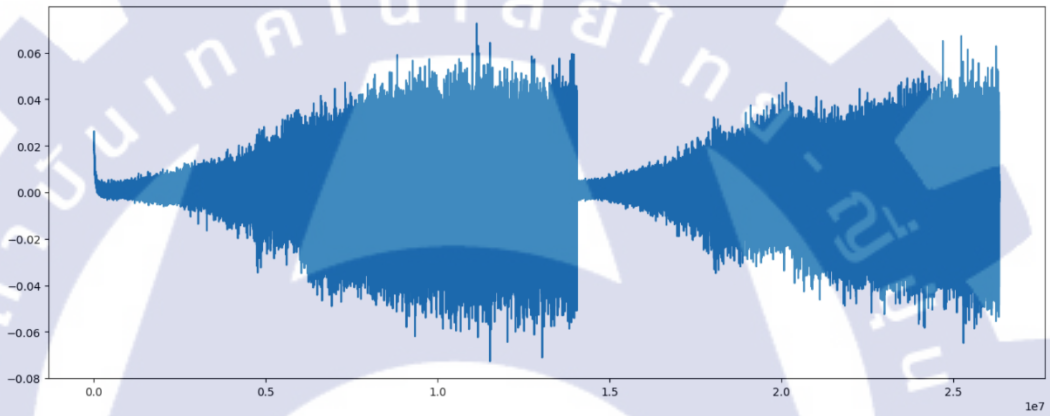
i. X2D = my_augmenter.augment(X1D) : สร้างชุดข้อมูลใหม่ X2D โดยการเพิ่มความหลากหลายให้กับข้อมูล X1D ด้วย my_augmenter

ii. X4D = my_augmenter.augment(X3D): สร้างชุดข้อมูลใหม่ X4D โดยการเพิ่มความหลากหลายให้กับข้อมูล X3D ด้วย my_augmenter

[9]:
```
import tsaug
from tsaug.visualization import plot
from tsaug import TimeWarp, Crop, Quantize, Drift, Reverse, Pool, AddNoise

# my_augmenter = (
#     Drift(max_drift=(0.1, 0.5)) @ 0.5  # with 80% probability, random drift the signal up to 10% - 50%
#     + TimeWarp(n_speed_change=5, max_speed_ratio=3)
#     + Pool(kind='ave', size=1)
#     + AddNoise(scale=0.01)
#     )

my_augmenter = (
    Drift(max_drift=(0.001, 0.05)) @ 0.05
    + TimeWarp(n_speed_change=5, max_speed_ratio=2)
    + Pool(kind='ave', size=1)
    + TimeWarp(n_speed_change=5, max_speed_ratio=2)
    + AddNoise(scale=0.002)
)
```

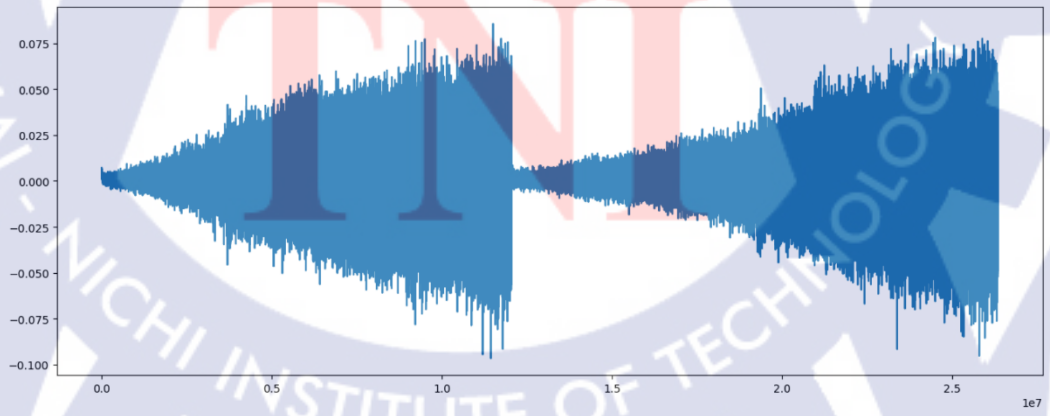
[10]:
```
X2D = my_augmenter.augment(X1D)
print(X2D.shape)
plt.figure(figsize=(16, 6))
plt.plot(X2D)  # Plotting the first column of X0
plt.show()
```

```
(26356911,)
```



[11]:
```
X4D = my_augmenter.augment(X3D)
print(X4D.shape)
plt.figure(figsize=(16, 6))
plt.plot(X4D)  # Plotting the first column of X0
plt.show()
```

```
(26337906,)
```



รูปที่ 3.6 โค้ดการสร้างชุดข้อมูลที่หลากหลาย

3) การดึงฟีเจอร์ แยกชุดข้อมูล และเตรียมข้อมูลสำหรับการฝึกโมเดล :

a. การดึงฟีเจอร์ : โค้ดนี้ใช้ฟังก์ชั่น get_features ดึงฟีเจอร์จากข้อมูล Time Series โดยฟังก์ชั่นนี้รับข้อมูล (X0D, X1D, etc.) และป้ายกำกับ ("no_unbalance" หรือ "unbalance") เป็นอินพุต ซึ่งฟังก์ชั่นดึงฟีเจอร์ที่เกี่ยวข้อง (อาจเป็นค่าทางสถิติหรือลักษณะเฉพาะของโดเมน) จากข้อมูล และส่งคืนฟีเจอร์ (X) และป้ายกำกับที่เกี่ยวข้อง (y)

b. การรวมฟีเจอร์และป้ายกำกับ : โค้ดทำการวนซ้ำผ่านแหล่งข้อมูลหลายแหล่ง (X0D, X1D, X2D, etc.) และดึงฟีเจอร์โดยใช้ get_features จากนั้นจะรวมฟีเจอร์ (X) และป้ายกำกับ (y) ที่ดึงมาจากทุกแหล่งเป็นอาร์เรย์เดียว

c. แยกชุดข้อมูลออกเป็นชุดฝึกและชุดตรวจสอบ:คล้ายกับขั้นตอนก่อนหน้า โค้ดจะดึงฟีเจอร์จากแหล่งข้อมูลการตรวจสอบ (X0E, X1E, etc.) จากนั้นรวมฟีเจอร์และป้ายกำกับจากทุกแหล่งตรวจสอบ และสุดท้าย ใช้ train_test_split จากsklearn.model_selection แยกชุดข้อมูลการฝึก (X, y) ออกเป็นชุดฝึกและชุดทดสอบ

i. train_test_ratio ตั้งค่าเป็น 0.9 หมายความว่า 90% ของข้อมูลจะไปที่ชุดฝึก และ 10% ไปที่ชุดทดสอบ

ii. random_state ตั้งค่าเป็น 0 เพื่อการ reproduce (ensuring the same split if the code is run multiple times)

d. พิมพ์ขนาดข้อมูล: โค้ดพิมพ์ขนาดของชุดข้อมูลฝึกและชุดข้อมูลทดสอบ (ฟีเจอร์และป้ายกำกับ) เพื่อยืนยันขนาด

```
[12]: X0,y0 = get_features(X0D, "no_unbalance")
      X1,y1 = get_features(X1D, "unbalance")
      X2,y2 = get_features(X2D, "unbalance")
      X3,y3 = get_features(X3D, "unbalance")
      X4,y4 = get_features(X4D, "unbalance")
      X=np.concatenate([X0, X1, X2, X3, X4])
      y=np.concatenate([y0, y1, y2, y3, y4])

      X0_val, y0_val = get_features(X0E, "no_unbalance")
      X1_val, y1_val = get_features(X1E, "unbalance")
      X2_val, y2_val = get_features(X2E, "unbalance")
      X3_val, y3_val = get_features(X3E, "unbalance")
      X4_val, y4_val = get_features(X4E, "unbalance")
      X_val=np.concatenate([X0_val, X1_val, X2_val, X3_val, X4_val])
      y_val=np.concatenate([y0_val, y1_val, y2_val, y3_val, y4_val])

      print(X.shape, y.shape, X_val.shape, y_val.shape)

      (32166, 4096) (32166,) (8359, 4096) (8359,)

[13]: from sklearn.model_selection import train_test_split
      train_test_ratio = 0.9
      X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 1-train_test_ratio, random_state = 0)

      print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)

      (28949, 4096) (28949,) (3217, 4096) (3217,)
```

รูปที่ 3.7 โค้ดการแยกชุดและเตรียมข้อมูล

4) การแปลงสัญญาณเวลาเป็นความถี่ (Time Series to Frequency Domain) : โค้ดนี้ทำการแปลงสัญญาณเวลา (Time Series) ไปเป็นสัญญาณความถี่ (Frequency Domain) เพื่อวิเคราะห์ข้อมูลในมุมมองความถี่ ซึ่งมีประโยชน์ในงานประมวลผลสัญญาณหลายประเภท

a. แปลงสัญญาณด้วย FFT: X_fft = np.abs(np.fft.rfft(X,axis=1))[:,:int(window/2)] บรรทัดนี้เป็นหัวใจสำคัญ

i. np.fft.rfft: ใช้สำหรับแปลงสัญญาณจากเวลาเป็นความถี่ โดยใช้ Fast Fourier Transform (FFT)

ii. axis=1: ระบุแกนที่ใช้สำหรับการแปลง (แกนที่ 1 ของข้อมูล) np.abs: เอาค่าสัมบูรณ์ (absolute value) ของผลลัพธ์จาก FFT (ส่วนประกอบแท้)

iii. [:,:int(window/2)]: เก็บเฉพาะครึ่งแรกของผลลัพธ์ FFT เนื่องจากสัญญาณจริงมีค่าจริง (real) ที่ความถี่ 0 และข้อมูลซ้ำกันอีกครึ่งหลังที่ความถี่สูงกว่า

b. การประยุกต์ใช้กับชุดข้อมูลต่างๆ : โค้ดทำการแปลงสัญญาณด้วย

FFT กับชุดข้อมูลทั้งหมด (X, X_train, X_test, X_val) แยกตามฟังก์ชั่น np.fft.rfftโดยเก็บเฉพาะครึ่งแรกของผลลัพธ์ ([:,:int(window/2)]) เหมือนเดิม

c. การยกเว้นความถี่ 0: X_fft[:,0]=0 บรรทัดนี้กำหนดค่าความถี่ 0 (DC component) ให้เป็น 0 เนื่องจากความถี่ 0 ไม่มีประโยชน์สำหรับการวิเคราะห์หาความถี่ของสัญญาณรบกวนหรือลักษณะของสัญญาณ

d. แสดงขนาดข้อมูล : พิมพ์ขนาดของข้อมูลหลังการแปลง (ฟีเจอร์ความถี่) เพื่อยืนยัน

```
[14]: X_fft = np.abs(np.fft.rfft(X, axis=1))[:,:int(window/2)]
      X_train_fft = np.abs(np.fft.rfft(X_train, axis=1))[:,:int(window/2)]
      X_test_fft = np.abs(np.fft.rfft(X_test, axis=1))[:,:int(window/2)]
      X_val_fft = np.abs(np.fft.rfft(X_val, axis=1))[:,:int(window/2)]

      X_fft[:,0]=0
      X_train_fft[:,0]=0
      X_test_fft[:,0]=0
      X_val_fft[:,0]=0

      print(X_fft.shape, X_train_fft.shape, X_test_fft.shape, X_val_fft.shape)

      (32166, 2048) (28949, 2048) (3217, 2048) (8359, 2048)
```

รูปที่ 3.8 โค้ดการสร้าง FFT

5) สร้างและประยุกต์ใช้ RobustScaler สำหรับปรับขนาดข้อมูล :โค้ดนี้สร้าง RobustScaler (scaler) “quantile_range=(5,95)” ที่เหมาะกับข้อมูลชุดฝึก (X_train_fft) จากนั้นนำไปปรับขนาดข้อมูลทั้งชุดฝึก (X_train_fft_sc), ชุดทดสอบ (X_test_fft_sc), ชุดตรวจสอบ (X_val_fft_sc) และชุดข้อมูลทั้งหมด (X_fft_sc)

```
[15]: scaler = RobustScaler(quantile_range=(5,95)).fit(X_train_fft)

X_fft_sc = scaler.transform(X_fft)
X_train_fft_sc = scaler.transform(X_train_fft)
X_test_fft_sc = scaler.transform(X_test_fft)
X_val_fft_sc = scaler.transform(X_val_fft)
```

รูปที่ 3.9 โค้ดการประยุกต์ใช้ RobustScaler

6) การสร้างและฝึกโมเดล FCN 5 เลเยอร์สำหรับตรวจจับความไม่สมดุล :โค้ดนี้สร้างโมเดล FCN 5 เลเยอร์เพื่อตรวจจับความไม่สมดุลจากข้อมูล สเปคตรัมความถี่ (frequency spectra) ที่ผ่านการปรับขนาดมาแล้ว

a. นำเข้าไลบรารีที่จำเป็น :
   i. tensorflow.keras : สำหรับการสร้างโมเดลและจัดการเลเยอร์
   ii. ModelCheckpoint : เพื่อบันทึกโมเดลที่ดีที่สุดระหว่างการฝึก
   iii. l1_l2: สำหรับ regularization
b. คำนวณค่าน้ำหนักแต่ละคลาส (ไม่จำเป็น) :
   i. weight_for_0, weight_for_1 : คำนวณน้ำหนักเพื่อปรับสมดุลระหว่างข้อมูลแต่ละคลาส
   ii. class_weight : ให้น้ำหนักแต่ละคลาสเพื่อแก้ปัญหาข้อมูลไม่สมดุล
c. กำหนดจำนวนชุดการฝึก : epochs = 100 เป็นค่าเริ่มต้น
d. สร้างโมเดล :
   i. X_in กำหนด Input layer ให้รับข้อมูลตามขนาดของข้อมูล
   ii. for j in range(5) สร้าง 5 hidden layers เป็น Dense layers ขนาด 1024 หน่วย
   iii. ใช้ LeakyReLU activation ต่อจากแต่ละ Dense layer ด้วย alpha = 0.05
   iv. X_out ให้ output เป็นค่าเดียวผ่าน sigmoid เป็น 0 หรือ 1 สำหรับทำนายคลาส

e. กำหนด checkpoint เพื่อบันทึกโมเดล :

i. best_model_filepath : กำหนดตำแหน่งและชื่อไฟล์สำหรับบันทึกโมเดล

ii. ModelCheckpoint : สร้าง callback ให้บันทึกโมเดลเมื่อ validation loss น้อยที่สุด

f. Compile model :

i. optimizer = Adam(learning_rate=0.0005): ใช้ optimizer เป็น Adam พร้อม learning rate เป็น 0.0005

ii. loss = 'binary_crossentropy': เมทริกประเมิน model สำหรับ binary classification

iii. metrics = ['accuracy']: ให้คำนวณ accuracy ระหว่างการฝึก

g. การแสดง model summary : แสดงข้อมูลโครงสร้างของ model

```
[16]: from tensorflow.keras.models import Sequential, load_model, Model
      from tensorflow.keras.layers import BatchNormalization,LeakyReLU,Dense,Dropout
      from tensorflow.keras.layers import Input,Conv1D,MaxPooling1D,Flatten,ReLU
      from tensorflow.keras.optimizers import Adam
      from tensorflow.keras.callbacks import ModelCheckpoint
      from tensorflow.keras.regularizers import l1_l2

      weight_for_0 = len(y)/(2*len(y[y==0]))
      weight_for_1 = len(y)/(2*len(y[y==1]))
      class_weight = {0: weight_for_0, 1: weight_for_1}

      epochs = 100

      X_in = Input(shape=(X_train_fft.shape[1],), name="cam_layer")
      x = X_in
      #the 4 hidden layers
      for j in range(5):
          x = Dense(units = 1024, activation="linear")(x)
          x = LeakyReLU(alpha=0.05)(x)
      X_out = Dense(units = 1, activation = 'sigmoid')(x)
      model_i = Model(X_in, X_out)


      best_model_filepath = f"{model_path}/fft_fcn_5_layers_aug.h5"
      checkpoint = ModelCheckpoint(best_model_filepath, monitor='val_loss',
                                   verbose=1, save_best_only=True, mode='min')

      model_i.compile(optimizer = Adam(learning_rate=0.0005), loss = 'binary_crossentrc
                      metrics = ['accuracy'])
      model_i.summary()
```

รูปที่ 3.10 การโค้ดสร้างเพื่อฝึกโมเดล FCN

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 cam_layer (InputLayer)      [(None, 2048)]            0

 dense (Dense)               (None, 1024)              2098176

 leaky_re_lu (LeakyReLU)     (None, 1024)              0

 dense_1 (Dense)             (None, 1024)              1049600

 leaky_re_lu_1 (LeakyReLU)   (None, 1024)              0

 dense_2 (Dense)             (None, 1024)              1049600

 leaky_re_lu_2 (LeakyReLU)   (None, 1024)              0

 dense_3 (Dense)             (None, 1024)              1049600

 leaky_re_lu_3 (LeakyReLU)   (None, 1024)              0

 dense_4 (Dense)             (None, 1024)              1049600

 leaky_re_lu_4 (LeakyReLU)   (None, 1024)              0

 dense_5 (Dense)             (None, 1)                 1025

=================================================================
Total params: 6297601 (24.02 MB)
Trainable params: 6297601 (24.02 MB)
Non-trainable params: 0 (0.00 Byte)
```

รูปที่ 3.10 การโค้ดสร้างเพื่อฝึกโมเดล FCN (ต่อ)

7) เป็นการนำข้อมูลสเปกตรัมความถี่ที่ปรับขนาดแล้วมาฝึกโมเดล FCN เพื่อให้โมเดลสามารถแยกแยะข้อมูลที่มีและไม่มีความไม่สมดุลได้ โดยมีการประเมินประสิทธิภาพระหว่างการฝึกด้วยข้อมูลตรวจสอบเพื่อป้องกันการเรียนรู้มากเกินไป (overfitting) :

a. epochs = 20 คือจำนวนรอบที่วนลูปข้อมูลทั้งหมดสำหรับการฝึก (20 รอบในกรณีนี้)

b. batch_size = 128 คือจำนวนตัวอย่างของข้อมูลที่ใช้ฝึกในแต่ละรอบ

c. validation_data=(X_test_fft_sc, y_test) ใช้ข้อมูลแยกส่วนหนึ่ง (X_test_fft_sc, y_test) ประเมินประสิทธิภาพโมเดลระหว่างการฝึก

d. callbacks=[checkpoint] บันทึกโมเดลที่มีประสิทธิภาพดีที่สุด (ค่า loss ต่ำสุด) จากการประเมินบนชุดข้อมูลตรวจสอบ

e. class_weight=class_weight ช่วยปรับสมดุลของข้อมูลที่มีจำนวนตัวอย่างไม่เท่ากันในแต่ละคลาส

```
[17]: model_i.fit(X_train_fft_sc, y_train, epochs = 20, batch_size = 128,
                  validation_data=(X_test_fft_sc, y_test), callbacks=[checkpoint],
                class_weight=class_weight)

Epoch 1/20
225/227 [============================>.] - ETA: 0s - loss: 0.1206 - accuracy: 0.9528
Epoch 1: val_loss improved from inf to 0.11999, saving model to ./models/fft_fcn_5_layers_aug.h5
227/227 [==============================] - 7s 27ms/step - loss: 0.1200 - accuracy: 0.9531 - val_loss: 0.1200 - val_accuracy: 0.9565
Epoch 2/20
  3/227 [..............................] - ETA: 6s - loss: 0.0804 - accuracy: 0.9531
/opt/anaconda3/envs/thesis01/lib/python3.9/site-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model
as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(
225/227 [============================>.] - ETA: 0s - loss: 0.0317 - accuracy: 0.9873
Epoch 2: val_loss improved from 0.11999 to 0.05575, saving model to ./models/fft_fcn_5_layers_aug.h5
227/227 [==============================] - 6s 27ms/step - loss: 0.0321 - accuracy: 0.9872 - val_loss: 0.0557 - val_accuracy: 0.9876

Epoch 19: val_loss did not improve from 0.00028
227/227 [==============================] - 6s 28ms/step - loss: 0.0020 - accuracy: 0.9996 - val_loss: 0.0029 - val_accuracy: 0.9994
Epoch 20/20
226/227 [============================>.] - ETA: 0s - loss: 3.0346e-04 - accuracy: 1.0000
Epoch 20: val_loss did not improve from 0.00028
227/227 [==============================] - 6s 27ms/step - loss: 3.0324e-04 - accuracy: 1.0000 - val_loss: 0.0013 - val_accuracy: 0.
9997
[17]: <keras.src.callbacks.History at 0x297d22730>
```

รูปที่ 3.11 โค้ดการฝึกของโมเดล

8) การประเมินโมเดลที่ฝึกแล้ว :

a. โหลดโมเดลที่ดีที่สุด :

i. from tensorflow.keras.models import load_model: นำเข้าฟังก์ชันสำหรับโหลดโมเดล

ii. best_model_filepath: เส้นทางและชื่อไฟล์ของโมเดลที่ดีที่สุด (บันทึกไว้จาก ModelCheckpoint ตอนฝึก)

iii. model_i = load_model(best_model_filepath): โหลดโมเดลที่บันทึกไว้

b. ประเมินประสิทธิภาพ :test_results = model_i(X_val_fft_sc, training=False) : ป้อนข้อมูลตรวจสอบ (X_val_fft_sc) เข้าโมเดลที่โหลดมาtraining=False บอกว่าไม่ใช่การฝึก แต่เป็นการประเมินผล

c. val_acc_ges = model_i.evaluate(X_val_fft_sc, y_val): ประเมินประสิทธิภาพโมเดลบนข้อมูลตรวจสอบ (X_val_fft_sc, y_val) ค่าที่ได้ (val_acc_ges) บ่งบอกถึงความแม่นยำ (accuracy) และ loss ของโมเดลบนข้อมูลตรวจสอบ

d. แปลงผลลัพธ์เป็น DataFrame : การประเมินเป็น Data Frame ชื่อ results_to_pandas และแปลงข้อมูลความจริง (y_val) ของข้อมูลตรวจสอบเป็น Data Frame ชื่อ test_to_pandas

```
[18]: from tensorflow.keras.models import load_model

best_model_filepath = f"{model_path}/fft_fcn_5_layers_aug.h5"
model_i = load_model(best_model_filepath)
#train_acc_ges = model_i.evaluate(X_train_fft_sc, y_train)
#test_acc_ges = model_i.evaluate(X_test_fft_sc, y_test)
test_results = model_i(X_val_fft_sc,training=False)
val_acc_ges = model_i.evaluate(X_val_fft_sc, y_val)
results_to_pandas = pd.DataFrame(test_results[0])
test_to_pandas = pd.DataFrame(y_val)

WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam` runs slowly on M1/M2 Macs, please use the legacy Keras o
ptimizer instead, located at `tf.keras.optimizers.legacy.Adam`.
262/262 [==============================] - 1s 3ms/step - loss: 0.2442 - accuracy: 0.9646
```

รูปที่ 3.12 โค้ดการประเมินโมเดล

9) การสร้าง Confusion Matrix และ Classification Report :

a. นำเข้าไลบรารี:

i. from sklearn.metrics import classification_report, confusion_matrix: สำหรับสร้างรายงานการประเมิน

ii. import seaborn as sns: สำหรับสร้าง Confusion Matrix ที่สวยงาม

b. สร้างการคาดการณ์:

i. Y_pred = model_i.predict(X_val_fft_sc): ใช้โมเดลทำนายบนข้อมูลตรวจสอบ

ii. Y_pred1 = np.rint(Y_pred).astype(int): ปัดเศษผลลัพธ์การคาดการณ์ให้เป็น 0 หรือ 1 (สำหรับ binary classification)

iii. y_val1 = y_val.astype(int): แปลงข้อมูลความจริงของข้อมูลตรวจสอบให้เป็น integers

c. สร้าง Confusion Matrix:

i. confusion_mtx = confusion_matrix(y_val1, Y_pred1): สร้าง Confusion Matrix เพื่อดูการกระจายของการทำนายเปรียบเทียบกับข้อมูลความจริงและใช้ seaborn สร้าง Heatmap เพื่อแสดง Confusion Matrix แบบสวยงาม

d. สร้าง Classification Report:

i. print(classification_report(y_val1, Y_pred1)): แสดงรายงานซึ่งมี precision, recall, f1-score และ accuracy ของแต่ละคลาส

```
[19]: from sklearn.metrics import classification_report
      from sklearn.metrics import confusion_matrix
      import seaborn as sns


      Y_pred=model_i.predict(X_val_fft_sc)
      Y_pred1 = np.rint(Y_pred).astype(int)
      y_val1 = y_val.astype(int)
      # Y_pred_classes=np.argmax(Y_pred, axis=1)
      # Y_true=np.argmax(y_val, axis=1)
      confusion_mtx=confusion_matrix(y_val1,Y_pred1)

      f,ax=plt.subplots(figsize=(8,6))
      sns.heatmap(confusion_mtx, annot=True, linewidths=0.01, cmap="Blues", linecolor="black", fmt=".1f", ax=ax)
      plt.xlabel("Predict Label")
      plt.ylabel("True Label")
      plt.title("Confusion Matrix")
      plt.show()

      # classification report:
      print(classification_report(y_val1,Y_pred1))

      262/262 [==============================] - 1s 3ms/step
```

รูปที่ 3.13 โค้ดการสร้าง Confusion Matrix และ Classification Report

บทที่ 4
ผลการดำเนินงาน

4.1 วิเคราะห์สัญญาณจากค่าการวัด

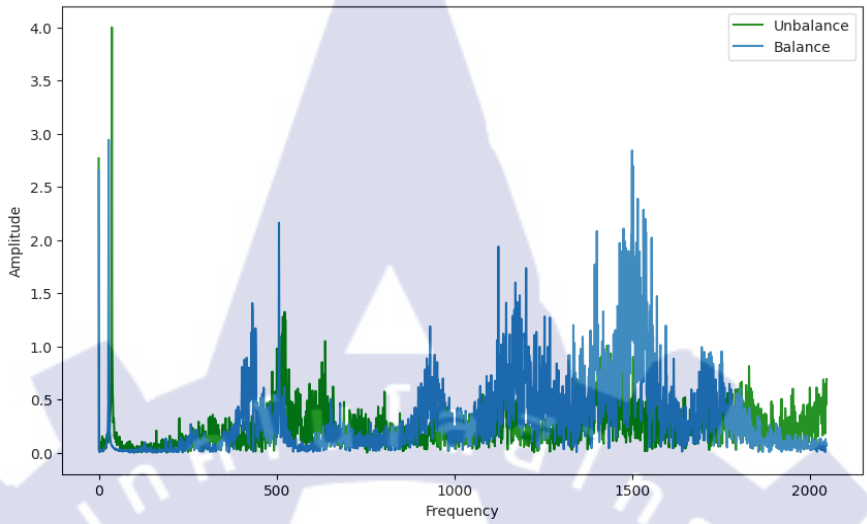
การทดลองที่เกิดขึ้นใช้วิธีการบันทึกและจัดเตรียมข้อมูลสัญญาณจากเซ็นเซอร์เพื่อวิเคราะห์ความไม่สมดุลของเพลาหมุน โดยข้อมูลสามารถนำไปใช้ประเมินประสิทธิภาพของอัลกอริทึมต่างๆ ด้วยวิธีการบันทึกข้อมูลสัญญาณจากเซ็นเซอร์วัดความสั่นสะเทือน 3 ตัว ภายใต้เงื่อนไขความไม่สมดุลของเพลาหมุน 4 ระดับ รวมถึงกรณีที่ไม่มีความไม่สมดุลจากนั้นทำการแบ่งข้อมูลเป็นสองชุดคือ ชุด ชุดพัฒนา (development) และชุดประเมิน (evaluation) ใช้วิธีการบันทึกข้อมูลแยกกัน จากนั้นจึงเพิ่มความแปรปรวนของข้อมูลด้วยการถอดประกอบและประกอบเพลาหมุนใหม่ระหว่างการบันทึกข้อมูล ซึ่งมีชุดข้อมูลอันประกอบไปด้วย 1) แรงดันไฟฟ้าขาเข้าของมอเตอร์ (Vin), 2) ความเร็วรอบการหมุนที่วัดได้ (Measured RPM), 3) สัญญาณจากเซ็นเซอร์วัดความสั่นสะเทือน 3 ตัว (Vibration 1, Vibration 2, Vibration 3)

รูปแบบของข้อมูลที่บันทึกจะเป็นไฟล์ CSV (Comma-Separated Values) ด้วยอัตราการสุ่มตัวอย่างที่ 4096 ค่าต่อวินาทีสำหรับทุกคอลัมน์ โดยการตั้งค่าวิเคราะห์ดังนี้

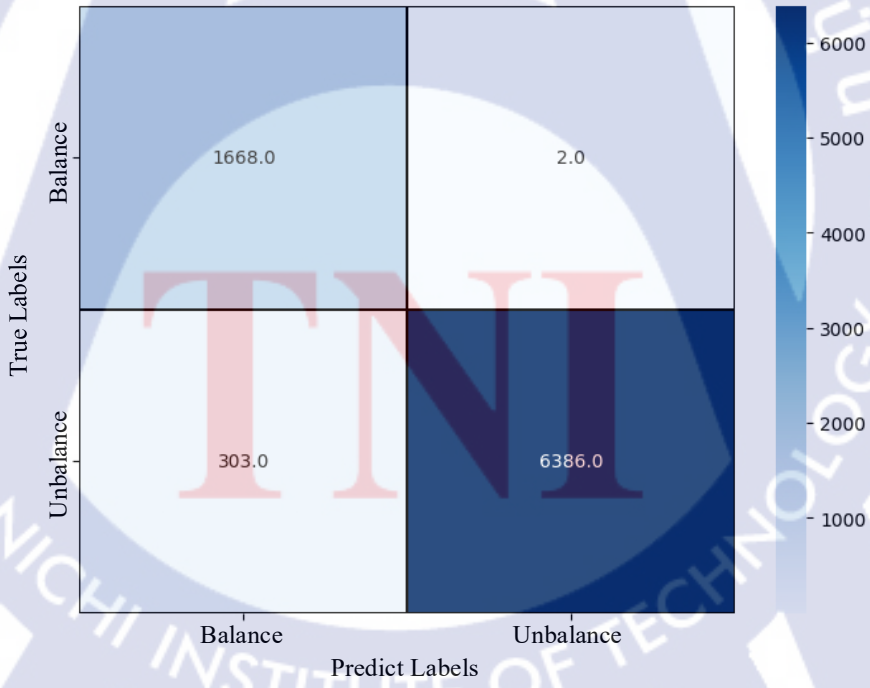
- แรงดันไฟฟ้ามอเตอร์ (Vin) :
  - ชุดพัฒนา : เพิ่มจาก 2.0 V ถึง 10.05 V (step ละ 0.05 V)
  - ชุดประเมิน : เพิ่มจาก 4.0 V ถึง 8.1 V (step ละ 0.1 V)
- ระยะเวลาคงที่: 20 วินาที
- ทำการทดสอบสองครั้งสำหรับแต่ละชุดข้อมูล

การคำนวณความเร็วรอบการหมุนได้จากสูตร $n\ (RPM) \approx 212 * Vin\ (V) + 209\ (2\ V \leq Vin \leq 10\ V)$ ซึ่งข้อมูลตัวอย่าง 50,000 ค่าแรก (ประมาณ 10 วินาที) ถูกตัดออกจากแต่ละชุดข้อมูลก่อนการแบ่งข้อมูล โดยจำนวนตัวอย่างขึ้นอยู่กับช่วงเวลาของข้อมูลที่ใช้งานได้

4.2 ผลการทดลอง



รูปที่ 4.1 สัญญาณจากการแปลงฟูเรียร์อย่างเร็ว



รูปที่ 4.2 แสดงประสิทธิภาพของโมเดลผ่าน confusion matrix

รูปที่ 4.1 แสดงการเปรียบเทียบสัญญาณที่ผ่านการแปลงฟูเรียร์อย่างเร็วจากเซ็นเซอร์การสั่นสะเทือนระหว่างมอเตอร์ที่มีและไม่มีมวลถ่วงน้ำหนัก จากรูปดังกล่าวจะเห็นได้ว่าสัญญาณที่ผ่านการแปลงฟูเรียร์อย่างเร็วทั้งสองสัญญาณมีความต่างกันอย่างชัดเจน ซึ่งทำให้โมเดลการเรียนรู้เชิงลึกสามารถแยกแยะรูปแบบข้อมูลที่แฝงอยู่ได้ชัดเจนมากขึ้น

งานวิจัยนี้ได้ใช้เทคนิคการขยายขนาดข้อมูลในเชิงเวลาในหัวข้อ 3.3.2 เพื่อสังเคราะห์ข้อมูลที่ใช้ในการฝึกสอนโมเดล D2 จากชุดข้อมูลเดิมคือ D1 และทำการสังเคราะห์ข้อมูล D4 จากข้อมูลเดิม D3 จากนั้นจึงนำข้อมูลทั้งหมด (D0 – D4) เข้าสู่กระบวนการฝึกสอนโมเดล และขั้นตอนการวัดประสิทธิภาพของโมเดลด้วยชุดข้อมูลทดสอบ (E0 -E4) ตามลำดับ รูปที่ 4.2 แสดงแผนภาพประสิทธิภาพของโมเดล หรือ Confusion Matrix ซึ่งแสดงให้เห็นถึงประสิทธิภาพของโมเดลในการจำแนกประเภทของข้อมูล จากแผนภาพ Confusion Matrix สามารถสรุปประสิทธิภาพของโมเดลได้ ตารางที่ 4.1 แสดงตัวชี้วัดประสิทธิภาพในการตรวจจับความผิดปกติในข้อมูล ประกอบด้วย

• ความแม่นยำ (Accuracy) บ่งชี้ว่าโดยรวมแล้ว โมเดลมีการคาดการณ์ถูกต้องบ่อยแค่ไหน ซึ่งมีค่าสูงถึง 96%

• ความเที่ยงตรง (Precision) บ่งชี้ว่าเมื่อโมเดลทำนายผลบวก การคาดการณ์นั้นถูกต้องบ่อยแค่ไหน ซึ่งผลการทดลองได้ค่า 97% ของตัวอย่างที่โมเดลคาดการณ์ว่าเป็นผลบวกนั้นถูกต้องจริง หมายความว่า โมเดลมีโอกาสน้อยที่จะคาดการณ์ผลบวกผิดพลาด

• ความไว (Recall) บ่งชี้ว่าโมเดลสามารถระบุผลบวกที่แท้จริงได้ดีเพียงใด ซึ่งจากการทดสอบได้ค่าสูงถึง 97%

• F1-Score บ่งชี้ถึงความสมดุลระหว่างความเที่ยงตรงและความไว ซึ่งจากการทดสอบได้ค่า 96% แสดงให้เห็นว่าโมเดลมีสมดุลที่ดีระหว่างความเที่ยงตรงและความไว

ตารางที่ 4.1 สรุปตัวชี้วัดประสิทธิภาพของโมเดล

Accuracy	Precision	Recall	F1-Score
96%	97%	97%	96%

จากตัวชี้วัดประสิทธิภาพทั้งหมด เห็นได้ว่าโมเดลมีประสิทธิภาพโดยรวมที่ค่อนข้างสูง โดยเฉพาะอย่างยิ่งในการระบุตัวอย่างที่เป็นผลบวก (การคาดการณ์ที่ถูกต้องเมื่อโมเดลจำแนกตัวอย่างเป็นค่าจริงอย่างถูกต้อง) ซึ่งผลการทดลองนี้แสดงให้เห็นว่าสามารถลดจำนวนข้อมูลที่ใช้ฝึกสอนโมเดลจากเทคนิคการขยายขนาดข้อมูลในเชิงเวลาได้ถึง 40% (2.3GB จาก 3.8GB) โดยไม่ทำให้ประสิทธิภาพของโมเดลลดลง

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

งานวิจัยนี้นำเสนอแนวทางใหม่สำหรับการตรวจจับความผิดปกติในอุตสาหกรรมโดยใช้โมเดลการเรียนรู้เชิงลึก (Deep Learning) ร่วมกับเทคนิคการขยายชุดข้อมูลเชิงเวลา (Time Series Data Augmentation) เทคนิคนี้ช่วยเพิ่มขนาดและความหลากหลายของชุดข้อมูลฝึกอบรมโดยการสังเคราะห์ตัวอย่างข้อมูลใหม่จากข้อมูลที่มีอยู่ โดยใช้เทคนิคการแปลงข้อมูล

5.1.1 การเบี่ยงเบนทางเวลา (Time Shift): เปลี่ยนแปลงตำแหน่งของจุดข้อมูลในแกนเวลา

5.1.2 การบิดเบี้ยวทางเวลา (Time Warp): ยืดหรือหดช่วงเวลาของข้อมูล

5.1.3 การสร้างสัญญาณรบกวน (Noise Injection): เพิ่มสัญญาณรบกวนแบบสุ่มไปยังข้อมูล

5.1.4 การลดความละเอียดของเวลา (Time Resolution Reduction): ลดจำนวนจุดข้อมูลในช่วงเวลา

งานวิจัยนี้ใช้โมเดลโครงข่ายประสาทเทียมแบบ Fully Connected ในการตรวจจับสัญญาณความผิดปกติ โดยใช้ข้อมูลที่สังเคราะห์ขึ้นบางส่วนร่วมกับข้อมูลจริง ผลการทดสอบแสดงให้เห็นว่าโมเดลมีประสิทธิภาพสูงในการตรวจจับสัญญาณผิดปกติในข้อมูลเชิงเวลา โดยสามารถลดจำนวนข้อมูลจริงที่ต้องวัดผลจากเครื่องจักรได้ถึง 40% งานวิจัยนี้แสดงให้เห็นถึงศักยภาพของโมเดลการเรียนรู้เชิงลึกและเทคนิคการขยายชุดข้อมูลเชิงเวลาในการตรวจจับความผิดปกติของเครื่องจักรในภาคอุตสาหกรรม เทคนิคนี้สามารถช่วยเพิ่มประสิทธิภาพการตรวจจับความผิดปกติ ลดต้นทุนการตรวจสอบ และใช้งานง่าย เหมาะสำหรับใช้งานในสภาพแวดล้อมที่มีข้อมูลจำกัด

5.2 แนวทางการวิจัยในอนาคต

5.2.1 พัฒนาโมเดลที่สามารถทำงานกับข้อมูลเชิงเวลาประเภทอื่น ๆ

5.2.2 ศึกษาเทคนิคการขยายชุดข้อมูลเชิงเวลาเพิ่มเติม

5.2.3 ทดสอบโมเดลกับข้อมูลจริงในสภาพแวดล้อมการใช้งานจริง

5.3 ปัญหาและอุปสรรค

5.3.1 งานวิจัยนี้อาศัยข้อมูลจากการวัดของงานวิจัยอื่นนำมาศึกษาซึ่งอาจมาความคลาดเคลื่อนจากการวัดซึ่งผู้วิจัยไม่อาจชี้ชัดได้

5.3.2 ในขั้นตอนของการปรับแต่งโมเดลให้มีความเหมาะสมและสอดคล้องกับความต้องการเป็นไปได้อยากรวมทั้งผลลัพท์ที่ได้ก็เป็นเพียงการนำผลการวัดจากงานวิจัยอื่นมาทดสอบเท่านั้นไม่สามารถหา data จากการวัดจริงหรืองานวิจัยอื่นๆมาทวบสอบเพื่อความชัดเจนได้

5.4 ข้อเสนอแนะ

5.4.1 สามารถตรวจสอบความถูกต้องได้จากการทดลองวัดค่าจริงเพื่อเปรียบเทียบอีกครั้ง

5.4.2 เพิ่มเติมการคาดการณ์เป็นตัวเลขและทดสอบด้วยการตรวจวัด condition จริงของมอเตอร์อีกครั้ง

บรรณานุกรม

บรรณานุกรม

[1] Y. Jiang et al., “A machine vision-based realtime anomaly detection method for industrial products using deep learning,” *2019 Chinese Automation Congress (CAC)*, Hangzhou, China, November 22-24, 2019, pp. 4842-4847.

[2] M. H. M. Ghazali and W. Rahiman, “Vibration analysis for machine monitoring and diagnosis: A systematic review,” *Shock and Vibration*, vol. 2021, pp. 1-25, August 2021.

[3] P. Po and H. Kim, “Smart factory mold injection anomaly detection using deep autoencoder neural network,” *IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, Busan, Korea, October 22-23, 2023, pp. 1-3.

[4] G. K. Durbhaka and B. Selvaraj, “Predictive maintenance for wind turbine diagnostics using vibration signal analysis based on collaborative recommendation approach,” *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Jaipur, India, September 21-24, 2016, pp. 1839-1842.

[5] X. Xu et al., “LSTM-GAN-XGBOOST Based anomaly detection algorithm for time series data,” *11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan)*, Jinan, China, October 22-25, 2020, pp. 334-339.

[6] O. Mey et al., “Machine learning-based unbalance detection of a rotating shaft using vibration data,” *25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),* Vienna, Austria, September 8-11, 2020, pp. 1610-1617.

[7] O. Mey et al., “*Vibration Measurements on a Rotating Shaft At Different Unbalance Strengths,*”[Online]. Available: https://fordatis.fraunhofer.de/handle /fordatis/151.3. [Accessed: March 5, 2023].

[8] O. Mey et al., “Explainable ai algorithms for vibration data-based fault detection: Use case-adadpted methods and critical evaluation,” *Sensors*, vol.22, no.23, pp. 9037, November 2022.

ภาคผนวก

ภาคผนวก ก.

Code Program

```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
import tensorflow as tf
import zipfile
from sklearn.preprocessing import RobustScaler

url = './fraunhofer_eas_dataset_for_unbalance_detection_v1.zip'

use_reference_models = False

model_path = './models'

labels = {'no_unbalance':0, 'unbalance':1}
sensor = 'Vibration_1'
samples_per_second = 4096
seconds_per_analysis = 1.0
window = int(samples_per_second*seconds_per_analysis)
X0D = np.array(data0D[sensor][50000:len(data0D[sensor])])
X1D = np.array(data1D[sensor][50000:len(data1D[sensor])])
X3D = np.array(data3D[sensor][50000:len(data3D[sensor])])

X0E = np.array(data0E[sensor][50000:len(data0E[sensor])])
X1E = np.array(data1E[sensor][50000:len(data1E[sensor])])
X2E = np.array(data2E[sensor][50000:len(data2E[sensor])])
X3E = np.array(data3E[sensor][50000:len(data3E[sensor])])
X4E = np.array(data4E[sensor][50000:len(data4E[sensor])])

print(X0D.shape, X0E.shape)
plt.figure(figsize=(16, 6))
```

```
plt.plot(X0E)  # Plotting the first column of X0
plt.show()
plt.figure(figsize=(16, 6))
plt.plot(X1D)  # Plotting the first column of X0
plt.show()
import tsaug
from tsaug.visualization import plot
from tsaug import TimeWarp, Crop, Quantize, Drift, Reverse, Pool, AddNoise

# my_augmenter = (
#     Drift(max_drift=(0.1, 0.5)) @ 0.5  # with 80% probability, random drift the signal
up to 10% - 50%
#     + TimeWarp(n_speed_change=5, max_speed_ratio=3)
#     + Pool(kind='ave', size=1)
#     + AddNoise(scale=0.01)
# )

my_augmenter = (
    Drift(max_drift=(0.001, 0.05)) @ 0.05
    + TimeWarp(n_speed_change=5, max_speed_ratio=2)
    + Pool(kind='ave', size=1)
    + TimeWarp(n_speed_change=5, max_speed_ratio=2)
    + AddNoise(scale=
X2D = my_augmenter.augment(X1D)
print(X2D.shape)
plt.figure(figsize=(16, 6))
plt.plot(X2D)  # Plotting the first column of X0
plt.show()
X4D = my_augmenter.augment(X3D)
print(X4D.shape)
plt.figure(figsize=(16, 6))
```

```
plt.plot(X4D)  # Plotting the first column of X0
plt.show()
X0,y0 = get_features(X0D, "no_unbalance")
X1,y1 = get_features(X1D, "unbalance")
X2,y2 = get_features(X2D, "unbalance")
X3,y3 = get_features(X3D, "unbalance")
X4,y4 = get_features(X4D, "unbalance")
X=np.concatenate([X0, X1, X2, X3, X4])
y=np.concatenate([y0, y1, y2, y3, y4])

X0_val, y0_val = get_features(X0E, "no_unbalance")
X1_val, y1_val = get_features(X1E, "unbalance")
X2_val, y2_val = get_features(X2E, "unbalance")
X3_val, y3_val = get_features(X3E, "unbalance")
X4_val, y4_val = get_features(X4E, "unbalance")
X_val=np.concatenate([X0_val, X1_val, X2_val, X3_val, X4_val])
y_val=np.concatenate([y0_val, y1_val, y2_val, y3_val, y4_val])

print(X.shape, y.shape, X_val.shape, y_val.shape)
from sklearn.model_selection import train_test_split
train_test_ratio = 0.9
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 1-train_test_ratio,
random_state = 0)

print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
X_fft = np.abs(np.fft.rfft(X, axis=1))[:,:int(window/2)]
X_train_fft = np.abs(np.fft.rfft(X_train, axis=1))[:,:int(window/2)]
X_test_fft = np.abs(np.fft.rfft(X_test, axis=1))[:,:int(window/2)]
X_val_fft = np.abs(np.fft.rfft(X_val, axis=1))[:,:int(window/2)]

X_fft[:,0]=0
```

```
X_train_fft[:,0]=0
X_test_fft[:,0]=0
X_val_fft[:,0]=0

print(X_fft.shape, X_train_fft.shape, X_test_fft.shape, X_val_fft.shape)
scaler = RobustScaler(quantile_range=(5,95)).fit(X_train_fft)

X_fft_sc = scaler.transform(X_fft)
X_train_fft_sc = scaler.transform(X_train_fft)
X_test_fft_sc = scaler.transform(X_test_fft)
X_val_fft_sc = scaler.transform(X_val_fft)
from tensorflow.keras.models import Sequential, load_model, Model
from tensorflow.keras.layers import BatchNormalization,LeakyReLU,Dense,Dropout
from tensorflow.keras.layers import Input,Conv1D,MaxPooling1D,Flatten,ReLU
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.regularizers import l1_l2

weight_for_0 = len(y)/(2*len(y[y==0]))
weight_for_1 = len(y)/(2*len(y[y==1]))
class_weight = {0: weight_for_0, 1: weight_for_1}

epochs = 100

X_in = Input(shape=(X_train_fft.shape[1],), name="cam_layer")
x = X_in
#the 4 hidden layers
for j in range(5):
    x = Dense(units = 1024, activation="linear")(x)
    x = LeakyReLU(alpha=0.05)(x)
X_out = Dense(units = 1, activation = 'sigmoid')(x)
```

```
model_i = Model(X_in, X_out)

best_model_filepath = f"{model_path}/fft_fcn_5_layers_aug.h5"
checkpoint = ModelCheckpoint(best_model_filepath, monitor='val_loss',
                             verbose=1, save_best_only=True, mode='min')

model_i.compile(optimizer = Adam(learning_rate=0.0005), loss =
'binary_crossentropy',
                metrics = ['accuracy'])
model_i.summary()

model_i.fit(X_train_fft_sc, y_train, epochs = 20, batch_size = 128,
              validation_data=(X_test_fft_sc, y_test), callbacks=[checkpoint],
            class_weight=class_weight)
from tensorflow.keras.models import load_model

best_model_filepath = f"{model_path}/fft_fcn_5_layers_aug.h5"
model_i = load_model(best_model_filepath)
#train_acc_ges = model_i.evaluate(X_train_fft_sc, y_train)
#test_acc_ges = model_i.evaluate(X_test_fft_sc, y_test)
test_results = model_i(X_val_fft_sc,training=False)
val_acc_ges = model_i.evaluate(X_val_fft_sc, y_val)
results_to_pandas = pd.DataFrame(test_results[0])
test_to_pandas = pd.DataFrame(y_val)
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import seaborn as sns

Y_pred=model_i.predict(X_val_fft_sc)
Y_pred1 = np.rint(Y_pred).astype(int)
y_val1 = y_val.astype(int)
```

```
# Y_pred_classes=np.argmax(Y_pred, axis=1)
# Y_true=np.argmax(y_val, axis=1)
confusion_mtx=confusion_matrix(y_val1,Y_pred1)

f,ax=plt.subplots(figsize=(8,6))
sns.heatmap(confusion_mtx, annot=True, linewidths=0.01, cmap="Blues",
linecolor="black", fmt=".1f", ax=ax)
plt.xlabel("Predict Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

# classification report:
print(classification_report(y_val1,Y_pred1))
```