Data Preparation / Data Preprocessing with Artificial Intelligence

Applications and Limitations

presented by

Kaicheng Zhao

December 2024

This thesis was submitted to the

Chair for Intelligence in Quality Sensing

Laboratory for Machine Tools and Production Engineering (WZL)

Presented by:	Kaicheng Zhao
	kaicheng.zhao@rwth-aachen.de
	Matriculation: ij364637
First Examiner:	Prof. Dr.-Ing. Robert H. Schmitt
Research Associate:	Mario Moser, M.Sc.
	mario.moser@wzl-iqs.rwth-aachen.de

Abstract

During the rapid developing of artificial intelligence technology and the increasing complexity of raw datasets, data preprocessing and preparation have become increasingly important. To train high-quality models, there are often various targeted approaches to address the issues exposed in raw datasets. This research project will conduct an extensive study on four common data preprocessing topics: Data Cleaning, Feature Engineering, Data Augmentation, and Data Standardization. Based on artificial intelligence and machine learning methods, we applied various algorithms including regression models for missing data, RANSAC to detect outliers, feature mapping based on LDA and feature clustering by PCA and K-means, GAN, RL and GPT based data augmentation for images, audios and text data separately, and multimodal data based on pretrained model ViT and Bert. The key aspect of the research lies in providing data preprocessing methods based on artificial intelligence or machine learning and evaluating the specific methods by experiment results and related scientific references. The final results illustrate that artificial intelligence and machine learning methods can positively influence the process of data preprocessing. All of the experiments' codes have been save at following url address https://git.rwth-aachen.de/studenttheses/pt-aidp.git and under branch `1556286639-main-patch-84276`.

Contents

List of Abbreviations **iv**

List of Symbols **vi**

List of Figures **vii**

List of Tables **ix**

1 Introduction **1**

2 State of the Art **3**

2.1 Artificial Intelligence 3

2.2 Data Preprocessing 4

2.2.1 Data Clearing 5

2.2.2 Feature Engineering 6

2.2.3 Data Augmentation 6

2.2.4 Data Standardization 7

3 Data Clearing **9**

3.1 Missing Data 9

3.1.1 Categories of Missing Data and Multiple Imputation 9

3.1.2 Regressor Models for MCAR, MAR and MNAR problems 11

3.1.3 Advantage and Limitation of three Kinds of Regression Model 15

3.2 Outliers 15

3.2.1 Description and Causes of Outliers 15

3.2.2 RANSAC for Outliers 16

3.2.3 Advantage and Limitation of RANSAC 18

4 Feature Engineering **19**

4.1 Feature Mapping 19

4.1.1 Linear Discriminant Analysis and Demonstration Experiment 19

4.1.2 Evaluation on Linear Discriminant Analysis 21

4.2 Feature Clustering 22

4.2.1 Mathematical Steps in PCA 22

4.2.2 Mathematical Steps in K-means Algorithm 23

4.2.3 Feature Clustering Demonstration based on PCA and K-means 24

4.2.4 Evaluation on Feature Clustering based on PCA and K-means 25

5 Data Augmentation **27**
5.1 Image Data Augmentation . . . 27
5.1.1 Common Image Manipulation . . . 27
5.1.2 Generative Model to expand dataset . . . 28
5.1.3 Evaluation on Advantage and Limitation between Common Method and GANs Method . . . 31
5.2 Audio Data Augmentation . . . 32
5.2.1 Common non-AI Audio Augmentation . . . 33
5.2.2 RL based Audio Augmentation . . . 35
5.2.3 Evaluation on Advantage and Limitation between Common Method and RL Method . . . 41
5.3 Text Data Augmentation . . . 41
5.3.1 Text Augmentation based on Generative Pre-trained Transformer . . . 42
5.3.2 Evaluation on text data augmentation based on Generative Pre-trained Transformer . . . 44

6 Data Standardization **45**
6.1 Single Modal Data Standardization . . . 45
6.2 Multimodal Data Standardization . . . 46
6.3 Evaluation on AI/ML Influence on Data Standardization . . . 47

7 Experiments Results **48**

8 Conclusion, Limitation and Outlook **52**
8.1 Conclusion . . . 52
8.2 Limitation . . . 52
8.3 Outlook . . . 53

Bibliography **54**

List of Abbreviations

AI Artificial Intelligence

API Application Programming Interface

BCELoss Binary Cross-Entropy Loss

BERT Bidirectional Encoder Representations from Transformers

CLIP Contrastive Language-Image Pre-training

DNN Dense Neural Network

GANs Generative Adversarial Networks

GLIP Grounded Language-Image Pre-training

GPT Generative Pre-trained Transformer

ISTFT Inverse Short-Time Fourier Transform

LDA Linear Discriminant Analisys

LDM Linear Discriminant Model

MAR Missing at Random

MCAR Missing Completely at Random

ML Machine Learning

MLP Multilayer Perceptron

MNAR Missing Not at Random

MNIST Modified National Institute of Standards and Technology

MSE Mean Square Error

PCA Principal Component Analysis

RANSAC Random Sample Consensus

ReLU Rectified Linear Unit

RL Reinforcement Learning

STFT Short-Time Fourier Transform

SVM Support Vector Machine

ViT Vision Transformer

List of Figures

1.1 Time Distribution from 2020 Anaconda Report [3] 1
1.2 Time Distribution from 2022 Anaconda Report [4] 1

2.1 Types of Learning [14] 3
2.2 Knowledge Discovery in Databases Process [15] 4
2.3 Three Kinds of Problems of Real World Dataset [19] 4
2.4 CLIP Model Architecture [56] 8

3.1 Missing Data Illustration 10
3.2 Multilayer Perceptron Architecture 13
3.3 Prediction of MCAR missing values with Ridge, Lasso and MLP 13
3.4 Prediction of MAR missing values with Ridge, Lasso and MLP 13
3.5 Prediction of MNAR missing values with Ridge, Lasso and MLP 14
3.6 Comparative Experiment MSE 14
3.7 RANSAC Raw Dataset 17
3.8 RANSAC Iteration 17

4.1 LDA Feature Mapping 21
4.2 Wine Dataset 24
4.3 Two-dimensional Dataset under PCA 25
4.4 Three-dimensional Dataset under PCA 25
4.5 Two-dimensional Dataset K-Means Clustering 25
4.6 Three-dimensional Dataset K-Means Clustering 25

5.1 Image Manipulation 28
5.2 GAN Block Diagram 29
5.3 Raw Noisy Data 30
5.4 The First Three Iteration Results 31
5.5 Written Digits Update Results 31
5.6 Mel Spectrum Demonstration 34
5.7 Mel Spectrum Demonstration by Data Augmentation 35
5.8 DNN Architecture 36
5.9 DNN Scores Line Chart 37
5.10 RL Architecture 38
5.11 Data Augmentation Probability Evolution 40
5.12 Reward Evolution 40

6.1 Image-Text Matches Demonstration 47

List of Tables

3.1 Categories of Missing Data 9
3.2 Specific Examples of Missing Data 10
3.3 Inliers During RANSAC 18

5.1 Performance metrics on the validation set 37
5.2 Parameter Candidates for RL 39
5.3 Experimental BERTScore F1 Scores for Annealing, Quenching, and Tempering . . . 44

7.1 Overview of Tasks, Topics, and Experiment Methods 48
7.2 Overview of Topics, Methods, and Datasets 49
7.3 Topics, Experiment Methods, and Parametrisation References 50

1 Introduction

In the technical field of modern machine learning (ML) and deep learning, datasets are playing a core role of model training and evaluation. ML and deep learning models are trained by using datasets. The dataset provides models input features and ideal outputs, and the models can learn features and rules by analyzing this dataset.

Under different situations, the application of ML and deep learning requires specific types of datasets. The common tasks which have strong requirement of datasets include computer vision, nature language processing, medical imaging, autonomous driving and so on. A wide range of tasks need to collect and preproceed raw dataset from real life.

However, the datasets which are collected from the real world are susceptible to many external factors such as experiment equipment errors and manual recording faults. A low-quality dataset has more chance to produce a poorer model during training process. Moreover, low-quality datasets can also negatively impact the economic performance. According to a report by Gartner, low-quality data causes an annual economic loss of approximately $3.1 trillion for companies [1]. This figure includes various costs, such as decreased productivity, wasted resources, and missed market opportunities due to data errors. Similarly, a 2017 report by Harvard Business Review revealed that only 3% of company data meets basic quality standards of datasets, and a single root cause could potentially lead to thousands of future errors and millions in losses [2]. Therefore, the process of data preprocession is necessary for both scientific and economic requirements.

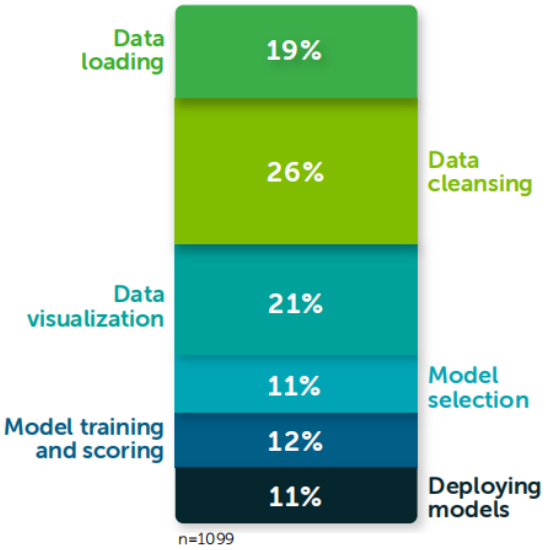

Figure 1.1: Time Distribution from 2020 Anaconda Report [3]

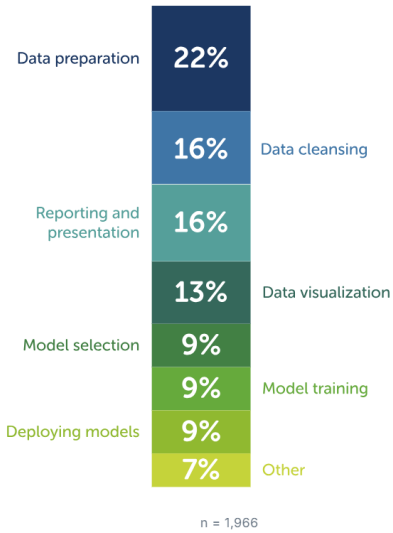

Figure 1.2: Time Distribution from 2022 Anaconda Report [4]

In the recent years, data preprocessing has been a more and more challenging task for ML and big data engineers. In Anaconda's 2020 State of Data Science report [3], a survey of over 2,300 data professionals from more than 100 countries revealed that data cleaning and data loading took up 26% and 19% of their time, respectively. This means that nearly 50% of their time was spent on data preprocessing tasks which significantly consuming the working hours of engineers. The time distribution is shown in Figure 1.1. Similarly, the 2022 Anaconda report [4], which surveyed 1,966 data science engineers worldwide. The technical record illustrated that data preparation and data cleansing continued to be time-consuming tasks, accounting for 22% and 16% of their time and shown in Figure 1.2. In the 2023 report [5], a focus on the security of open-source packages showed that over 40% of surveyed companies relied on manual checks against a vulnerability database to ensure safety. Further, 80% of engineers and their teams reported spending over 25% of their time on manual security checks, with more than 20% of respondents dedicating more than 50% of their time to such tasks. These findings highlight that data preprocessing continues to occupy a conspicuous ratio of engineers' time. There remains substantial potential for leveraging artificial intelligence (AI) and ML methods to replace labor-intensive manual processes and improve efficiency.

This research project aims to investigate various common data preprocessing techniques and explore the potential of ML/AI technologies to enhance these processes. The study will evaluate the strengths and limitations of these ML/AI approaches through practical coding implementations and a review of relevant scientific literature. By coding demonstration and relevant references, the influence of AI/ML during data preprocessing will be illustrated.

In the following sections, the State of The Art in data preprocessing will first be introduced. Subsequently, four selected topics in data preprocessing, Data Cleaning, Feature Engineering, Data Augmentation, and Data Standardization, will be addressed. Finally, the results of the research will be analyzed.

2 State of the Art

In this chapter, a State of The Art discussion and description of data preprocessing and AI will be conducted. Firstly, the definition of data preprocessing and AI will be illustrated. Then four data preprocessing topics, data cleaning, feature engineering, data augmentation, and data standardization which are involved with AI methods in previous researches will be introduced.

2.1 Artificial Intelligence

The definition of AI is the science and engineering of making intelligent machines, especially intelligent computer programs [6]. From the core technologies of AI, multiple subfields can be derived, each applied to different domains. These include ML and deep learning [7], which train models using data to simulate human brain thinking; Natural Language Processing [8], which enables machines to understand human language and analyze the logic behind it; Computer Vision, which focuses on enabling machines to acquire and interpret visual information [9]; and robotics, which allows AI to perform automated tasks [10].

The project focuses on exploring the positive impact of AI technologies in ML on data preprocessing [11]. As a critical component of AI, ML emphasizes enabling machines to mimic the process of human learning [12]. Through model parameter adjustments and iterative execution of algorithms, machines trained on large datasets will be capable of accepting inputs and predicting outputs [13]. ML methods can be broadly categorized into supervised learning, unsupervised learning, reinforcement learning (RL), neural networks and so on, which have been shown in Figure 2.1.

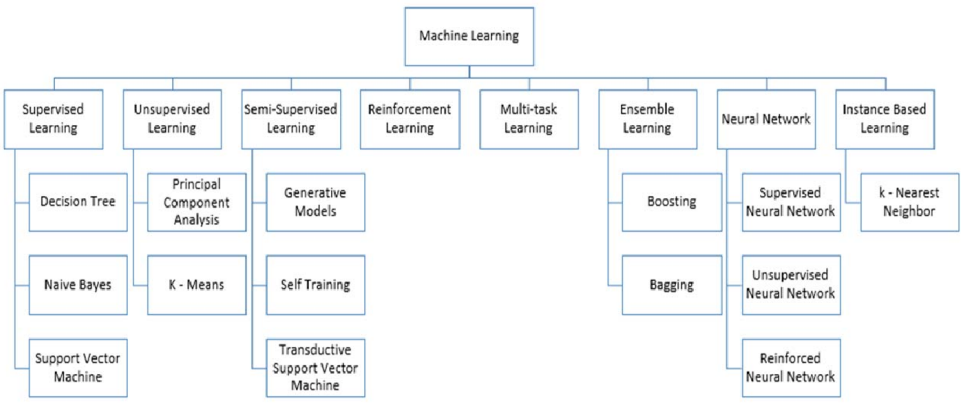

Figure 2.1: `Types of Learning` **[14]**

2.2 Data Preprocessing

The definition of data preprocessing is the set of techniques which are used before the application of data mining method, as shown in Figure 2.2. [15].

As one of the most significant steps of knowledge discovery in databases, data preprocessing includes preparation and transformation of data into a suitable status for further mining procedure [16]. The target of data preprocession is to help improve the quality of the data with high efficiency [17]. Besides that, data preprocession can also accelerate the speed of computation [18]. With less content to be considered during training period, the computation pressure is reduced.

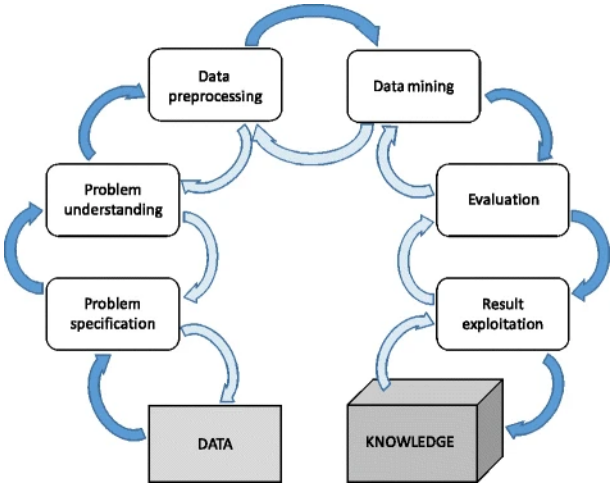

Figure 2.2: `Knowledge Discovery in Databases Process` **[15]**

The dataset which is collected from real world has to face with problems of incomplete, noisy and inconsistent data. The problems can be divided into three groups, too much data, too little data and fractured data which has been shown in Figure 2.3. [19]

Problems with data collected from real-world		
Too much data	**Too little data**	**Fractured data**
- Feature extraction - Irrelevant data - Substantial data sizes - Categorical/Numerical mixed data - Noisy data	- Missing attributes - Missing values - Less amount of data	- Incompatible data - Data collected from multiple sources - Data from various levels of granularity

Figure 2.3: `Three Kinds of Problems of Real World Dataset` **[19]**

For missing data, noisy data and irrelevant data problems, data clearing is one kind of potential solution [20]. When it comes to feature extraction, feature engineering is commonly performed to extract the main features of the raw data set [21]. When the amount of data is severely limited, data augmentation is often imported to expand the size of raw dataset [22]. For incompatible data and data

from multiple sources, it is common to face the risk of mismatch on size and range. In this situation, data standardization can be employed to solve this problem [23]. To address the aforementioned issues, AI methods have been frequently applied to data preprocessing tasks in recent years. The following is an introduction to the State of The Art approaches for the four data preprocessing topics: data cleaning, feature engineering, data augmentation, and data standardization.

2.2.1 Data Clearing

In this subsection, two specific directions will be focused on, namely missing data and outliers. Let us start with missing data.

In recent research, the regression model is one of core methods to deal with missing data [24][25]. Jammbe Musoro and Aeilko Zwinderman utilized the Lasso regression technique to develop predictive models based on a cohort of patients with Chronic Obstructive Pulmonary Disease (COPD) [26]. Their work focused on forecasting the outcomes for the Chronic Respiratory Questionnaire dyspnea score, specifically aiming to predict patient status six months into the future. Hengfang Wang and Jae Kwang Kim applied Ridge Regression to tackle nonparametric imputation, leveraging regularized M-estimation techniques within the framework of the reproducing kernel Hilbert space [27]. This approach effectively addresses the complexity of imputing missing data by combining regularization with the flexibility of kernel methods. Zhang Yan, Xu Tianxin and their team applied Multilayer Perceptron (MLP) regressor technology to reconstruct missing data in long-term climate monitoring [28]. This included reconstruction of missing data such as temperature, humidity, wind speed, and sunshine duration. Their research consequences show Lasso, Ridge, and MLP regressors as ML models can be applied in different fields and achieve promising results.

When it comes to outliers, engineers tend to split them out of raw dataset. Random Sample Consensus (RANSAC) is one common method to divide dataset into inliers and outlier [29]. Because it converges fast and convenient to employ, RANSAC remains a popular method in various fields. From recent references, RANSAC has been applied to identity outliers in the classification of tumor samples from synthetic and biomedical datasets which is published by The Cancer Genome Atlas (TCGA) and the UC Irvine ML Repository [30]. Mingheng Zou, Bin Zhou and their collaborators adopted RANSAC algorithm to identify outliers of operational data from wind turbine generators in order to improve the data's availability [31].From the survey on RANSAC algorithm by Martínez Otzeta et al., RANSAC can also be used for finding simple geometric shapes or estimating the best transformation between different views of camera with less influence from outliers [32]. Based on the mentioned literature, it can be concluded that RANSAC is a popular algorithm for outlier identification and has been widely used across various fields.

2.2.2 Feature Engineering

Feature engineering often involves mapping and transformation of raw data into a feature space to enhance the performance of ML models. This process includes transforming or extracting relevant features from the raw dataset. The linear discriminant analysis is one potential method to find optimal linear transformation of dataset from high dimensions to low dimension representation [33][34]. Principal Component Analysis (PCA) is also an important component of feature engineering, but it is one statistical data proceeding method. By compressing high-dimensional raw data into relatively lower dimensions, it can significantly reduce the complexity of the dataset, decrease noise within the data, and accelerate the training time of deep learning models [35]. The technology of PCA keeps consistently popular in recent times for real-world datasets. In recent researches, Jiawei Liu, Hou Kang and their collaborators adopted PCA to integrate spatial characteristics of soil contamination aligning with linear transformations to evaluate the extent of multi-source soil pollution [36]. Kyamelia Roy and her colleague applied PCA to extract main characteristics of tomato leaf disease data to improve the evaluation accuracy of deep learning model [37]. Driss Hammoumi, Hefdhallah Al-Aizari, and their team utilized PCA technology to study the seasonal changes in surface water quality of the Nador Canal in Morocco. They concluded that the main sources of pollution were particles of magnesium, sodium, and calcium both in summer and winter [38].For Feature Clustering tasks, many scholars apply PCA and K-means as a combination to work on unsupervised learning tasks [39]. Festa Davide and their team used PCA and K-means to explore the detection of InSAR time series pattern [40]. Jansson Nils and their collaborators applied PCA and K-means to detect Zn skarn deposits and industrial carbonates of Sala area in Sweden [41]. Related literature presents that the combination of PCA nad K-means is one potential method to realize Feature Clustering.

2.2.3 Data Augmentation

When training a model using a dataset, it is often found that the size of the dataset is insufficient. However, the cost of data collection can be very high, especially in fields like military, medicine, and business. These datasets are usually difficult to collect a large size in a short period of time due to legal restrictions or individual willingness. Data augmentation allows us to expand the dataset without increasing the number of original data samples and improving the generalization ability of the model. [22] Common data augmentation techniques include augmenting images, text, and audio data.

For image data, the most common augmentation techniques involve applying transformations such as rotation, cropping, and scaling to the existing images to generate new ones. As Kiran Maharana and his team mentioned in their review, methods like flipping and rotating with slight degrees can effectively expand the dataset without distorting the original data [19]. In addition to using image processing techniques, ML and AI methods can also be employed to expand the dataset. A common approach is using Generative Adversarial Networks (GANs) [42]. In recent research, Jordan Bird and colleagues utilized GANs to generate more fruit data samples, which ultimately improved the

model's detection rate of defective fruits [43]. Ye Zhang, Zhixiang Wang, and their collaborators used GANs to augment a medical image dataset with a limited number of samples, and they were able to generate data that is very similar to the original images [44]. Zhao Zhang and his team also utilized the GANs structure to augment the Rice Leaf Disease Image Data. The augmented dataset resulted in a model trained on ResNet-18 with an accuracy that was 4.57% higher than that trained on the original dataset [45]. Gans Architecture generally provides wider size of raw dataset which can potentially improve the quality of training results.

For text data, there are now more modern methods to expand the dataset in recent five years, such as using Generative Pre-trained Transformer (GPT) to generate similar text data. Keith Cochran and his team utilized GPT-3.5 to generate student responses in two different domains, and used the generated data to train a multilingual Bidirectional Encoder Representations from Transformers (BERT) model. When the generated text was used as part of the dataset, it improved the performance of the trained model [46]. Salvador Balkus and Donghui Yan used GPT-3 to generate additional datasets in order to improve the accuracy of two classifiers for short text classification tasks [47]. Huanhuan Zhao, Haihua Chen, and their team used GPT-3.5 to augment the samples in the Reuters news data and the Mitigation dataset. Their results show that the new data generated by GPT-3.5 consistently improved the classification performance of the model on these two datasets [48]. As a highly successful generative AI model, GPT is very well-suited for data augmentation in text datasets.

For audio data augmentation, common process firstly visualizes the raw audio signals by converting them into Mel-spectrograms and then uses these spectrograms as features. This approach allows us to transform audio tasks into image-based tasks. Image processing techniques such as translation, rotation, and scaling can then be applied to perform data augmentation on the spectrograms [49]. Moreover, GAN is also suitable for generating additional Mel-spectrograms and completes the augmentation of the visualized audio data. Based on previous strategy, Fei Ma, Yang Li and their team adopted GAN architecture to enlarge size of emotion dataset and relieved the imblance between samples with different labels [50].

2.2.4 Data Standardization

In the process of data standardization, it typically compresses data into a specified range or adjust it to follow a certain mathematical distribution (e.g., a normal distribution). From the perspective of data modality, data can be divided into single modal and multimodal data. Single modal data refers to data with a single form, such as images, text, or audio. In contrast, multimodal data is a set of data describing the same target, for instance, an image and its descriptive text about the same object [51].

For single modal data, Min-Max scaling [52] and Z-score standardization [53] are two commonly used standardization methods. However, for multimodal data, it is often necessary to use pre-trained feature extraction models to obtain global features with the same dimensions. Currently, popular

models for extracting image and text features are Vision Transformer (ViT) [54] and BERT [55], respectively, both of which are Transformer-based models.

After feature extraction, standardization is typically performed. Many large models are trained using contrastive learning after extracting features with ViT and BERT and standardizing them. Contrastive Language-Image Pre-training (CLIP) [56] and Grounded Language-Image Pre-training (GLIP) [57] are representative examples of image-text multimodal models. The architecture of CLIP model has been illustrated in Figure 2.4. With the combination of ViT and BERT, CLIP can extract local and global information from both texts and images. [56]

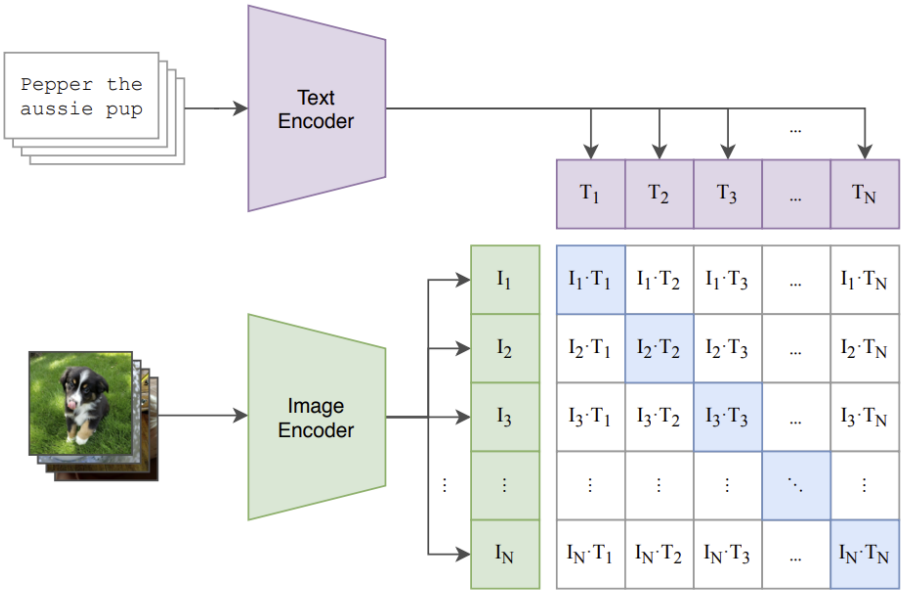

Figure 2.4: `CLIP Model Architecture [56]`

3 Data Clearing

With the increasing capacity of computing units and storage devices, a great deal of larger and more complex datasets are available to analysis engineering. However, the requirement of detection and modification process also faces higher quality challenge and frequent technology iteration. In the recent years, the common potential types of issues encountered by engineers during the data preprocessing period are missing data and outliers. Analysts have to perform data cleaning before utilizing the datasets for database research. The chapter will introduce the definition, causes, potential solutions and analyze the performance of our experiments for missing data and outliers individually.

3.1 Missing Data

In statistics and data science tasks, "missing data" is a very typical problem. The causes of this error can include measurement errors, data recording loss, or incomplete experiments. However, based on the relationship between the missing values and the dataset itself, the missing values can be classified into three categories: Missing Completely at Random (MCAR), Missing at Random (MAR), and Missing Not at Random (MNAR) [58].

3.1.1 Categories of Missing Data and Multiple Imputation

According to the definitions of three kinds of missing data from Donald Rubin [59], they can be distinguished by their relations to observed and unobserved variables as Table 3.1 shows.

Category	Description
MCAR	the missing data is unrelated to any variables, including both observed and unobserved variables
MAR	the missing data is related only to the observed data and is unrelated to the unobserved data
MNAR	the missing data is related to the unobserved data, and even after considering the observed data, the probability of missing is still related to the unobserved data

Table 3.1: Categories of Missing Data

Since the above conceptual introduction might seem abstract, providing an example can enhance understanding. In the following demonstration, we will use data based on a normal distribution as raw data. The reason for choosing a normal distribution is that its characteristics are relatively simple and easy to interpret. In the analysis of missing data, selecting a normal distribution allows us to more clearly observe the impact of different types of missing data (MCAR, MAR, and MNAR) on

the dataset. Based on this framework, let us assume we have a set of numbers(1,000 samples) that follow a normal distribution, falling within the interval [-1, 1] on the x-axis. The y-axis is then derived from a linear transformation of the x-axis data, with added noise. In this scenario, the three types of missing data can be described as follows in Table 3.2. Our initial assumption is that there is no error in the values of X, and all missing data are reflected on the y-axis.

Category	Description	Specific Condition
MCAR	Randomly selects a proportion of the data and sets their Y values to NaN, unrelated to the distribution of X and Y	The Y values of some data points are missing at random. The points with missing values account for 20% of the total data points, and this miss is unrelated to any observed or unobserved features in the dataset.
MAR	Conditionally and randomly sets some Y values to NaN based on X, related to X but not to Y	Approximately 30% of Y values will be set to NaN in samples where $X > 0$.
MNAR	Directly sets Y values to NaN based on the magnitude of Y, related to the values of Y itself	Missing data occurs when Y values are large and is unrelated to X (e.g., when the absolute value of Y is greater than 0.8, Y values are set to NaN).

Table 3.2: Specific Examples of Missing Data

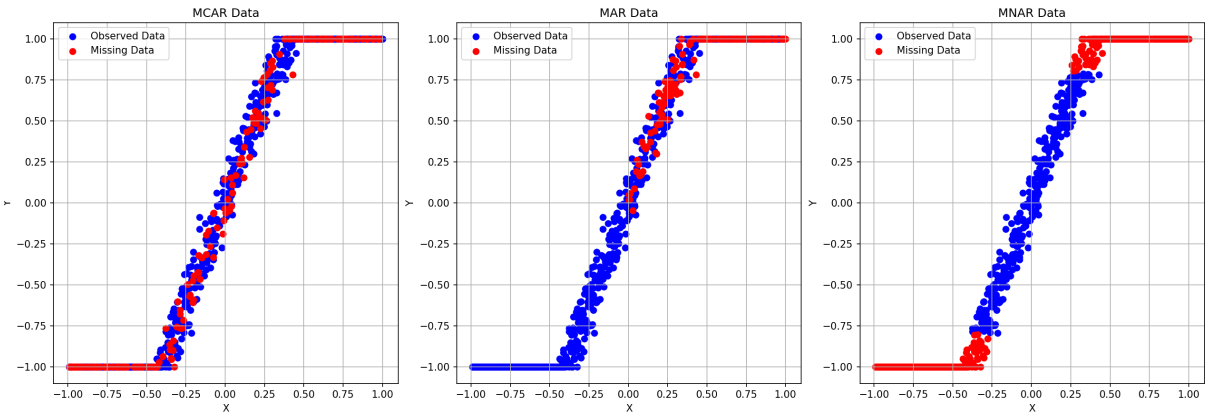

Figure 3.1: `Missing Data Illustration`

Based on the specific conditions, the Figure 3.1, which presents missing data and true value, can be illustrated as above. In the region of statistics, a commonly used method for addressing missing data is Multiple Imputation [60]. As a statistical method, Multiple Imputation fills in the missing data by calculating several plausible values and generate multiple complete datasets. However, it is not

reasonable to all three types of missing data.

For MCAR (Missing Completely at Random) and MAR (Missing at Random) problems, Multiple Imputation is able to effectively simulate the original data. In the case of MCAR, the data is missing completely at random. It means that the missing values are unrelated to any other components of the data. Therefore, the imputed data can serve as an unbiased estimate of the original data. For MAR, Multiple Imputation can also effectively restore the dataset because the missing data depends on the observed data. When using the existing observed data to estimate the missing values, it becomes easier to provide reasonable imputation results.

However, Multiple Imputation is less effective for handling MNAR (Missing Not at Random) issues because, in MNAR problems, the missing data is directly related to the missing values themselves. For example, in the previous case, the missing Y-axis values corresponded to the larger absolute values of Y, which were unrelated to X. In such cases, Multiple Imputation tends to fail.

3.1.2 Regressor Models for MCAR, MAR and MNAR problems

As mentioned above, common Multiple Imputation methods are not effective in solving MNAR problems. On the other hand, regression models have a well-established previous cases of being applied to address these three types of missing data issues. And the comparative is carried out based on three kinds of regressors which are mentioned in the State of the Art chapter because they have latent advantage on dealing with missing data, expecially for MNAR. The first two regressors are Lasso and Ridge Regressors, while the third is the Multilayer Perceptron Regressor. The first two models are linear regression models, whereas the third model can function as a nonlinear regression model by configuring the activation functions.

Lasso and Ridge regression are linear regression models, and their main difference lies in the different regularization terms they use. The mathematical expressions for these two models are shown as Formula 3.1 and 3.2.

Lasso Regression Formula [61]:

$$\hat{\beta}^{lasso} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{3.1}$$

Ridge Regression Formula [62]:

$$\hat{\beta}^{ridge} = \arg\min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{3.2}$$

- $\hat{\beta}$: Estimated coefficient vector ($\hat{\beta}^{lasso}$ for Lasso, $\hat{\beta}^{ridge}$ for Ridge)
- β: Coefficient vector to be estimated

- $\arg\min$: Operation to find the argument that minimizes the given expression
- $\sum_{i=1}^{n}$: Summation over all n observations in the dataset
- y_i: Actual output or target value for the i-th observation
- x_{ij}: Value of the j-th predictor variable for the i-th observation
- λ: Regularization parameter which controls the strength of the penalty
- $\sum_{j=1}^{p}$: Summation over all p predictor variables
- $|\beta_j|$: Absolute value of the j-th coefficient, used in L1 regularization (for Lasso)
- β_j^2: Square of the j-th coefficient, used in L2 regularization (for Ridge)

In Lasso and Ridge regression, the regularization term serves as a penalty that helps control the complexity of the model and prevent overfitting, thereby alleviating the issue where the model performs well on training data but its predictive power decreases on unseen data. As for the nonlinear model MLP, its formula [18] is shown as follows in Formula 3.3:

$$\hat{y} = f\left(\sum_{j=1}^{m} w_j^{(2)} \cdot g\left(\sum_{i=1}^{p} w_{ij}^{(1)} x_i + b_j^{(1)}\right) + b^{(2)}\right) \tag{3.3}$$

- $\hat{y}$: The predicted output of the model.
- f: The activation function of the output layer (e.g., sigmoid, softmax).
- g: The activation function of the hidden layer (e.g., ReLU (Rectified Linear Unit), tanh).
- $w_j^{(2)}$: The weight of the j-th node in the output layer.
- $w_{ij}^{(1)}$: The weight from the i-th node in the input layer to the j-th node in the hidden layer.
- x_i: The i-th input feature.
- $b_j^{(1)}$: The bias term of the j-th node in the hidden layer.
- $b^{(2)}$: The bias term of the output layer.
- m: The number of nodes in the hidden layer.
- p: The number of input features.

Through training the network parameters, the MLP structure can learn the features within the dataset and use this knowledge to predict the missing y-coordinate data based solely on the existing x-coordinate data. The Figure 3.2 below illustrates one typical MLP structure.

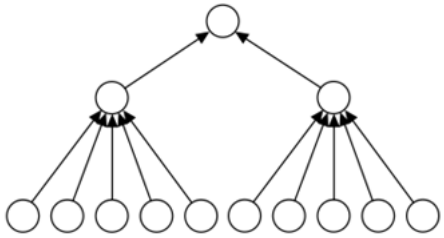

Figure 3.2: `Multilayer Perceptron Architecture`

After providing the alpha values for Ridge Regression and Lasso Regression and assigning the parameters for the MLP, the next step is to demonstrate the regression performance for three different categories of missing data.

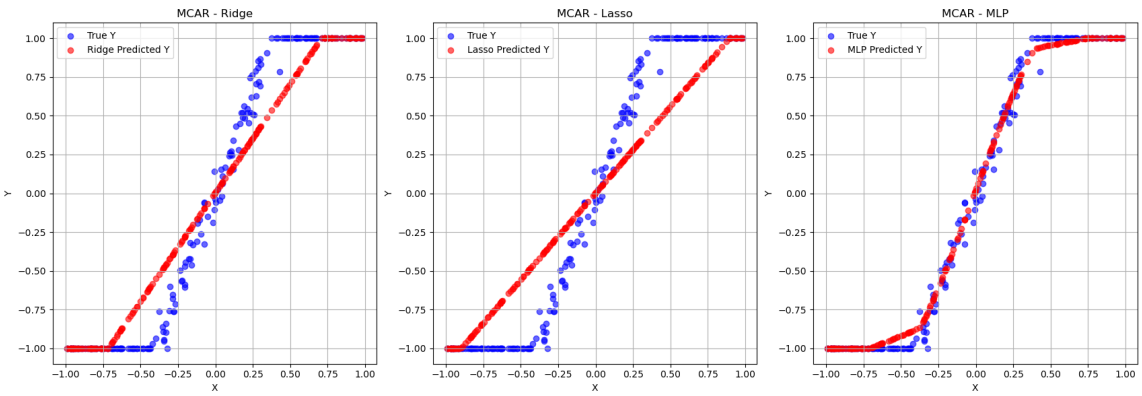

Figure 3.3: `Prediction of MCAR missing values with Ridge, Lasso and MLP`

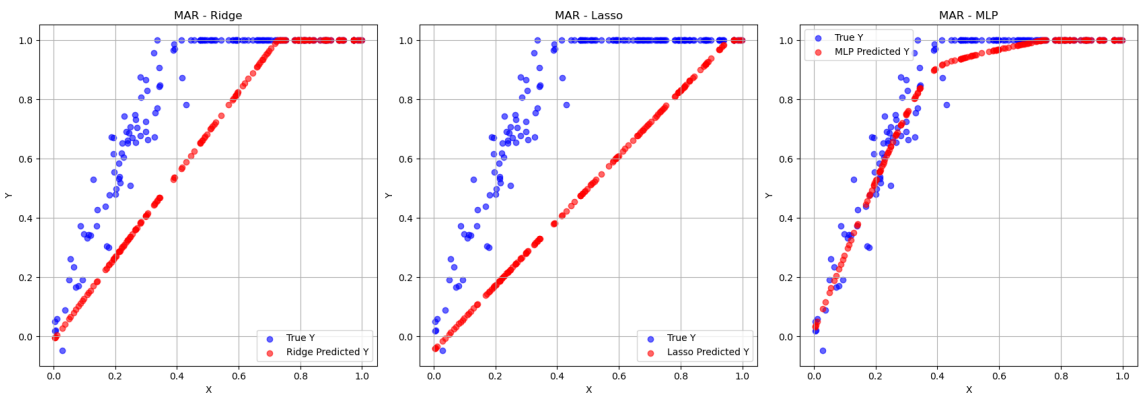

Figure 3.4: `Prediction of MAR missing values with Ridge, Lasso and MLP`

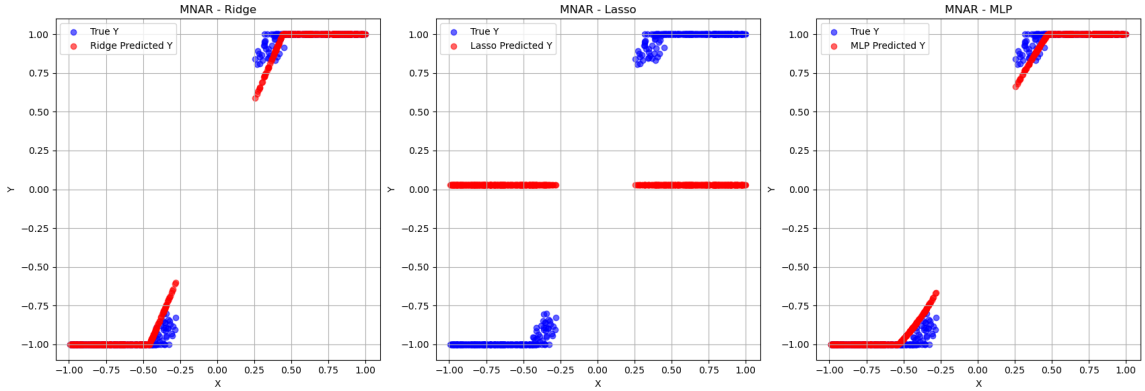

Figure 3.5: `Prediction of MNAR missing values with Ridge, Lasso and MLP`

The Figure 3.3 above clearly show that for the MCAR problem, which has been shown in Figure 3.4, the prediction performance of all three regression models is fairly good, with the MLP model being the most ideal. However, in the MAR problem that is provided in Figure 3.5, the Ridge regression and Lasso regression perform significantly worse compared to the MLP. In the MNAR problem, the Mean Square Error (MSE) values for Ridge and MLP are similar. The specific comparison data can be seen in the following Figure 3.6, where their MSE values are displayed. Overall, it can be concluded that the MLP has a better restoring function for this type of missing data, with the lowest average MSE value over three kinds of missing data (0.0047), compared to Ridge and Lasso regression (0.0343 and 0.3990 respectively).

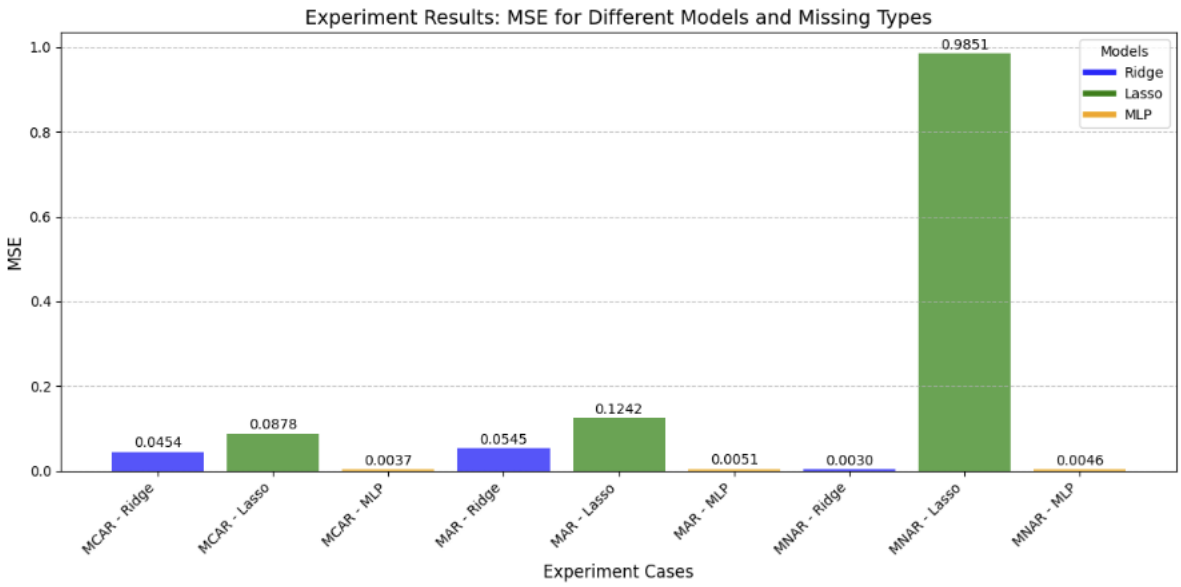

Figure 3.6: `Comparative Experiment MSE`

3.1.3 Advantage and Limitation of three Kinds of Regression Model

The reason why the MLP performs better overall compared to the other two models is that it uses the nonlinear activation function ReLU. The mathematical is shown in Formula 3.4. By adding a nonlinear activation function between two layers of the perceptron, the model is able to learn nonlinear features, which is the greatest advantage of the MLP [63]. However, the downside of the perceptron is that if the number of parameters is small, it can easily overfit during training. This issue is reflected in the high accuracy on the training set, but the estimation performance on unseen new data can drop sharply [64].

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{3.4}$$

The advantage of linear regression models like Lasso and Ridge lies in their ability to suppress overfitting through their built-in regularization terms [65]. However, their main limitation is that they can only provide simple linear models, lacking the ability to learn complex features. They can only adjust coefficients based on the underlying curve, which also restricts the model's subsequent generalization ability and transfer effectiveness. [65]

Nevertheless, regardless of the models mentioned above, their application scope is broader than the original Multiple Imputation, especially when dealing with MNAR problems. From our experiment result, MLP presents the relatively better performance among all three regression models.

3.2 Outliers

Outliers are the values which are unlike other values in the samples and the ignorance of them may lead to incorrect estimates. [66][67] The issue of outliers refers to data points in a dataset that significantly deviate from the other data values. Outliers can severely disrupt the stability of supervised learning in ML and computer vision tasks. The causes of outliers can be categorized into two types: measurement issues and manual causes.

3.2.1 Description and Causes of Outliers

Outliers differ from typical noisy data a lot. They are unlike noise, outliers are data instances that deviate obviously from the predicted standards or expected behavior of the dataset. As Douglas Hawkins suggested [68], if an observation differs from the others, it raises the suspicion that it may have been generated by a different mechanism. In some cases, it may be preferable to identity them instead of deleting them. [69] Actually, there are still wide discussion on how to deal with outliers, but the consensus has not been reached. Common processing methods include correction or removing outliers. [70][71] Considering the situation, the focus of our work is to clarify the outliers.

The causes of outliers can be primarily divided into two types: measurement issues and manual causes. The former includes factors such as instrument error, environmental interference, data transmission error, and sampling issues. These problems often arise due to improper sampling conditions or insufficient accuracy of the measuring instruments.

For manual causes, the reasons behind the generation of outliers can be further divided into subjective and objective factors. Objective factors include errors such as incorrect recording of experimental data due to operational mistakes or unsuitable data conversion, such as failing to convert units. Besides that, subjective factors are mainly related to human bias, such as when the opinions of the test subjects significantly differ from the majority, or when certain data is modified by the analyst based on personal intentions. [72]

3.2.2 RANSAC for Outliers

To detect the outliers on subsequent tasks, a common and straightforward unsupervised learning method called RANSAC is used in many scenarios, such as computer vision [29] and robotics [32]. RANSAC is an iterative algorithm designed to estimate the parameters of a model within a dataset that contains outliers. It effectively identifies and detects many conspicuous outliers from datasets. [29]

Specifically, the core idea of RANSAC is to randomly sample a small subset from the dataset, such as selecting two points, and then setting a threshold on both sides of the line defined by these points. During each iteration, data points within the threshold are considered inliers. If the number of inliers that fit the model is sufficient in appointed iterations, the model is considered a good one. Otherwise, the algorithm continues sampling and fitting until a satisfactory model is found. One demonstration of pseudocode to describe the process follows below. [29]

RANSAC Algorithm :

1. Initialize: best_model = None, best_inliers = 0
2. For $i = 1$ to k:
    a) Randomly select a subset S of n data points
    b) Fit model to the subset S
    c) Compute the error for all data points
    d) Select the inliers as those points with error less than threshold t
    e) If the number of inliers is greater than d and greater than best_inliers:
        i. Update best_model to the current model

ii. Update best_inliers to the number of inliers

3. Return best_model

Similar to previous experiment, we continue to constitute dataset with normal distributed data. The use of the normal distribution as the basis for demonstrating RANSAC's effectiveness in detecting and removing outliers is strategic due to the distribution's symmetrical nature and the clear separation of extreme values. In a normal distribution, outliers which are located far from the mean, are more easily identified. This design can provide an ideal environment for testing RANSAC's performance. By applying RANSAC to this distribution, we can clearly see how it distinguishes and eliminates points that deviate from the central trend. Additionally, since normal distributions are commonly found in real-world data, this approach offers broader applicability and allows us to generalize the results to other scenarios where the data may approximate a normal distribution. This ensures that the analysis of RANSAC's outlier-handling capabilities is both practical and relevant to real-world use cases. In this experiment, noise points were added to a set of data points where X follows a normal distribution within the range [-5,5]. Outliers have 10 times noise value over than normal values which have been shown in Figure 3.7.

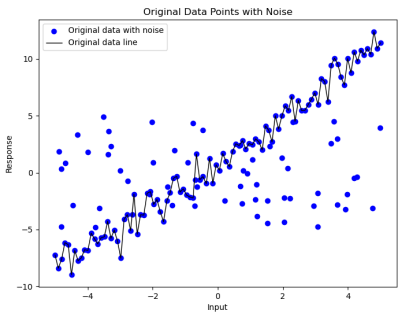

Figure 3.7: `RANSAC Raw Dataset`

As the number of iterations of the RANSAC model increases, the number of points classified as inliers is summarized in Table 3.3 and the iteration of models can present in Figure 3.8:

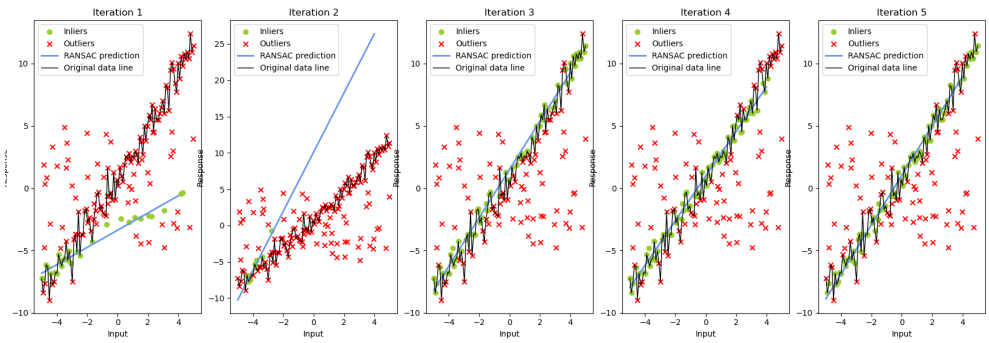

Figure 3.8: `RANSAC Iteration`

Iteration	Inliers
1	27
2	9
3	64
4	67
5	70

Table 3.3: Inliers During RANSAC

3.2.3 Advantage and Limitation of RANSAC

RANSAC is an iterative model computation algorithm that has gained widespread popularity due to its remarkable efficiency and robustness, especially in the presence of noisy data or outliers [73]. Its ability to identify and isolate outliers while still producing reliable results makes it particularly suitable for high-speed operations. This efficiency has made RANSAC a popular algorithm in various fields, particularly in real-time applications where speed and accuracy are critical. Although our demo uses RANSAC for linear model fitting, its model shape can be adjusted as needed, extending to nonlinear model fitting, such as plane fitting, circle fitting, and more.

However, the core drawback of RANSAC is that its hyperparameters lack adaptability and must be carefully selected [74]. Actually, we also fix parameters in the beginning, but it is just picked based on prior experience. In order to realize better performance, more combination of hyper-parameters should be considered. Parameters such as the number of iterations and inlier threshold cannot be changed once the algorithm starts running. Besides that, a sufficient number of inliers is required for the model to converge [75], meaning that prior knowledge of the general type of nonlinear function the model should fit is necessary for effective convergence. Another limitation is RANSAC's poor parallelism, as each iteration can only update the model's position one at a time. To improve inlier count quickly, methods like LO-RANSAC (Locally Optimized RANSAC) [76] refine the model locally after finding an initial model, significantly enhancing final model accuracy. Furthermore, R-RANSAC (Randomized RANSAC) [77] accelerates iteration speed by reducing the number of inlier verifications during the iteration process.

4 Feature Engineering

The purpose of feature engineering is to transform or map features of raw dataset into a feature space to improve the expression ability or learning performance of trained models. This chapter will introduce the operation of feature mapping and feature clustering which can resize the dimensions of raw dataset and cluster samples features respectively.

4.1 Feature Mapping

As a common AI technique, feature mapping transforms raw data into new dimensions, and the space in which these new dimensions exist is called the feature space. This section will provide a detailed introduction to feature mapping for linear discriminant model (LDM).

4.1.1 Linear Discriminant Analysis and Demonstration Experiment

Linear Discriminant Analysis (LDA) is one kind of supervised learning methods and works for dimension reduction. Its goal is to find a projection direction that maximizes the between-class variance for samples with different labels and minimizes the within-class variance for samples with the same labels. After feature mapping, the original data is projected onto a new feature space where it is separated as much as possible [33][34].

To demonstrate the working principle of LDA, here is an example of applying LDA for dimension reduction on the classic Iris dataset [78]. This dataset contains data on three types of iris flowers: Setosa, Versicolour, and Virginica. Each sample has four features: sepal length, sepal width, petal length, and petal width. There are 50 samples for each flower species, so the dataset is balanced. Our goal is to reduce the four-dimensional features to a lower-dimensional space, such as two or three dimensions, while ensuring that the three types of iris flowers remain clearly separated in the reduced space.

The following content is the complete process to compute the feature tensor after compressing dimensions by LDA. To begin with, we need to calculate the Mean Vectors and Global Mean Vector.

For each class C_i, the class mean vector μ_i and global mean vector μ of the dataset are given by Formula 4.1 and 4.2 respectively:

$$\mu_i = \frac{1}{N_i} \sum_{x \in C_i} x \tag{4.1}$$

$$\mu = \frac{1}{N} \sum_{i=1}^{C} N_i \mu_i \tag{4.2}$$

where N is the total number of samples in the dataset, and C is the total number of classes. In this

dataset, N is 150 and C is 3. N_i is the number of samples in class C_i, and $x \in C_i$ represents the feature vectors of samples in that class. The value of N should be 50.

The second part is to calculate the Within-Class Scatter Matrix and the Between-Class Scatter Matrix. The within-class scatter matrix and the between-class scatter matrix, which measures the spread of samples within each class and the separation between different class means, are given by following Formula 4.3 and 4.4 individually:

$$S_W = \sum_{i=1}^{C} \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T \tag{4.3}$$

$$S_B = \sum_{i=1}^{C} N_i (\mu_i - \mu)(\mu_i - \mu)^T \tag{4.4}$$

This matrix is of size $d \times d$, where d is the number of features and it should be 4 here.

Then the third step is to compute Eigenvalue of the features from raw dataset, and the formula is shown as Formula 4.5:

$$S_W^{-1} S_B W = \lambda W \tag{4.5}$$

W is the matrix of eigenvectors (projection directions) and λ are the corresponding eigenvalues, representing the discriminatory power of the projection directions. Both the value of W and λ can be calculated from this function.

After that, the final step is to calculate the lower dimensional feature vectors. The original data matrix X is projected into the new space using the projection matrix W, which has been illustrated in Formula 4.6:

$$X_{\text{LDA}} = XW \tag{4.6}$$

X_{LDA} is the projected data in the new space. And W is the matrix of selected eigenvectors corresponding to the largest eigenvalues. In figure 4.1, red represents Setosa, blue represents Versicolor, and green represents Virginica. LD1 and LD2 are the new features of these samples after the LDA process. The following Figure 4.1 presents the distribution of the three labeled samples in the lower-dimensional feature space after feature compression.

It is clear that the boundaries between the three types of Iris flowers are very distinct, indicating that LDA's feature mapping has a very effective classification method for these three types of samples.

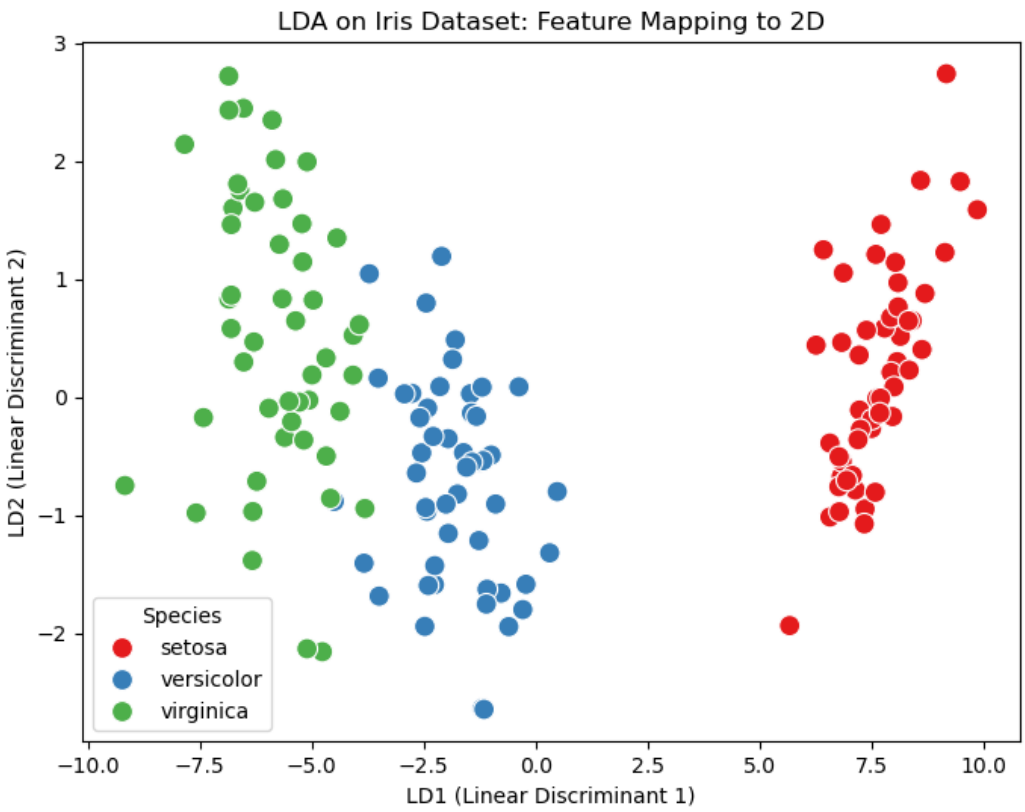

Figure 4.1: `LDA Feature Mapping`

4.1.2 Evaluation on Linear Discriminant Analysis

In this section, by applying feature mapping in LDA, a detailed explanation of how the original feature dimensions are compressed into a lower-dimensional feature space to make the data linearly separable has been illustrated. As a highly efficient supervised learning method, LDA is known for its computational efficiency and its advantages in handling data that follows a normal distribution with slight noise. [79][80]

However, it also has some notable drawbacks. Firstly, LDA is sensitive to the distribution characteristics of the data, and its performance decreases significantly when dealing with data that deviates substantially from a normal distribution [81]. Additionally, it cannot handle nonlinear boundary problems. If the data exhibits nonlinear characteristics, Kernel LDA can serve as an alternative approach [82].

4.2 Feature Clustering

Feature clustering is a common data processing technique that uses AI based methods to cluster data features in a raw dataset, grouping samples with similar characteristics into the same class to facilitate further data mining. Feature clustering typically involves two main operations: first, dimensionality reduction of the data features, followed by the application of a clustering algorithm to group features based on the reduced-dimension data. For feature clustering, a common combination of dimensionality reduction and clustering algorithms is PCA and K-means [83][84].

4.2.1 Mathematical Steps in PCA

PCA focuses on reducing the feature of raw dataset. However, the main difference is that the previous LDA operation is supervised learning operation, while PCA is one unsupervised learning operation [85]. PCA does not require the use of labels during the dimensional reduction process. In this section, the mathematical principles of PCA are introduced. In this optimization strategy, the mean and variance of the data need to be calculated first. Suppose the original dataset is represented by the matrix X, where X is an $n \times d$ matrix representing n samples, each with d features. To center the data, we compute the mean of each feature and subtract it from the data as Formula 4.7. The resulting matrix X' of dataset is calculated as Formula 4.8:

$$\mu_j = \frac{1}{n} \sum_{i=1}^{n} X_{ij} \tag{4.7}$$

$$X'_{ij} = X_{ij} - \mu_j \tag{4.8}$$

where μ_j is the mean of the j-th feature. Then, subtract the mean from each feature to center the dataset. The resulting matrix X' is zero-mean data.

The covariance matrix can capture the linear relationships between different features. The covariance matrix Σ is defined as Formula 4.9:

$$\Sigma = \frac{1}{n-1} X'^T X' \tag{4.9}$$

where Σ is a $d \times d$ matrix, and X'^T is the transpose of X'.

Then the second step is to compute eigenvalues and eigenvectors of covariance matrix. Because covariance matrix is symmetric and can be decomposed into eigenvalues and eigenvectors. We need to find the eigenvalues and eigenvectors of the covariance matrix Σ. Suppose the eigenvalues are $\lambda_1, \lambda_2, \ldots, \lambda_d$, and the corresponding eigenvectors are $v_1, v_2, \ldots, v_d$, which satisfy the equation 4.10:

$$\Sigma v_i = \lambda_i v_i \tag{4.10}$$

where the eigenvalue λ_i represents the variance of the data along the direction of the eigenvector v_i.

After that, the next step is to select principal components. The eigenvalues are arranged in descending order, with larger eigenvalues corresponding to directions of greater variance in the data. We select the top k principal components, corresponding to the k largest eigenvalues and their associated eigenvectors. Let the top k eigenvectors be $v_1, v_2, \ldots, v_k$. These eigenvectors form a matrix V_k of size $d \times k$ as following Formula 4.11:

$$V_k = [v_1, v_2, \ldots, v_k] \tag{4.11}$$

During this step, we can select appointed number vectors from left to right to reconstruct the new dataset. The approximation of the original data from the projected data using the selected principal components has been shown as Formula 4.12:

$$\hat{X} = ZV_k^T + \mu \tag{4.12}$$

where $\hat{X}$ is the reconstructed approximation of the original data, and μ is the mean vector that was subtracted earlier. Z is an $n \times k$ matrix (with n being the number of samples and k the number of principal components chosen).

4.2.2 Mathematical Steps in K-means Algorithm

K-means is a widely used clustering algorithm [86]. After pre-specifying the number of clusters k, each data point will be assigned to one of these clusters. Mathematically, K-means aims to ensure that the features of data points within the same cluster are as similar as possible, while those in different clusters are as distinct as possible.

Similar to last session, the mathematical process of k-means will be introduced. The first step is to initialize the cluster centers. We randomly select k points as centroids. These initial center points can be either from the dataset or randomly generated points.

Then we need to calculate the distance between every sample and all centroids. The formula follows as Formula 4.13:

$$d(x_i, \mu_j) = \sqrt{\sum_{n=1}^{N} (x_{in} - \mu_{jn})^2} \tag{4.13}$$

where x_i is the data point, μ_j is the centroid of the j-th cluster, and N is the number of features. Assign each data point to the cluster with the nearest centroid.

After we assign all data points to their specific clusters, the next step is to update the positions for every centroid as Formula 4.14.

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i \tag{4.14}$$

where C_j represents the j-th cluster, and $|C_j|$ is the number of points in the cluster.

Then we need to repeat the procedures of 'Data Points Assignment' and 'Cluster Centroids Update', until the centroids positions realize convergence state or the maximal update.

4.2.3 Feature Clustering Demonstration based on PCA and K-means

In this section, an example for feature clustering will be illustrated. Firstly, PCA will be used for dimensionality reduction on the Wine dataset. After that, K-means will be conducted to complete feature clustering. The Wine dataset comes from three types of wine produced from grapes grown in the same region in Italy: Barolo, Grignolino, and Barbera. The dataset records 13 parameters of these wines, including Alcohol, Malic Acid, Magnesium, etc. There are 178 samples in total, with 59 for Barolo, 71 for Grignolino, and 48 for Barbera. Unlike the Iris dataset, the Wine dataset is no longer completely balanced and contains more samples. The distribution of this dataset has been shown in Figure 4.2.

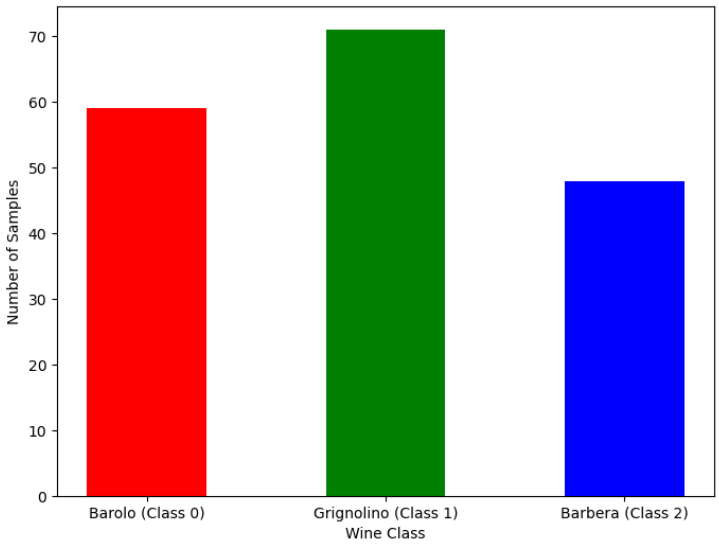

Figure 4.2: `Wine Dataset`

The first step is to transform the 13 features of the original dataset into a data matrix. After performing the mathematical operations from the previous section, we can obtain a distribution map of the data points in the low-dimensional feature space after dimension reduction. The results of two PCA experiments where the features of the Wine dataset are reduced to three dimensions and two dimensions are shown in Figure 4.3 and 4.4.

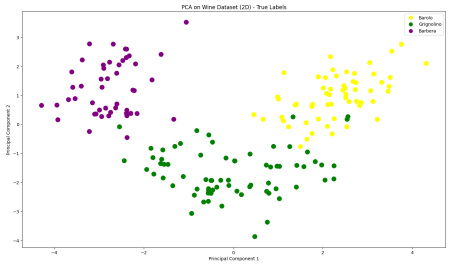

Figure 4.3: `Two-dimensional Dataset under PCA`

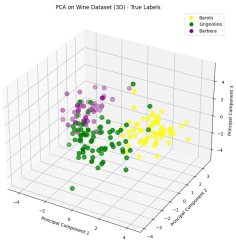

Figure 4.4: `Three-dimensional Dataset under PCA`

As can be seen from the Figure 4.3 and 4.4, the classification performance is better than in two dimensions when PCA compresses the original data into a three-dimensional space. The final result demonstrates that even though the original data has as many as 13 dimensions, PCA can still capture the main features of the dataset and reduce its dimensions.

After PCA operation, K-means is applied to complete feature clustering. We set the value of K to 3 because this number matches the number of classes in the WINE dataset. The initial positions of centers are selected by random seed, which allows the repeatability of the experiments. The experiment results have been shown in the Figure 4.5 and 4.6:

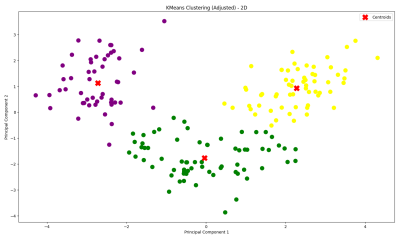

Figure 4.5: `Two-dimensional Dataset K-Means Clustering`

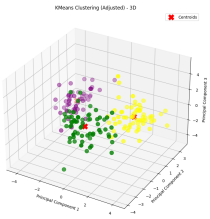

Figure 4.6: `Three-dimensional Dataset K-Means Clustering`

Where Centroids are marked as red 'X'.

In both experiments, the number of correctly clustered samples is 172, resulting in a clustering accuracy of 96.63% for both.

4.2.4 Evaluation on Feature Clustering based on PCA and K-means

As previously mentioned, the most important advantage of PCA is to reduce high-dimensional original data to lower dimensions and enhance the separability of different classes of data [87][88]. During the dimensional reduction process, PCA also reduces the noise present in the original data across

different dimensions. Additionally, datasets processed with PCA tend to speed up the training process of the convergence of subsequent classification models, such as neural networks because of the significant reduction in the overall data size. When it comes to advantages of K-means, the most impressive excellence is the implement efficiency. Since both PCA and K-means do not rely on labeled data, they're classified as unsupervised learning approaches, ideal for exploratory data analysis where patterns are unknown beforehand.

Feature Clustering combined with PCA and K-means are widely used in data prepossessing, but they have some limitations. Even though the limitations of this combination isn't conspicuous in our experiments, we still shouldn't ignore the latent disadvantages. The biggest limitation of PCA is that it assumes the sample distribution of the data follows a linear relationship, so it can only capture linear features in the data. Therefore, PCA is not effective for handling data with nonlinear patterns [89]. In real-world applications, the data distribution is often nonlinear, so PCA may perform poorly in such cases. Additionally, the principal components generated by PCA are linear combinations of the original features, which often lack clear physical which makes the results less interpretable [90]. K-means also has some drawbacks. Firstly, it assumes clusters are spherical shape and uniformly distributed [86], which limits its adaptability to datasets with uncommon shapes or high density distribution. Besides that, K-means is sensitive to the initial centroid positions [91]. Random initial center positions may cause more iterations to realize convergence.

5 Data Augmentation

As ML and computer vision technologies continue to flourish, the demand for the generalization ability of trained models is also increasing. Generalization refers to the performance difference of a trained model when evaluated on training data versus test data [92]. Compared to other methods for enhancing generalization, such as using dropout [93], batch normalization [94] and transfer learning [95], data augmentation has significant advantages. By directly expanding the size of the original dataset, data augmentation fundamentally enhances the generalization ability of the trained model. In this chapter,common data augmentation strategies for three types of files, images, text, and audio will be introduced.

5.1 Image Data Augmentation

Common image-based data augmentation methods can be divided into Image Manipulation, which is based on image transformations, and generative methods, which are AI-based. Common Image Manipulation techniques include rotation, flipping, and cropping [96]. For AI-based method, one common strategy is to apply GAN architecture to extend raw dataset. In this section, both of them will be illustrated [97].

5.1.1 Common Image Manipulation

Image Manipulation generates new samples by applying geometric transformations to images, enhancing the model's generalization ability. Three main image manipulation methods are rotation, flipping, and cropping.

Rotation method is to rotate the image by a relatively small angle, such as 10-30 degrees. Rotation help the model become invariant to the orientation of objects within the image and increase robustness ability to different perspectives. Random rotation angles in determined range can be used before training to improve generalization capacity.

Flipping the image horizontally or vertically can help the model generalize better when objects appear in different orientations. Horizontal flipping is common in tasks where objects can appear in mirrored positions, for example, humans and animals at the view of cameras or in the mirrors. While vertical flipping is less commonly used but can be beneficial in specific scenarios.

Cropping is used for selecting a random or centered portion of the image, resizing it if needed, and using it as the input. This technique helps the model focus on different parts of the object, improving its robustness to scale and viewpoint variations. Center cropping is common in validation, while random cropping is often used for training.

To demonstrate the effects of data augmentation using rotation, flipping, and cropping, we will use the MNIST (Modified National Institute of Standards and Technology) dataset for visual presentation. The MNIST dataset is derived from the original dataset created by the National Institute of Standards and Technology (NIST). It is primarily used for training and testing image processing systems and is considered a benchmark for evaluating various ML models, especially in the context of handwritten digit recognition [98]. For the entire MNIST dataset, the training set includes 60,000 images and test set includes 10,000 images. The size of each image is 28*28 pixels. All the images are based on gray-scale, and the digits are written digit ranging from 0 to 9.

We apply Rotation Flipping and Cropping to the first 8 written digits. The output has been shown in Figure 5.1 as a demonstration.

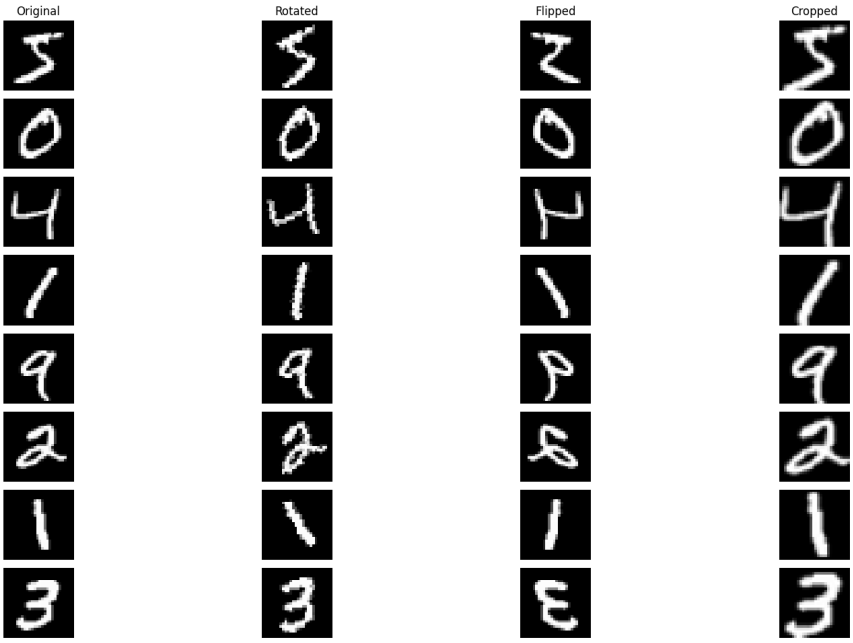

Figure 5.1: `Image Manipulation`

From the displayed result images, it is evident that the new digit samples obtained through image manipulation are not satisfactory. Especially for the flipping method, the generated results completely altered the structure of the digits. As for cropping, it changes the original size of the image, which means that the image must be resized again before being fed into subsequent neural network structures. Even the relatively better-performing rotation method shows poor results for certain samples, such as the third '4' and the seventh '1'. In conclusion, image manipulation is not suitable for data augmentation of character images which needs concrete semantic requirement.

5.1.2 Generative Model to expand dataset

Compared to the image manipulation methods mentioned above, generative models generate entirely new data rather than just performing transformations on the original images. Common generative

models include GANs [99], Diffusion Models [100], and large models based on Transformer structures such as ChatGPT [101] and DALL-E [102]. Although Diffusion Models and Transformer-based large models are more likely to generate higher-quality data, they consume significantly more computational power compared to GANs. Therefore, GANs will be used here to demonstrate the effectiveness of generative models.

This introduction covers the structure of the GAN model and the functions of its main working modules. The structure of GANs differs from that of conventional neural networks. Its adversarial nets consist of a Generative net and a Discriminator net [99].

The purpose of the Generative net is to take randomly generated noise data and output it as a new sample that closely resembles the samples in the training dataset. For the image-related problem we are currently discussing, its task is to output a new image similar to those in the original dataset.

Unlike the generative net, the discriminator net is a binary classification network used to distinguish whether the input sample belongs to the dataset. When the input sample is from the original dataset or is a high-quality sample generated by the generative net, the expected output of the discriminator net is "yes." When the sample quality from the generative net is low, the expected output of the discriminator net is "no.".

This type of network is called an adversarial net which has been shown in Figure 5.2, because it includes a training process where the generative net and the discriminator net compete against each other. They are trained alternately during the training phase. The ultimate goal is for the generator to create artificial images that are so similar to the original dataset that the discriminator finds it difficult to tell them apart.

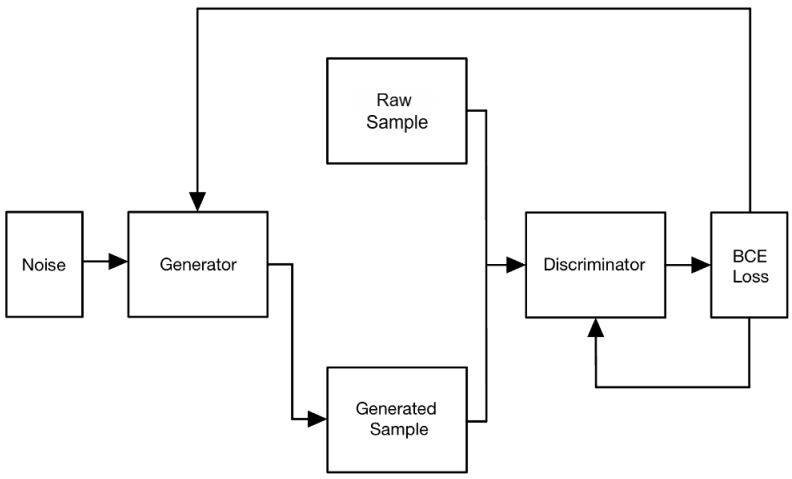

Figure 5.2: GAN Block Diagram

For the GAN architecture, the specific entropy loss formula follows in Formula 5.1 [103]:

$$\text{BCELoss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)] \tag{5.1}$$

- N: Total number of samples.
- y_i: Target label for the ith sample, either 0 or 1.
  - $y_i = 1$: Indicates the sample is from the real data distribution (positive class).
  - $y_i = 0$: Indicates the sample is from the generated data distribution (negative class).
- $\hat{y}_i$: Predicted value for the ith sample, between 0 and 1. 0 and 1 share same meaning with y_i

We used this GAN structure with the training parameters set as follows: 100 training epochs, batch size of 256, learning rate of 0.0005, beta value of 0.5 for the Adam optimizer, and a noise vector dimension of 100.

During each training epoch, even for the same sample, the input noise for the generator is different each time. This is because a new random noise vector is generated at each iteration as the input to the generator. These noise vectors are randomly sampled from the standard normal distribution in order to guarantee that the generated images have randomness each time, such as noisy input illustrated in Figure 5.3.

We can evaluate the performance of the GAN by observing the outputs of the generative network for the same noise input after each epoch update. Below is a set of 64 digit images generated from random noise, which we will use to assess the learning capability of the GAN.

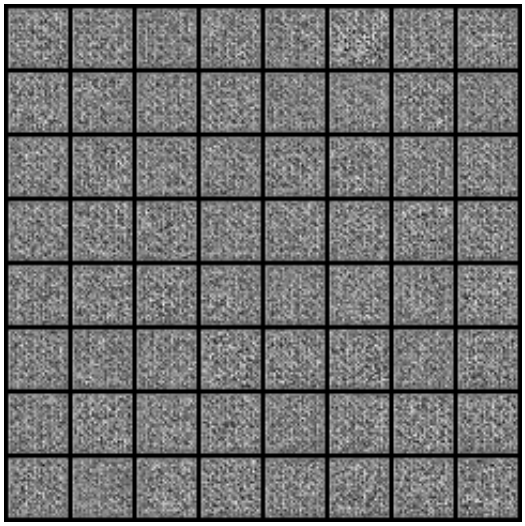

Figure 5.3: `Raw Noisy Data`

The initial training results of the GAN for the first 3 epochs are shown in Figure 5.4. The improvements are quite significant, with the contours of the digits gradually becoming visible. We conducted a total of 100 iterations, with the results of each update shown below for every ten iterations:

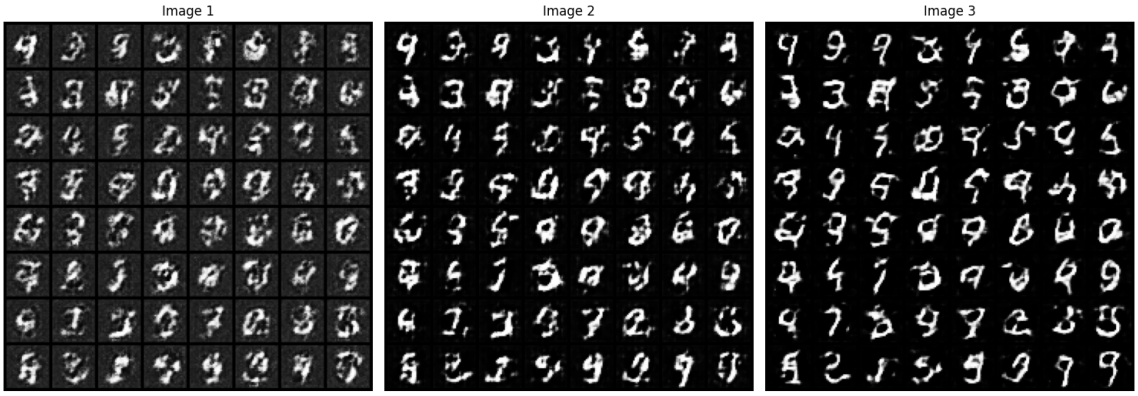

Figure 5.4: The First Three Iteration Results

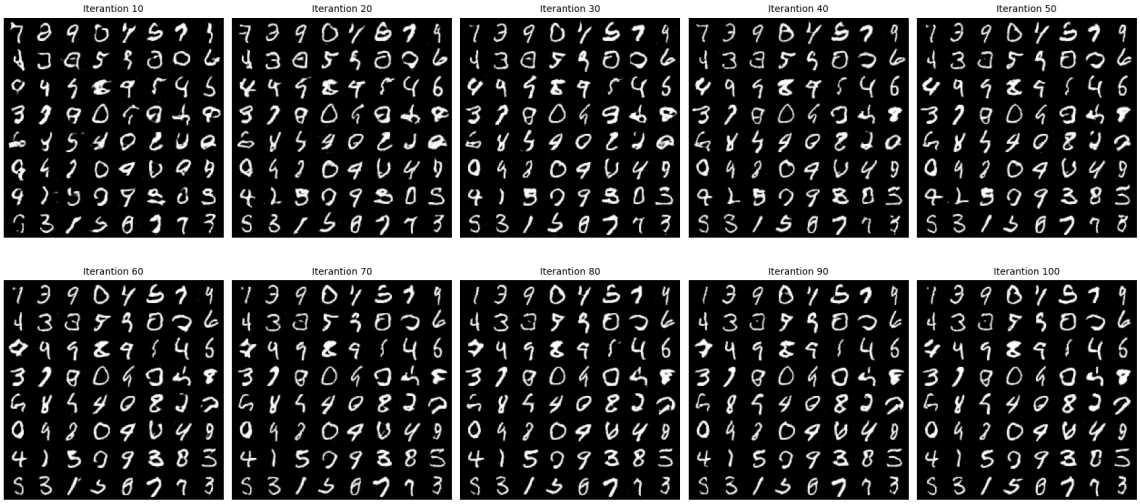

Figure 5.5: Written Digits Update Results

After 100 iterations, the final output that presents in Figure 5.5 has shown many high-quality digits. Even humans would find it difficult to distinguish whether these digits are handwritten or generated artificial data. By selecting suitable handwritten digits, the original dataset can be greatly expanded, which is the goal of using GAN networks for image datasets.

5.1.3 Evaluation on Advantage and Limitation between Common Method and GANs Method

The greatest advantage of image manipulation compared to using generative models like GANs is that it does not require any significant computational resources to train neural networks or perform output calculations. Transformations like rotation and flipping, which are minor modifications of images, can produce samples. The root cause for preferring GANs over image manipulation is that image

manipulation is merely a simple transformation of an existing dataset, meanwhile GANs generate entirely new data samples.

However, this does not mean that GANs are without drawbacks. Since the GANs we train are based on a relatively small dataset like MNIST, which only has tens of thousands of samples, the weaknesses of this structure have not been fully revealed. But in other application areas, some of the limitations of GANs have already been exposed.

Firstly, during our training process, we can observe that the network has difficulty converging, and this issue is quite common [104]. The reason for this problem is that the solution to the GAN training can be represented by a Nash equilibrium, where a point where both the discriminator and generator reach a minimum cost with respect to their parameters. For the Discriminator net and the Generative net, if the performance of either improves, for example, when the Generative net produces more realistic samples, the Discriminator net will tend to find it more difficult to distinguish between real and generated samples. If the Discriminator net performs better, it is often associated with the Generative net producing lower-quality samples. The continuous competition between the two makes it challenging to find a perfect balance point between them.

Another potential issue with GANs is mode collapse [105]. Although we did not encounter this during our training, it is still an usual problem. This error means that for the generative model, regardless of the noise data it receives, the generative net can only produce a limited number of results, losing the diversity of the original dataset. For example, in the case of the MNIST dataset, this error would mean that the GAN only outputs a few types of handwritten digits, such as only generating '2's and '3's. There are some methods that can help mitigate the issue of mode collapse to a certain extent, such as imbalanced loss and zero-centered gradient penalties [106].

5.2 Audio Data Augmentation

In the current data augmentation task, audio data augmentation is also an important part of data preprocessing. Data augmentation has been used in numerous studies, such as for mechanical equipment [107], pathological analysis [108][109], music analysis [110], emotion analysis [111][112] and so on, and has achieved good results. Especially for medical data and other data involving personal privacy, it is very difficult to obtain large datasets due to policy regulations and collection challenges. Therefore, using data augmentation to enhance such data is almost an essential task. In this chapter, common non-AI and AI methods will be explored. Then both their advantages and disadvantages will be evaluated.

5.2.1 Common non-AI Audio Augmentation

Basic data augmentation techniques are usually applied directly to the original audio. These methods include time stretch, noise injection, pitch shift, and volume control. They simulate different audio characteristics and environments, enhancing the model's adaptability to various recording conditions, speech rates, pitch, and volume. These methods help ML models better handle real-world scenarios they may encounter, making them more robust and reliable in practical applications.

Before presenting these methods, it is essential to briefly introduce the dataset used for demonstration: ESC-50 dataset. In the ESC-50 dataset, there are five major categories and fifty subcategories of data, including Animals, Natural Soundscapes and Water Sounds, Human Non-Speech Sounds, Interior/Domestic Sounds, and Exterior/Urban Noises. In this experiment, we selected two categories related to machinery, namely 'Train' and 'Engine', to demonstrate data augmentation. Both of them include 40 samples which last 5 seconds and sampled at 44.1 kHz.

There are several common non-AI methods for audio augmentation, including time stretch, noise injection, and pitch shift.

Time stretching is a specialized method utilizing Short-Time Fourier Transform (STFT), Phase Adjustment, and Inverse Short-Time Fourier Transform (ISTFT). This technique changes the duration of the audio by increasing or decreasing the total number of sampling points in the entire audio file, without altering its frequency, sampling rate, or pitch.

Noise injection is a common approach where white noise or Gaussian noise is added to the audio. By introducing noise, this method simulates how data might sound if recorded in noisy environments, making the model more robust.

Pitch shift is used to change the frequency of the audio along the time dimension. By increasing or decreasing the frequency, it generates sharper or deeper sounds, respectively.

Since the human ear cannot quantify audio information in detail, we use the Mel spectrum to represent the frequency and energy characteristics of audio over time. Research has shown that human hearing ability does not change linearly, and the sensitivity to pitch varies between different frequency ranges. [113][114] For example, humans can distinguish between frequencies in the range of 1000 Hz to 2000 Hz more accurately than those in the range of 10,000 Hz to 11,000 Hz.

To simulate the human ear's nonlinear sensitivity across different frequency ranges, Stanley Smith Stevens and his colleagues proposed the Mel scale in 1937 [115]. A spectrogram with time on the horizontal axis and the Mel scale on the vertical axis is called a Mel spectrogram. The Mel spectrum is generated by dividing the original audio into frames and then performing a STFT on each frame to convert the time-domain signal into a frequency-domain signal. Next, the frequencies are converted to the Mel scale, and the Mel filters and logarithmic transformation are applied to generate the Mel spectrogram.

In the Mel spectrum, the horizontal axis represents time, the vertical axis represents frequency, and the intersection of the two axes indicates the energy value at a specific time and frequency. Here, we set the time length and hop length of each STFT frame to 2048 and 512 sampling points, respectively, and use 128 Mel filters.The Figure 5.6 below demonstrates the Mel Spectrum of a sample from the ESC-50 dataset:

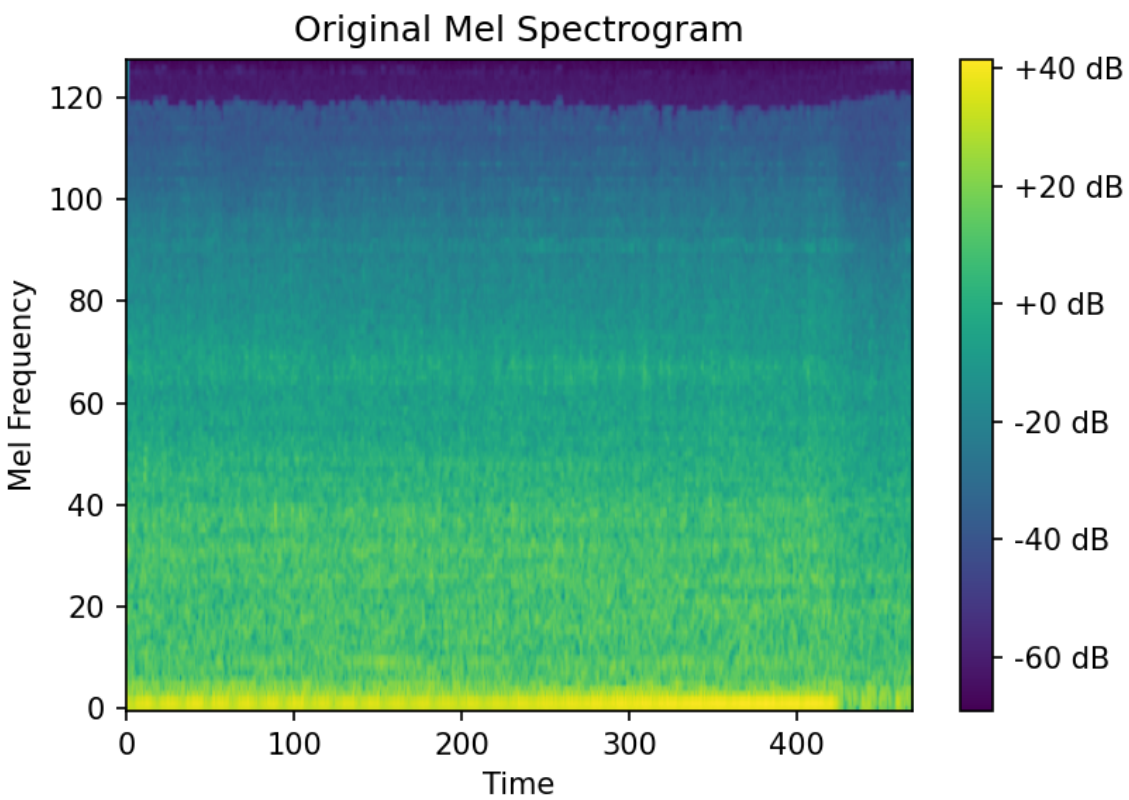

Figure 5.6: `Mel Spectrum Demonstration`

The Figure 5.7 respectively present the effects of time stretch, noise injection, and pitch shift on a randomly given audio sample (1-18527-B-44.wav) in ESC-50 dataset. Specifically, time stretch increases the audio duration by 1.1 times, noise injection adds Gaussian noise with a mean energy of $+0.05$, and pitch shift raises the pitch by 5 semitones.

From the Figure 5.7, it can be observed that the demonstration results of time stretch, noise injection and pitch shift on the Mel spectrum of Figure 5.6. Although these three methods are common approaches for audio data augmentation, there are still obvious drawbacks to this approach. The problem lies in the manually selected parameters, which cannot guarantee appropriateness. The generated data might not belong to the classification group it is supposed to, according to a pre-trained model. The quality of the newly generated data can only be assessed after the generation process is complete. If the initially set parameters are inappropriate, it may result in losing project time and computational resources.

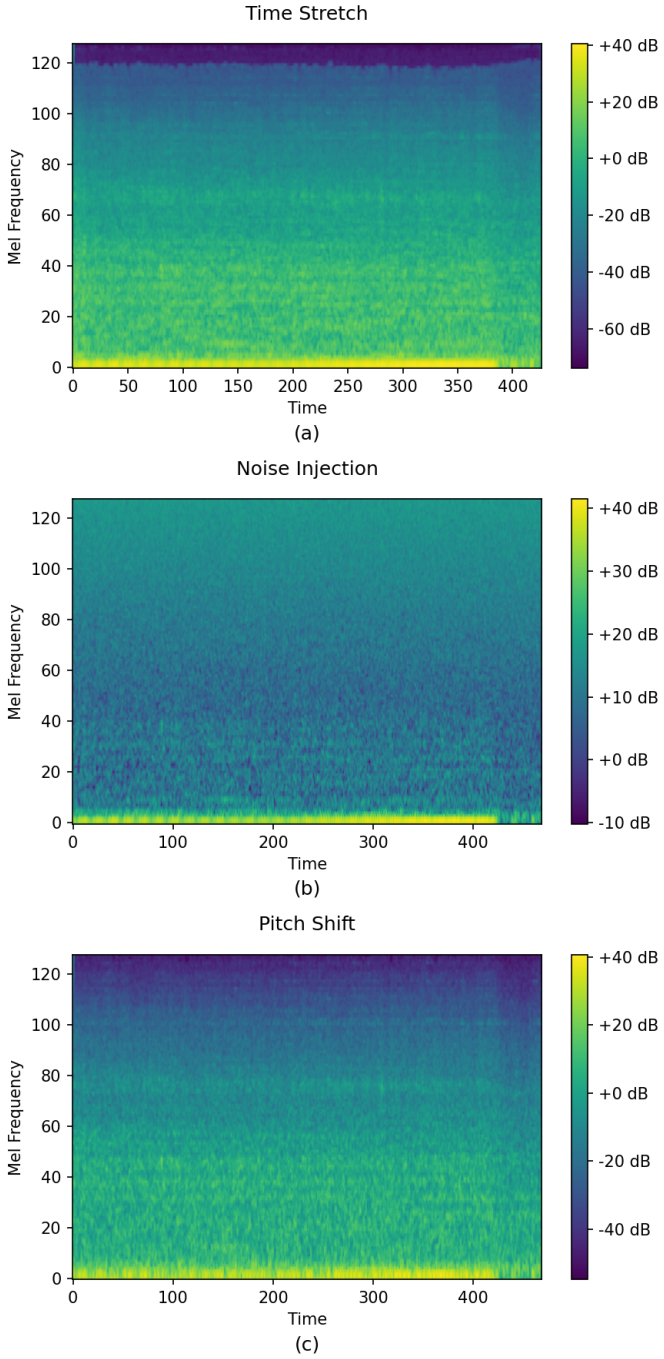

Figure 5.7: `Mel Spectrum Demonstration by Data Augmentation`

(a). Time Stretch by 1.1 time (b). Gaussian Noise with + 0.05 Mean Energy (c). Pitch Shift by + 5 Semitones

5.2.2 RL based Audio Augmentation

To find highly targeted parameters, we employ a relatively simple Policy Gradient method from RL to identify locally optimal solutions within a given range. The advantage of this approach is that it can provide suitable parameter points for data augmentation, enabling a more targeted data augmentation

process.

Before performing RL, we need to construct a discriminator model, which is somewhat similar to the function of the generator mentioned in the previous subsection. Its purpose is to evaluate whether the new audio generated by data augmentation can still be classified into the correct category. However, unlike before, the model here is required to be a pre-trained classification model. Therefore, we trained a Dense Neural Network (DNN) model for target classification. The input features for our DNN model are the mean values of each Mel filter across the entire time domain, resulting in a feature vector with a dimension of [128, 1]. Our dataset consists of 80 samples, with 40 labeled as "train" and the other 40 labeled as "engine." Among the 80 samples, 80% are used as the training set, while 20% are reserved as the validation set (including 8 "train" samples and 8 "engine" samples). The number of training epoch is 100. The structure of this DNN model is shown in the Figure 5.8. The architecture employs three fully connected layers. Between each pair of fully connected layers, a dropout layer with a drop rate of 0.5 and a ReLU activation function are applied.

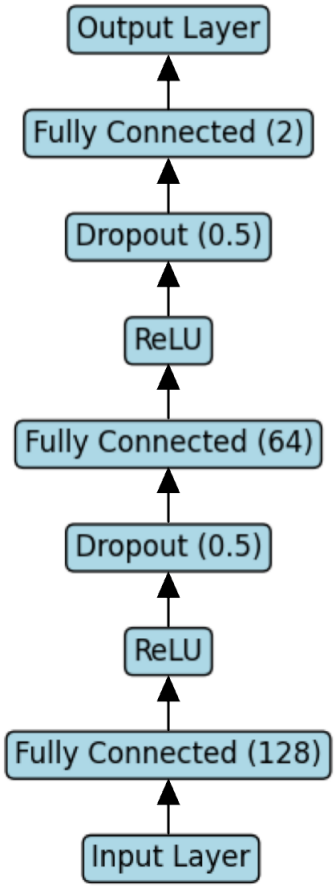

Figure 5.8: `DNN Architecture`

During the training process, the performance of the data on the validation set is shown in the line chart Figure 5.9.

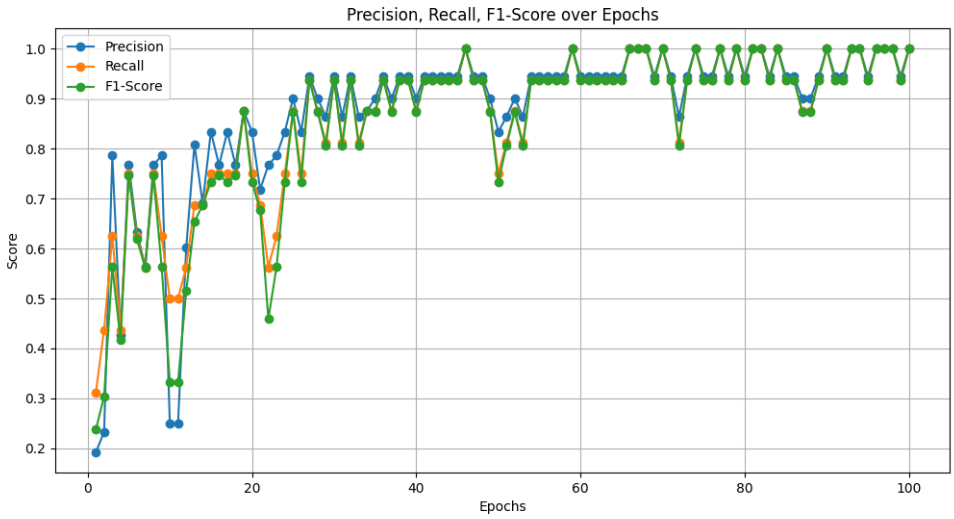

Figure 5.9: `DNN Scores Line Chart`

We tracked three metrics: precision, recall, and F1 score. Their final scores on the validation set are summarized in Table 5.1.

Category	Precision	Recall	F1-Score	Support
Train	1.00	1.00	1.00	8
Engine	1.00	1.00	1.00	8

Table 5.1: Performance metrics on the validation set

Based on the validation set scores, this model performs well in classifying audio into these two categories. Next, we will use it as a discriminator during the RL process to evaluate the parameter updates for the three augmentation methods: Time Stretch, Noise Injection, and Pitch Shift. This will determine whether the audio generated based on the trained parameters can be correctly classified into the corresponding labels.

After completing the training of the classifier, the classifier will be used in RL to calculate the confidence evaluation for data augmentation of all samples. In each training iteration, it outputs the current preference probabilities for hyperparameters of Time Stretch, Noise Injection, and Pitch Shift. These preferences are based on the rewards obtained after augmenting the original audio with these values. The selected action parameters are iterated state values in the RL network. In this setup, for each new sample that is correctly classified, the reward increases by one, with a maximum reward value of 80.

This RL framework includes two networks: the pre-trained DNN classifier and the Policy Network. Their structures are illustrated in Figure 5.10:

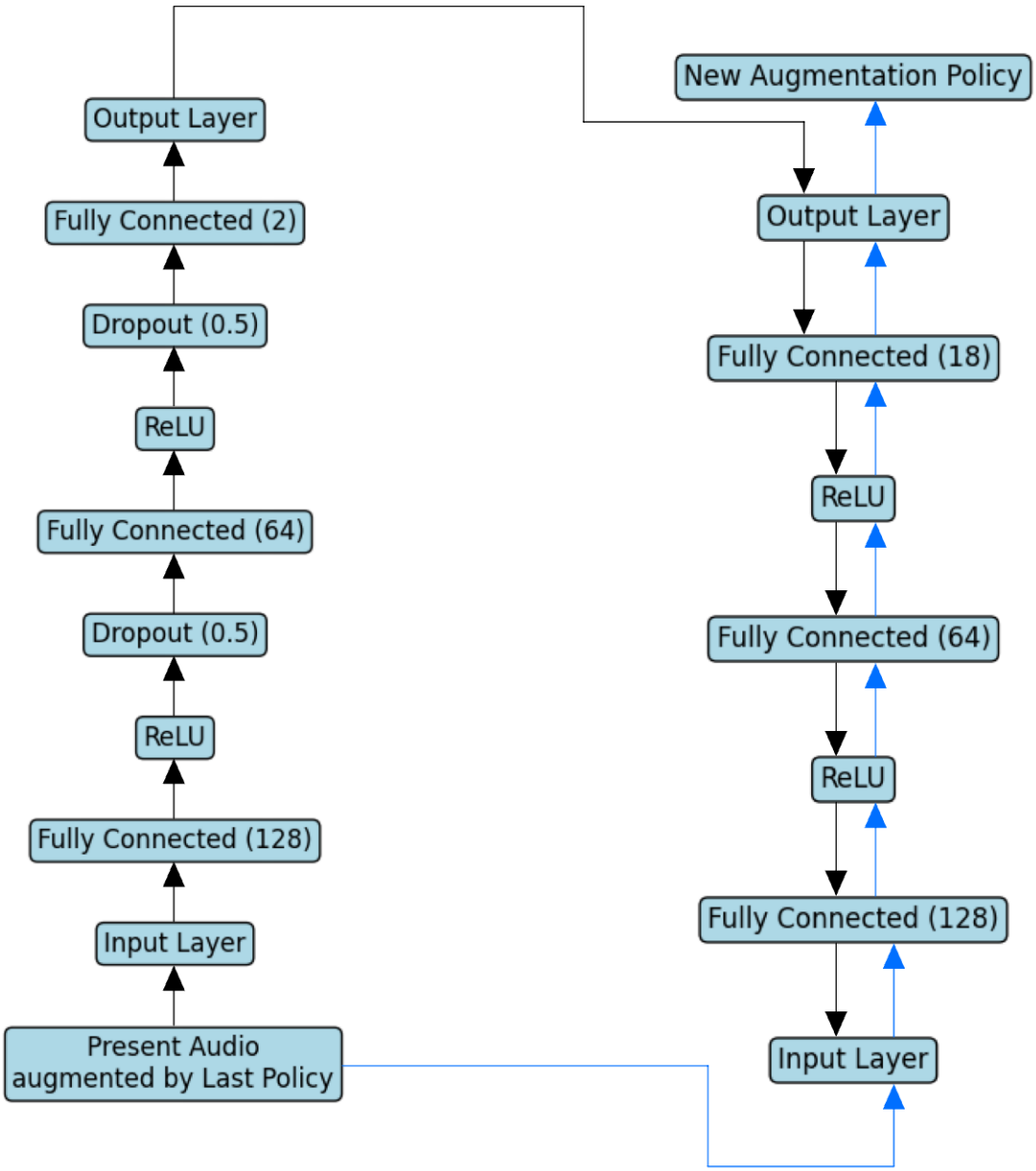

Figure 5.10: `RL Architecture`

In the figure, black lines indicate the forward propagation of the DNN network and the backpropagation logic for the RL network, while blue lines represent the forward propagation of the RL network.

In the RL process, we define three augmentation parameters with alternative options. These are distributed near the output of the original audio, but the output will not simultaneously reflect the original audio when all three hyperparameters are applied. The selected ranges for the three augmentation parameters are as follows in Table 5.2.

During training, the policy network computes the loss function using the log-probability of the selected

Parameter	Values
Pitch Shift Values	-6, -4, -2, 2, 4, 6
Time Stretch Values	0.75, 0.85, 0.95, 1.05, 1.15, 1.25
Noise Level Values	0.0, 0.04, 0.08, 0.12, 0.16, 0.2

Table 5.2: Parameter Candidates for RL

augmentation action combined with the reward outcome. Then it adjusts its parameters through backpropagation. These adjustments alter the probability distribution for selecting future augmentation parameters, increasing the likelihood that successful strategies will be chosen in subsequent iterations.

In this RL network, the loss function can be mathematically represented as Formula 5.2. The purpose of RL is to provide the optimal strategy based on the current sample, thereby enabling the selected action to achieve a higher expected reward. Through Formula 5.2, this loss function directly associates the reward of each iteration with the action suggested by the classifier. Additionally, as a more complex loss function, Actor-Critic Loss [116][117] introduces a value function to further evaluate the quality of an action. However, this approach requires training an additional Critic network, which adds complexity. For our limited dataset, such a more complex structure would be redundant.

$$L = -\left(\log P(a_1) + \log P(a_2) + \log P(a_3)\right) \cdot R \tag{5.2}$$

Where:

- L is the value of the loss function.
- a_1, a_2, a_3 represent the three actions in audio augmentation: pitch shift, time stretch, and noise level, respectively.
- $P(a_i)$ represents the probability of selecting the corresponding augmentation parameter from the policy network.
- R is the reward value, which is 1 if the classifier's prediction is correct, otherwise 0.

During the process of network learning, the probability trends of selecting the three augmentation parameters are shown in Figure 5.11.

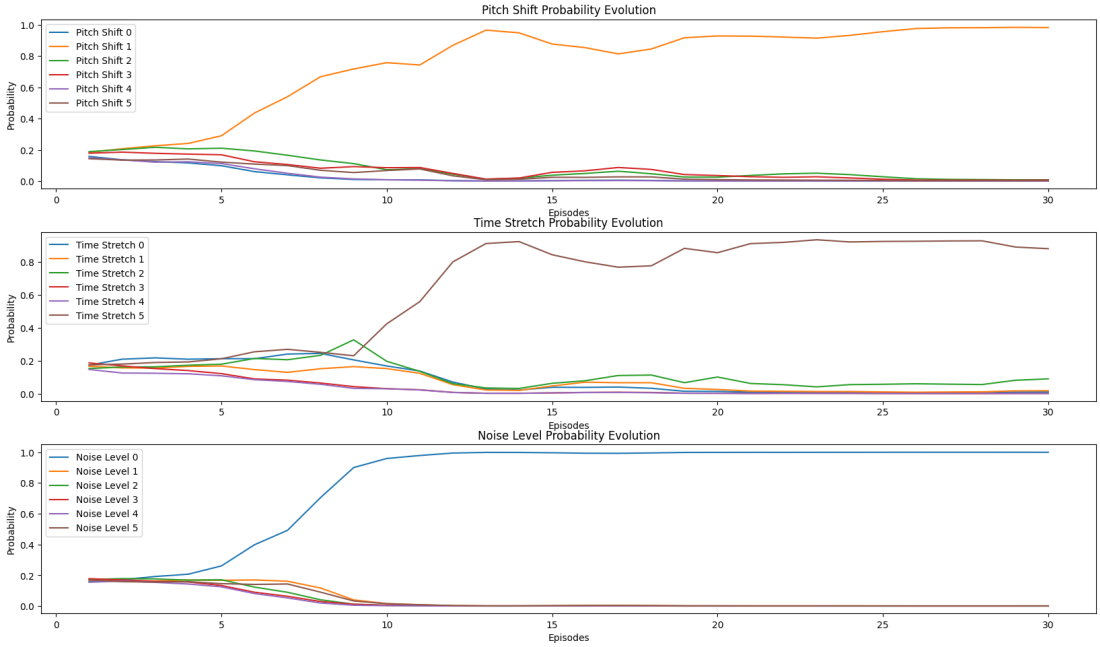

Figure 5.11: `Data Augmentation Probability Evolution`

From Figure 5.12, the change in rewards obtained after applying the three augmentation parameters to the samples during the training phase has presented.

Figure 5.12: `Reward Evolution`

After 30 iterations, the final combination of augmentation parameters is as follows: Pitch Shift: 6 semitones, Time Stretch: 1.25 times the original duration, and Noise Level: 0.0. Under the combined effect of these three augmentation methods, more than 80% of the generated samples are consistently classified correctly into the specific label by the classification network.

5.2.3 Evaluation on Advantage and Limitation between Common Method and RL Method

The most direct advantage of using common non-AI audio augmentation techniques, such as time stretch, noise injection, and pitch shift, is that they do not require strong computational resources and are not limited by the number of samples [118]. However, the downside is the difficulty in determining the specific augmentation parameters to use and the process to find suitable augmentation parameters is complicated and time-consuming [119]. These parameters are often selected based on experience and lack mathematical interpretability. [120][121] Therefore, directly using such data augmentation methods is a common approach for datasets with low augmentation requirements.

For datasets with higher augmentation requirements, where it is essential to ensure that the augmented samples align with the characteristics of template samples, ML methods are a better choice. In the previous section, we used a GAN network to perform augmentation on a dataset with tens of thousands of samples. However, for this chapter, with fewer than a hundred samples, the GAN network's generator struggles to learn the features of the original dataset due to the lack of generalized samples. We addressed this issue by using RL, which allowed us to learn the parameter values for augmentation methods and explore which augmentation parameters are most beneficial for producing high-quality audio samples. This makes data augmentation for small datasets feasible.

Compared to the previously mentioned non-AI methods, the disadvantage of the RL approach lies in its need for more computational power and time to first train a classification model and then train a RL network. In this situation, we need to train two models based on the raw dataset. Considering the dataset is lightweight and relatively small, the additional computational cost and complexity introduced by this RL method are not particularly significant. Its advantage lies in its ability to tailor augmentation parameters to the dataset's characteristics, providing justification for the selected parameters instead of relying solely on experience or random choices to expand the dataset [122].Moreover, our RL approach is based on jointly learning from multiple data augmentation methods applied to the original samples, which further enhances the robustness of the newly generated data. This multi-augmentation strategy has also been reflected in recent research [123].

5.3 Text Data Augmentation

Text Audio Augmentation is applied to expand the size of text dataset when the dataset only has limited samples. Common methods include Neural augmentation, Back-translation augmentation, Style augmentation and Synonym Replacement [124][125]. But considering about the thriving development of generative and pretrained models, transferring mature architectures or import their Application Programming Interface (API) can be viewed as potential methods [126].

5.3.1 Text Augmentation based on Generative Pre-trained Transformer

In this subsection, we conduct experiments of text augmentation based on the well-known Generative Pre-trained Transformer(GPT) version-3.5. GPT is a series of generative models which were trained based on transformer architecture [127] and functions include content generation, sentiment analysis, data collection and so on [128].

In our experiment, we directly utilized the GPT-3.5-turbo API to perform a demonstration of text augmentation. During the experiment, we used three different descriptions of Annealing, Quenching and Tempering with long sentence structures to observe the similar samples generated by GPT. For each sentence structure, five semantically similar sentences were generated.

Below are the original text samples used for text augmentation:

1. Annealing is a heat treatment method used in metallurgy or materials engineering to alter the microstructure of materials, thereby modifying mechanical properties such as hardness and strength.
2. Quenching is a heat treatment process applied to metals and glass to enhance the strength and hardness of steel and cast iron.
3. Tempering involves reheating quenched steel to an appropriate temperature below the austenitic transformation point to reduce brittleness and improve toughness.

Their respective transformation which produced by GPT presents below:

Group 1:

1. Annealing, a heat treatment technique employed in metallurgy and materials engineering, is utilized to change the microstructure of materials, hence adjusting mechanical properties like hardness and strength.
2. In metallurgy and materials engineering, annealing is a method of heat treatment that is applied to modify the microstructure of materials, thus changing mechanical attributes such as hardness and strength.
3. Annealing, a process commonly used in metallurgy and materials engineering, involves heating materials to alter their microstructure, consequently adjusting properties like hardness and strength.
4. Metallurgists and materials engineers often utilize annealing, a heat treatment process, to transform the microstructure of materials and modify mechanical characteristics such as hardness and strength.
5. In the realm of metallurgy and materials engineering, annealing is a process that modifies the

microstructure of materials, impacting properties such as hardness and strength.

Group 2:

1. The process of quenching is used on metals and glass to increase the strength and hardness of steel and cast iron through heat treatment.
2. Quenching is a heat treatment technique utilized on metals and glass to improve the strength and hardness of steel and cast iron.
3. Enhancing the strength and hardness of steel and cast iron is achieved through the heat treatment process known as quenching on metals and glass.
4. Quenching is a method of heat treatment that is employed on metals and glass to boost the strength and hardness of steel and cast iron.
5. The application of quenching on metals and glass is a heat treatment process used to elevate the strength and hardness of steel and cast iron.

Group 3:

1. Tempering includes heating hardened steel to a specific temperature below the austenitic transformation point to enhance toughness and decrease brittleness.
2. Tempering is the process of reheating quenched steel to a temperature below the austenitic transformation point in order to enhance toughness and reduce brittleness.
3. To improve toughness and reduce brittleness, tempered steel is reheated to a temperature below the austenitic transformation point.
4. The act of reheating quenched steel to a temperature below the austenitic transformation point is known as tempering, which aims to enhance toughness and decrease brittleness.
5. Tempering is a technique that involves heating hardened steel to a specific temperature below the austenitic transformation point, thus improving toughness and reducing brittleness.

As demonstrated in the newly generated texts above, all of them exhibit good readability and are similar in content to the original sentences. To better evaluate the generated texts, the BERTScore F1 value will be used to measure the similarity between the generated texts and the original text. BERTScore evaluates the similarity of each token in the candidate sentence to every token in the reference sentence. [129] From the experimental results, it can be observed that all BERTScore F1 scores are above 0.9, and the average F1 score for each set of experiments exceeds 0.95 which have been shown in Table 5.3.

Even though the descriptions of processing techniques are inherently specialized and technical, GPT still manages to perform remarkably well.

Experiment	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Average
Annealing	0.9758	0.9519	0.9595	0.9348	0.9423	0.9529
Quenching	0.9566	0.9874	0.9347	0.9806	0.9593	0.9637
Tempering	0.9680	0.9712	0.9443	0.9464	0.9540	0.9568

Table 5.3: Experimental BERTScore F1 Scores for Annealing, Quenching, and Tempering

5.3.2 Evaluation on text data augmentation based on Generative Pre-trained Transformer

Using GPT's API for text data augmentation offers two significant advantages. Firstly, it can generate text that aligns with the semantic context of the original input. This is attributed to GPT's generalization capability, which stems from its training on a vast corpus of data. Secondly, GPT can perform consistently well even in highly specialized domains. As demonstrated in our example of text augmentation for descriptive texts on manufacturing processes, GPT shows potential ability to understand industry-specific terminology and performing data augmentation effectively in these cases. [130][131][132]

When it comes to the drawbacks of GPT-generated text, even though no such issues occurred in our experiments, it does not mean they are nonexistent. GPT may produce logically inconsistent or overfitted content, particularly in areas where the model has not undergone fine-grained learning. Additionally, a critical risk that must be taken seriously is the bias inherent in the model. Since GPT's training data heavily relies on information from the internet, which often contains biases, the text generation process may amplify these preexisting biases. [133][134][135]

6 Data Standardization

Data Standardization is an important and commonly used data preprocessing technique that can be applied to various modalities such as images, text, and audio. Its primary purpose is to transform raw data into a unified scale or mathematical distribution. Some commonly used mathematical models, such as SVM (Support Vector Machine) [136] and certain neural networks [137], assume that input data has already undergone standardization. Additionally, in deep learning optimization algorithms, Data Standardization can accelerate the training of model parameters and reduce convergence time. In this chapter, we will discuss the standardization of single modal and multimodal data in section 6.1 and 6.2 respectively.

6.1 Single Modal Data Standardization

Single Modal Data is the data which only has one type feature, such as image, audio or text. Compared to multimodal data, this type of data does not require alignment with other modalities. The standardization methods commonly applied to it are Z-score standardization or Min-Max standardization.

The operational methods of these two approaches are entirely different. Z-score first calculates the mean and standard deviation of the data, then transforms the raw data into standardized data with a mean of 0 and a standard deviation of 1. Compared to Min-Max standardization, Z-score is less affected by outliers because its adjustments are based on the mean and standard deviation of the entire dataset. The mathematical formula for Z-score is as follows in Formula 6.1 [138]:

$$z = \frac{x - \mu}{\sigma} \tag{6.1}$$

Where:

- z is the standardized value (Z-score).
- x is the raw data value.
- μ is the mean of the data.
- σ is the standard deviation of the data.

On the other hand, Min-Max compresses the raw data into a specified range, such as $[-1, 1]$ or $[0, 1]$. While this method ensures that the data remains within the defined range, it is more susceptible to the influence of outliers. The mathematical formula for Min-Max is shown in Formula 6.2 [139].

$$x_{\text{normalized}} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{6.2}$$

Where:

- $x_{\text{normalized}}$ is the normalized value after Min-Max scaling.
- x is the raw data point.
- $\min(x)$ is the minimum value of the feature in the dataset.
- $\max(x)$ is the maximum value of the feature in the dataset.

6.2 Multimodal Data Standardization

For single-modal data, the standardization process typically involves directly adjusting the raw data using mathematical methods. However, for multi-modal data, it is more common to first use pre-trained models to extract features from the data, followed by necessary standardization based on alignment. As a result, the standardization process for multi-modal data is relatively more complicated than single modal data and often requires the integration of ML models, including steps such as alignment of dimension and unit.

To illustrate the detailed process of multi-modal data standardization, we choose the commonly used image-text data pair for demonstration. Grayscale images are typically represented as single-layer rectangular pixel matrices, whereas RGB images consist of three layers of matrices of the same size. For text data, the raw form is the text itself. The first step in standardization is to represent these two entirely different data types as tensors of the same dimension.

Here, we refer to the input framework of the image-text multi-modal large model CLIP [56] and GLIP [57]. For images, we use a pre-trained Vision Transformer (ViT), and for text, we use the BERT model. Both models are pre-configured to output global feature representations. ViT extracts the global features of an image into a tensor of shape $[768, 1]$, while BERT processes the text into a tensor of the same dimensions.

The image dataset we selected is CIFAR-10. This dataset consists of 60,000 color images, with 50,000 used as the training set and 10,000 as the test set. The dataset contains images of ten different object categories, such as airplane, automobile, and ship, with 6,000 samples per category. The data format is 32×32 pixel RGB images. Before feeding samples into ViT, images will be resized to 224×224, because ViT was trained based on 224×224 pixel samples and standard size can provide more stable global feature.

In the demonstration, the first five automobile images from the training set are processed by ViT to extract their global features, with each image corresponding to one tensor. The text tensors are manually created, providing keywords such as "car" and the color of the samples. For example, "Here is a red car" can serve as a text sample. Then, BERT is used to convert the five text descriptions into five tensors. On this basis, the global features extracted by the ViT and BERT models are

standardized using the method introduced in the previous section, resulting in standardized feature values for the image-text pairs. The demonstration of image-text matches has been shown as Figure 6.1:

Figure 6.1: `Image-Text Matches Demonstration`

After extracting global features of the images using the ViT model and the global features of the text using the BERT model, the next step is to standardize the two 768-dimensional tensors contained in the image-text matches. In this context, we need to make the two tensors in each data pair as similar as possible. Compared to Z-score standardization, the Min-Max method can constrain the numerical range of the tensor components. Therefore, the Min-Max method will be used to perform the standardization and output standardized multimodal feature tensors.

6.3 Evaluation on AI/ML Influence on Data Standardization

In modern data standardization processes, AI/ML has had a significant impact on this progression, particularly for multimodal data. Pre-trained models have the capability to extract global features from different modalities and align them within the same dimensional space. The use of such transformer-based models has greatly reduced the complexity of standardizing multimodal data, providing a strong foundation for subsequent multimodal model training [54]. However, for single modal data, while there are cases where global features are extracted using such models before standardization, this step is not strictly necessary. Transformer models impose higher computational and processing demands, making direct data-based standardization a more mainstream approach for single modal large-scale datasets [140][141].

7 Experiments Results

Although the design framework for all experiments has already been documented in previous chapters, this chapter revisits the earlier experiments to provide readers with a more direct understanding of our work. It organizes descriptions of all the experiments conducteed, the algorithms applied, and the datasets used.

Data Preprocessing Task	**Topics**	**Experiment Methods**
Data Cleaning	Missing Data	Regression Model
	Outliers	RANSAC
Feature Engineering	Feature Mapping	Linear Discriminant Model
	Feature Clustering	PCA and K-means
Data Augmentation	Image Data Augmentation	GANs
	Audio Data Augmentation	DNN and RL
	Text Data Augmentation	GPT API
Data Standardization	Multimodal Data Standardization	ViT and BERT

Table 7.1: Overview of Tasks, Topics, and Experiment Methods

In this thesis, four data preprocessing topics, Data Cleaning, Feature Engineering, Data Augmentation and Data Standardization were explored extensively. Both the subtasks and methods are shown in Table 7.1.

In our experiments, various datasets have been applied to demonstrate the performance of specific methods. For Missing Data and Outliers dataset, we used randomly samples which corresponded to Gaussian distribution to present the generalization ability of regression and Ransac methods. In feature mapping tasks, Iris and Circular dataset were applied to express the mapping performance by LDM. Then Wine dataset was imported to illustrate the result of feature clustering by PCA and K-means. In data augmentation tasks, we used MNIST, ESC-50 and Spontaneous Text as image,audio and text data augmentation dataset respectively. For data standardization topic, we

employed attention-based feature extraction models ViT and BERT to show the performance of ML models' function in mutilmodal data standardization process. All of the methods and specific datasets which we used in our experiments have been shown in Table 7.2.

Topics	**Experiment Methods**	**Datasets**
Missing Data	Regression Model based on Lasso, Ridge and MLP	Normal distribution within the range [-1, 1]
Outliers	RANSAC	Normal distribution within the range [-5, 5]
Feature Mapping	Linear Discriminant Model	Iris Dataset
Feature Clustering	PCA and K-means	Wine Dataset
Image Data Augmentation	GANs	MNIST
Audio Data Augmentation	DNN and RL	ESC-50
Text Data Augmentation	GPT API	Spontaneous Text
Multimodal Data Standardization	ViT and BERT	CIFAR-10

Table 7.2: Overview of Topics, Methods, and Datasets

In order to present the reproducing ability for our results, the key parameters which were set in our experiments are also listed as following content. The technical parameters include but not limited to parameters for training neural networks and specific pretrained model version and ML significant index. The content of key technical settings has presented in Table 7.3.

Topics	Experiment Methods	Parametrisation Settings
Missing Data	Regression Model based on Lasso, Ridge, and MLP	Lasso Alpha: 1.0; Ridge Alpha: 0.1; MLP: Activation Function: ReLU; Optimizer: Adam
Outliers	RANSAC	The minimum samples required for fitting the model: 2; residual threshold: 1.0
Feature Mapping	Linear Discriminant Model	Targeted Projection Dimension: 2
Feature Clustering	PCA and K-means	PCA Targeted Dimensions: 2 and 3; Clusters: 3
Image Data Augmentation	GANs	Batch size: 100; Training epochs: 100; lr: 5e-4; Beta1: 0.5
Audio Data Augmentation	DNN and RL	DNN: Adam Optimizer; lr: 1e-3; RL: Adam Optimizer; lr: 1e-3
Text Data Augmentation	GPT API	GPT Version: gpt-3.5-turbo
Multimodal Data Standardization	ViT and BERT	ViT Version: vit-base-patch16-224; BERT Version: bert-base-uncased

Table 7.3: Topics, Experiment Methods, and Parametrisation References

Since we conducted numerous experiments on a variety of datasets, it is necessary to summarize the results of these experiments individually.

In the Data Cleaning task, our experimental results demonstrated that the MLP regression method can infer missing data values based on existing information and accomplish the task of data imputation. For the identification of outliers, RANSAC effectively iterates to include more inliers within the model's maximum boundaries, while leaving outliers outside the boundaries.

In Feature Engineering, we discussed two subtasks: Feature Mapping and Feature Clustering. Through feature mapping experiments on the Iris dataset and Circular Dataset, we transformed the features of these datasets into a state more suitable for subsequent training and classification models. For the Feature Clustering problem, we utilized PCA and K-means, two unsupervised learning methods, to perform dimensionality reduction and clustering on the Wine dataset, where each sample has 13

features. This clustering method is well-suited for clustering unlabeled datasets, enabling subsequent labeling of the data.

In Data Augmentation, we performed augmentation on three types of data with different data types: images, audio, and text. First, we used GANs to generate new samples to supplement the handwritten digits dataset. Then, for scenarios with a limited number of samples where GANs are challenging to train, we employed RL to augment audio samples. In this case, we did not directly generate new samples but instead used RL to select the most suitable combination of existing data for augmentation, applying techniques such as noise injection, time stretching, and pitch shifting. The effectiveness of this approach was evaluated by the total reward calculated using a pre-trained DNN. This complex method allows for effective data augmentation even when the dataset contains only a few dozen samples. For text data, we utilized the GPT-3.5-turbo model to expand textual samples. The generated sentences retained similar semantic information, demonstrating the model's effectiveness in supplementing textual data.

In our final topic, Data Standardization, we leveraged ViT and BERT to extract features from multimodal data such as image-text pairs, followed by standardization processes. This approach facilitated subsequent contrastive training. Modern models such as CLIP and GLIP are excellent examples of large multimodal models trained using contrastive learning methods.

Our experiments and the literature reveal several limitations in the methods, some of which are common across different approaches. PCA and LDA assume specific data distributions, such as normalization, and their performance declines when the dataset deviates significantly from these assumptions. Ridge, LASSO, and MLP regression work well with linear data, while MLP is better suited for nonlinear data. RANSAC and K-means are highly sensitive to initial parameter settings, including thresholds and the number of centers. Neural network models primarily face challenges related to computational resources. GAN-based and RL-based data augmentation, as well as models like GPT, BERT, and ViT, require significantly more resources compared to non-deep learning methods.

8 Conclusion, Limitation and Outlook

8.1 Conclusion

This thesis has explored the essential role of data preprocessing in ensuring the effectiveness of AI and ML applications. Through practical coding implementations and relevant scientific literature, the potential advantages of employing AI and ML methods to conduct data preprocessing. Specifically, the experimental results indicate that AI and ML technologies have positively impacted the four selected topics: Data Cleaning, Feature Engineering, Data Augmentation, and Data Standardization. The research result demonstrated how various AI-driven techniques, such as regression models, RANSAC, GANs, and pre-trained models like ViT, BERT and other explored methods, contribute to the improvement of data quality.

After completing each experiment, we conducted evaluations of the AI/ML methods used, based on the experimental results and relevant scientific literature, to analyze the strengths and weaknesses of these methods.

In conclusion, the findings of this thesis underscore the critical interplay between data preprocessing and the performance of AI-driven systems and provide a reference for approaches to be adopted when addressing data preprocessing needs.

8.2 Limitation

Although we conducted extensive experiments on all research topics to illustrate how AI/ML methods can positively impact the complicated tasks of data preprocessing, there are still some limitations which require improvement.

Firstly, we only selected four of the common issues in data preprocessing: Data Cleaning, Feature Engineering, Data Augmentation and Data Standardization. However, the four topics are not enough to fully present the wide range of tasks that AI/ML methods can accomplish in the field of data preprocessing.

Then, the datasets which we used in our experiments were relatively lightweight and not particularly complex. The raw datasets from real-world are often challenging for ML methods. So it is essential to finetune methods based on the specific characteristics instead of transferring our models directly.

Last but not least, we didn't explore enough AI/ML-based methods for each subtask. In most cases, we analyzed only one potential solution for one specific subtask. But there may be multiple approaches to tackle data preprocessing for them.

8.3 Outlook

Building upon the limitations identified in Limitation, there are several directions for future work that could extend and enhance the findings presented in this thesis.

Firstly, future research could broaden the scope of data preprocessing tasks to encompass a more comprehensive range of issues beyond the four current explored topics. Secondly, exploring larger, more complicated, and diverse real-world datasets is essential for evaluating the robustness and scalability of AI/ML-based preprocessing techniques. Real-world datasets often present challenges such as noise, imbalance, and complex patterns, which require models to be specifically adapted and fine-tuned. Considering more datasets would help improve the robustness of trained models. Finally, for each specific subtask, more potential solutions should be considered to be explored and expand the candidates to work as options.

Bibliography

[1] Susan Moore. *How to Create a Business Case for Data Quality Improvement*. Accessed: September 19, 2024. 2018. URL: https://www.gartner.com/smarterwithgartner/how-to-create-a-business-case-for-data-quality-improvement.

[2] Thomas C. Redman Tadhg Nagle and David Sammon. *Only 3% of Companies' Data Meets Basic Quality Standards*. Accessed: September 19, 2024. 2017. URL: https://vtw.nl/data/media/files/10_-_Only_3_pct_of_Data_Meets_Basic_Quality_Standards_-_HBR.pdf.

[3] Inc. Anaconda. *Annual Report 2020*. Accessed: September 19, 2024. 2020. URL: https://www.anaconda.com/resources/whitepapers/state-of-data-science-2020.

[4] Inc. Anaconda. *Annual Report 2022*. Accessed: September 19, 2024. 2022. URL: https://www.anaconda.com/resources/whitepapers/state-of-data-science-report-2022.

[5] Inc. Anaconda. *State of Data Science 2023*. Accessed: September 19, 2024. 2023. URL: https://www.anaconda.com/lp/state-of-data-science-report-2023.

[6] John McCarthy et al. "What is artificial intelligence". In: (2007). URL: https://cse.unl.edu/~choueiry/S09-476-876/Documents/whatisai.pdf.

[7] Pramila P Shinde and Seema Shah. "A review of machine learning and deep learning applications". In: *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*. IEEE. 2018, pp. 1–6. DOI: 10.1109/ICCUBEA.2018.8697857.

[8] KR1442 Chowdhary and KR Chowdhary. "Natural language processing". In: *Fundamentals of artificial intelligence* (2020), pp. 603–649. DOI: https://doi.org/10.1007/978-81-322-3972-7_19.

[9] Xin Feng et al. "Computer vision algorithms and hardware implementations: A survey". In: *Integration* 69 (2019), pp. 309–320. DOI: https://doi.org/10.1016/j.vlsi.2019.07.005.

[10] Federico Vicentini. "Collaborative robotics: a survey". In: *Journal of Mechanical Design* 143.4 (2021), p. 040802. DOI: https://doi.org/10.1007/s10846-024-02141-z.

[11] Stefano A Bini. "Artificial intelligence, machine learning, deep learning, and cognitive computing: what do these terms mean and how will they impact health care?" In: *The Journal of arthroplasty* 33.8 (2018), pp. 2358–2361. DOI: https://doi.org/10.1016/j.arth.2018.02.067.

[12] C David Naylor. "On the prospects for a (deep) learning health care system". In: *Jama* 320.11 (2018), pp. 1099–1100. DOI: 10.1001/jama.2018.11103.

[13] J Matthew Helm et al. "Machine learning and artificial intelligence: definitions, applications, and future directions". In: *Current reviews in musculoskeletal medicine* 13 (2020), pp. 69–76. DOI: 10.1007/s12178-020-09600-8.

[14] Batta Mahesh. "Machine learning algorithms-a review". In: *International Journal of Science and Research (IJSR).[Internet]* 9.1 (2020), pp. 381–386. DOI: 10.21275/ART20203995.

[15] Salvador García, Julián Luengo, Francisco Herrera, et al. *Data preprocessing in data mining*. Vol. 72. Springer, 2015. DOI: https://doi.org/10.1007/978-3-319-10247-4.

[16] Suad A Alasadi and Wesam S Bhaya. "Review of data preprocessing techniques in data mining". In: *Journal of Engineering and Applied Sciences* 12.16 (2017), pp. 4102–4107. DOI: 10.35629/5252-040512061209.

[17] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. 3rd ed. Burlington, MA: Morgan Kaufmann, 2011. ISBN: 978-0123814791. URL: https://liacs.leidenuniv.nl/~bakkerem2/dbdm2007/05_dbdm2007_Data%20Mining.pdf.

[18] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. New York: Springer, 2006. ISBN: 978-0387310732. DOI: https://link.springer.com/book/9780387310732.

[19] Kiran Maharana, Surajit Mondal, and Bhushankumar Nemade. "A review: Data pre-processing and data augmentation techniques". In: *Global Transitions Proceedings* 3.1 (2022), pp. 91–99. DOI: https://doi.org/10.1016/j.gltp.2022.04.020.

[20] Xu Chu et al. "Data cleaning: Overview and emerging challenges". In: *Proceedings of the 2016 international conference on management of data*. 2016, pp. 2201–2206. DOI: https://doi.org/10.1145/2882903.2912574.

[21] Guozhu Dong and Huan Liu. *Feature engineering for machine learning and data analytics*. CRC press, 2018. DOI: https://doi.org/10.1201/9781315181080.

[22] Alhassan Mumuni and Fuseini Mumuni. "Data augmentation: A comprehensive survey of modern approaches". In: *Array* 16 (2022), p. 100258. DOI: https://doi.org/10.1016/j.array.2022.100258.

[23] Peshawa Jamal Muhammad Ali et al. "Data normalization and standardization: a technical report". In: *Mach Learn Tech Rep* 1.1 (2014), pp. 1–6. DOI: 10.13140/RG.2.2.28948.04489.

[24] James M Robins and Andrea Rotnitzky. "Semiparametric efficiency in multivariate regression models with missing data". In: *Journal of the American Statistical Association* 90.429 (1995), pp. 122–129. DOI: https://doi.org/10.2307/2291135.

[25] Carolina Gil Marcelino et al. "Missing data analysis in regression". In: *Applied Artificial Intelligence* 36.1 (2022), p. 2032925. DOI: https://doi.org/10.1080/08839514.2022.2032925.

[26] Jammbe Z Musoro et al. "Validation of prediction models based on lasso regression with multiply imputed data". In: *BMC medical research methodology* 14 (2014), pp. 1–13. DOI: https://doi.org/10.1186/1471-2288-14-116.

[27] Hengfang Wang and Jae Kwang Kim. "Statistical inference using regularized M-estimation in the reproducing kernel Hilbert space for handling missing data". In: *Annals of the Institute of Statistical Mathematics* 75.6 (2023), pp. 911–929. DOI: https://doi.org/10.1007/s10463-023-00872-8.

[28] Zhang Yan et al. "Deep learning tool: Reconstruction of long missing climate data based on multilayer perceptron (MLP)". In: *EGUsphere* 2024 (2024), pp. 1–26. DOI: https://doi.org/10.5194/egusphere-2024-439,2024.

[29] Konstantinos G Derpanis. "Overview of the RANSAC Algorithm". In: *Image Rochester NY* 4.1 (2010), pp. 2–3. URL: http://www.cse.yorku.ca/~kosta/CompVis_Notes/ransac.pdf.

[30] André Veríssimo et al. "Random Sample Consensus for the Robust Identification of Outliers in Cancer Data". In: *International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics*. Springer. 2019, pp. 108–118. DOI: https://doi.org/10.1007/978-3-030-63061-4_11.

[31] Mingheng Zou et al. "Offshore wind turbine wind speed power anomaly data cleaning method based on RANSAC regression and DBSCAN clustering". In: *2024 4th International Conference on Smart Grid and Renewable Energy (SGRE)*. IEEE. 2024, pp. 1–8. DOI: 10.1109/SGRE59715.2024.10428822.

[32] José María Martínez-Otzeta et al. "Ransac for robotic applications: A survey". In: *Sensors* 23.1 (2022), p. 327. DOI: https://doi.org/10.3390/s23010327.

[33] Tim Verdonck et al. "Special issue on feature engineering editorial". In: *Machine learning* 113.7 (2024), pp. 3917–3928. DOI: https://doi.org/10.1007/s10994-021-06042-2.

[34] Youness Aliyari Ghassabeh, Frank Rudzicz, and Hamid Abrishami Moghaddam. "Fast incremental LDA feature extraction". In: *Pattern Recognition* 48.6 (2015), pp. 1999–2012. DOI: https://doi.org/10.1016/j.patcog.2014.12.012.

[35] Sasan Karamizadeh et al. "An overview of principal component analysis". In: *Journal of signal and information processing* 4.3 (2013), pp. 173–175. DOI: 10.4236/jsip.2013.43B031.

[36] Jiawei Liu et al. "A spatial distribution–Principal component analysis (SD-PCA) model to assess pollution of heavy metals in soil". In: *Science of The Total Environment* 859 (2023), p. 160112. DOI: https://doi.org/10.1016/j.scitotenv.2022.160112.

[37] Kyamelia Roy et al. "Detection of tomato leaf diseases for agro-based industries using novel PCA DeepNet". In: *Ieee Access* 11 (2023), pp. 14983–15001. DOI: 10.1109/ACCESS.2023.3244499.

[38] Driss Hammoumi et al. "Seasonal Variations and Assessment of Surface Water Quality Using Water Quality Index (WQI) and Principal Component Analysis (PCA): A Case Study". In: *Sustainability* 16.13 (2024), p. 5644. DOI: https://doi.org/10.3390/su16135644.

[39] Chris Ding and Xiaofeng He. "K-means clustering via principal component analysis". In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 29. DOI: https://doi.org/10.1145/1015330.1015408.

[40] Davide Festa et al. "Unsupervised detection of InSAR time series patterns based on PCA and K-means clustering". In: *International Journal of Applied Earth Observation and Geoinformation* 118 (2023), p. 103276. DOI: https://doi.org/10.1016/j.jag.2023.103276.

[41] Nils F Jansson et al. "Principal component analysis and K-means clustering as tools during exploration for Zn skarn deposits and industrial carbonates, Sala area, Sweden". In: *Journal of Geochemical Exploration* 233 (2022), p. 106909. DOI: https://doi.org/10.1016/j.gexplo.2021.106909.

[42] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144. DOI: https://doi.org/10.1145/3422622.

[43] Jordan J Bird et al. "Fruit quality and defect image classification with conditional GAN data augmentation". In: *Scientia Horticulturae* 293 (2022), p. 110684. DOI: https://doi.org/10.1016/j.scienta.2021.110684.

[44] Ye Zhang et al. "GAN-based one dimensional medical data augmentation". In: *Soft Computing* 27.15 (2023), pp. 10481–10491. DOI: https://doi.org/10.1007/s00500-023-08345-z.

[45] Zhao Zhang et al. "A high-quality rice leaf disease image data augmentation method based on a dual GAN". In: *IEEE Access* 11 (2023), pp. 21176–21191. DOI: 10.1109/ACCESS.2023.3251098.

[46] Keith Cochran et al. "Improving automated evaluation of student text responses using gpt-3.5 for text data augmentation". In: *International Conference on Artificial Intelligence in Education*. Springer. 2023, pp. 217–228. DOI: https://doi.org/10.1007/978-3-031-36272-9_18.

[47] Salvador V Balkus and Donghui Yan. "Improving short text classification with augmented data using GPT-3". In: *Natural Language Engineering* (2022), pp. 1–30. DOI: 10.1017/S1351324923000438.

[48] Huanhuan Zhao et al. "Improving Text Classification with Large Language Model-Based Data Augmentation". In: *Electronics* 13.13 (2024), p. 2535. DOI: https://doi.org/10.3390/electronics13132535.

[49] Luo Jiqing et al. "Quad-rotor UAV Audio Recognition Based on Mel Spectrum with Binaural Representation and CNN". In: *2021 International Conference on Computer Engineering and Application (ICCEA)*. IEEE. 2021, pp. 285–290. DOI: 10.1109/ICCEA53728.2021.00063.

[50] Fei Ma et al. "Data augmentation for audio-visual emotion recognition with an efficient multimodal conditional GAN". In: *Applied Sciences* 12.1 (2022), p. 527. DOI: https://doi.org/10.3390/app12010527.

[51] Jing Gao et al. "A survey on deep learning for multimodal data fusion". In: *Neural Computation* 32.5 (2020), pp. 829–864. DOI: https://doi.org/10.1162/neco_a_01273.

[52] Jelena Diakonikolas, Constantinos Daskalakis, and Michael I Jordan. "Efficient methods for structured nonconvex-nonconcave min-max optimization". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 2746–2754. DOI: 10.48550/arXiv.2011.00364.

[53] Henderi Henderi, Tri Wahyuningsih, and Efana Rahwanto. "Comparison of Min-Max normalization and Z-Score Normalization in the K-nearest neighbor (kNN) Algorithm to Test the Accuracy of Types of Breast Cancer". In: *International Journal of Informatics and Information Systems* 4.1 (2021), pp. 13–20. DOI: https://doi.org/10.47738/ijiis.v4i1.73.

[54] Kai Han et al. "A survey on vision transformer". In: *IEEE transactions on pattern analysis and machine intelligence* 45.1 (2022), pp. 87–110. DOI: https://doi.org/10.48550/arXiv.2012.12556.

[55] Mikhail V Koroteev. "BERT: a review of applications in natural language processing and understanding". In: *arXiv preprint arXiv:2103.11943* (2021). DOI: https://doi.org/10.48550/arXiv.2103.11943.

[56] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763. DOI: https://doi.org/10.48550/arXiv.2103.00020.

[57] Liunian Harold Li et al. "Grounded language-image pre-training". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10965–10975. DOI: https://doi.org/10.48550/arXiv.2112.03857.

[58] Donald B Rubin. "Inference and missing data". In: *Biometrika* 63.3 (1976), pp. 581–592. DOI: https://doi.org/10.2307/2335739.

[59] Donald B Rubin. *Multiple imputation for nonresponse in surveys*. Vol. 81. John Wiley & Sons, 2004. DOI: 10.1002/9780470316696.

[60] James R Carpenter et al. *Multiple imputation and its application*. John Wiley & Sons, 2023. DOI: 10.1002/9781119942283.

[61] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58.1 (1996), pp. 267–288. DOI: https://doi.org/10.1111/j.2517-6161.1996.tb02080.x.

[62] Arthur E Hoerl and Robert W Kennard. "Ridge regression: Biased estimation for nonorthogonal problems". In: *Technometrics* 12.1 (1970), pp. 55–67. DOI: https://doi.org/10.1080/00401706.1970.10488634.

[63] Fabrice Rossi and Brieuc Conan-Guez. "Functional multi-layer perceptron: a non-linear tool for functional data analysis". In: *Neural networks* 18.1 (2005), pp. 45–60. DOI: https://doi.org/10.1016/j.neunet.2004.07.001.

[64] Joseph Rynkiewicz. "General bound of overfitting for MLP regression models". In: *Neurocomputing* 90 (2012), pp. 106–110. DOI: https://doi.org/10.1016/j.neucom.2011.11.028.

[65] LE Melkumova and S Ya Shatskikh. "Comparing Ridge and LASSO estimators for data analysis". In: *Procedia engineering* 201 (2017), pp. 746–755. DOI: https://doi.org/10.1016/j.proeng.2017.09.615.

[66] Joe H Sullivan, Merrill Warkentin, and Linda Wallace. "So many ways for assessing outliers: What really works and does it matter?" In: *Journal of Business Research* 132 (2021), pp. 530–543. DOI: https://doi.org/10.1016/j.jbusres.2021.03.066.

[67] Ray Chambers, Adao Hentges, and Xinqiang Zhao. "Robust automatic methods for outlier and error detection". In: *Journal of the Royal Statistical Society Series A: Statistics in Society* 167.2 (2004), pp. 323–339. DOI: https://doi.org/10.1111/j.1467-985X.2004.00748.x.

[68] D Hawkins. *Identification of Outliers*. 1980. DOI: https://doi.org/10.1007/978-94-015-3994-4.

[69] Herman Aguinis, Ryan K Gottfredson, and Harry Joo. "Best-practice recommendations for defining, identifying, and handling outliers". In: *Organizational research methods* 16.2 (2013), pp. 270–301. DOI: https://doi.org/10.1177/1094428112470848.

[70] Xavier Romão and Emilia Vasanelli. "Identification and Processing of Outliers". In: *Non-Destructive In Situ Strength Assessment of Concrete: Practical Application of the RILEM TC 249-ISC Recommendations*. Springer, 2021, pp. 161–180. DOI: https://doi.org/10.1007/978-3-030-64900-5_5.

[71] Jason W Osborne and Amy Overbay. "The power of outliers (and why researchers should always check for them)". In: *Practical Assessment, Research, and Evaluation* 9.1 (2019), p. 6. DOI: https://doi.org/10.7275/qf69-7k43.

[72] Egawati Panjei et al. "A survey on outlier explanations". In: *The VLDB Journal* 31.5 (2022), pp. 977–1008. DOI: https://doi.org/10.1007/s00778-021-00721-1.

[73] Ekaterina O Rybakova et al. "PESAC, the generalized framework for RANSAC-based methods on SIMD computing platforms". In: *IEEE Access* (2023). DOI: 10.1109/ACCESS.2023.3301777.

[74] Xi Shen et al. "Ransac-flow: generic two-stage image alignment". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*. Springer. 2020, pp. 618–637. DOI: https://doi.org/10.48550/arXiv.2004.01526.

[75] Bahram Salehi and Sina Jarahizadeh. "Improving the UAV-derived DSM by introducing a modified RANSAC algorithm". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 43 (2022), pp. 147–152. DOI: https://doi.org/10.5194/isprs-archives-XLIII-B2-2022-147-2022.

[76] Ondřej Chum, Jiří Matas, and Josef Kittler. "Locally optimized RANSAC". In: *Pattern Recognition: 25th DAGM Symposium, Magdeburg, Germany, September 10-12, 2003. Proceedings 25*. Springer. 2003, pp. 236–243. DOI: https://doi.org/10.1007/978-3-540-45243-0_31.

[77] Jiri Matas and Ondrej Chum. "Randomized RANSAC with Td, d test". In: *Image and vision computing* 22.10 (2004), pp. 837–842. DOI: https://doi.org/10.1016/j.imavis.2004.02.009.

[78] Ronald A Fisher. "The use of multiple measurements in taxonomic problems". In: *Annals of eugenics* 7.2 (1936), pp. 179–188. DOI: https://doi.org/10.1111/j.1469-1809.1936.tb02137.x.

[79] Shuping Zhao et al. "Linear discriminant analysis". In: *Nature Reviews Methods Primers* 4.1 (2024), p. 70. DOI: https://doi.org/10.1038/s43586-024-00346-y.

[80] Xiaojia Ran and Boxin Nie. "Linear Discriminant Analysis (LDA) based on auxiliary slicing for binary classification data". In: *Highlights in Science, Engineering and Technology* 101 (2024), pp. 778–785. DOI: https://doi.org/10.54097/gh17v872.

[81] Jianhang Zhou et al. "Latent linear discriminant analysis for feature extraction via isometric structural learning". In: *Pattern Recognition* 149 (2024), p. 110218. DOI: https://doi.org/10.1016/j.patcog.2023.110218.

[82] Yong Xu et al. "A fast kernel-based nonlinear discriminant analysis for multi-class problems". In: *Pattern Recognition* 39.6 (2006), pp. 1026–1033. DOI: https://doi.org/10.1016/j.patcog.2005.10.029.

[83] Dan Feldman, Melanie Schmidt, and Christian Sohler. "Turning big data into tiny data: Constant-size coresets for k-means, PCA, and projective clustering". In: *SIAM Journal on Computing* 49.3 (2020), pp. 601–657. DOI: https://doi.org/10.1137/18M1209854.

[84] Soma Bandyopadhyay, SS Thakur, and JK Mandal. "Product recommendation for e-commerce business by applying principal component analysis (PCA) and K-means clustering: benefit for the society". In: *Innovations in Systems and Software Engineering* 17.1 (2021), pp. 45–52. DOI: https://doi.org/10.1007/s11334-020-00372-5.

[85] Michael Greenacre et al. "Principal component analysis". In: *Nature Reviews Methods Primers* 2.1 (2022), p. 100. DOI: https://doi.org/10.1038/s43586-023-00209-y.

[86] Abiodun M Ikotun et al. "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data". In: *Information Sciences* 622 (2023), pp. 178–210. DOI: https://doi.org/10.1016/j.ins.2022.11.139.

[87] Ian T Jolliffe and Jorge Cadima. "Principal component analysis: a review and recent developments". In: *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences* 374.2065 (2016), p. 20150202. DOI: https://doi.org/10.1098/rsta.2015.0202.

[88] Basna Mohammed Salih Hasan and Adnan Mohsin Abdulazeez. "A review of principal component analysis algorithm for dimensionality reduction". In: *Journal of Soft Computing and Data Mining* 2.1 (2021), pp. 20–30. DOI: https://doi.org/10.30880/jscdm.2021.02.01.003.

[89] Diego Bueso, Maria Piles, and Gustau Camps-Valls. "Nonlinear PCA for spatio-temporal analysis of Earth observation data". In: *IEEE Transactions on Geoscience and Remote Sensing* 58.8 (2020), pp. 5752–5763. DOI: 10.1109/TGRS.2020.2969813.

[90] Eran Elhaik. "Principal Component Analyses (PCA)-based findings in population genetic studies are highly biased and must be reevaluated". In: *Scientific Reports* 12.1 (2022), p. 14683. DOI: https://doi.org/10.1038/s41598-022-14395-4.

[91] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. "The k-means algorithm: A comprehensive survey and performance evaluation". In: *Electronics* 9.8 (2020), p. 1295. DOI: https://doi.org/10.3390/electronics9081295.

[92] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning". In: *Journal of big data* 6.1 (2019), pp. 1–48. DOI: https://doi.org/10.1186/s40537-019-0197-0.

[93] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958. URL: https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf.

[94] Christian Garbin, Xingquan Zhu, and Oge Marques. "Dropout vs. batch normalization: an empirical study of their impact to deep learning". In: *Multimedia tools and applications* 79.19 (2020), pp. 12777–12815. DOI: https://doi.org/10.1007/s11042-019-08453-9.

[95] Xiaohan Chen et al. "Deep transfer learning for bearing fault diagnosis: A systematic review since 2016". In: *IEEE Transactions on Instrumentation and Measurement* 72 (2023), pp. 1–21. DOI: 10.1109/TIM.2023.3244237.

[96] Suorong Yang et al. "Image data augmentation for deep learning: A survey". In: *arXiv preprint arXiv:2204.08610* (2022). DOI: 10.1109/TIM.2023.3244237.

[97] Giovanni Mariani et al. "Bagan: Data augmentation with balancing gan". In: *arXiv preprint arXiv:1803.09655* (2018). DOI: https://doi.org/10.48550/arXiv.1803.09655.

[98] Gregory Cohen et al. "EMNIST: Extending MNIST to handwritten letters". In: *2017 international joint conference on neural networks (IJCNN)*. IEEE. 2017, pp. 2921–2926. DOI: 10.1109/IJCNN.2017.7966217.

[99] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014). DOI: https://doi.org/10.48550/arXiv.1406.2661.

[100] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851. DOI: https://doi.org/10.48550/arXiv.2006.11239.

[101] Tianyu Wu et al. "A brief overview of ChatGPT: The history, status quo and potential future development". In: *IEEE/CAA Journal of Automatica Sinica* 10.5 (2023), pp. 1122–1136. DOI: 10.1109/JAS.2023.123618.

[102] Kai-Qing Zhou and Hatem Nabus. "The ethical implications of DALL-E: Opportunities and challenges". In: *Mesopotamian Journal of Computer Science* 2023 (2023), pp. 16–21. DOI: https://doi.org/10.58496/MJCSC/2023/003.

[103] Baoren Xiao, Hao Ni, and Weixin Yang. "MCGAN: Enhancing GAN Training with Regression-Based Generator Loss". In: *arXiv preprint arXiv:2405.17191* (2024). DOI: https://doi.org/10.48550/arXiv.2405.17191.

[104] Hamed Alqahtani, Manolya Kavakli-Thorne, and Gulshan Kumar. "Applications of generative adversarial networks (gans): An updated review". In: *Archives of Computational Methods in Engineering* 28 (2021), pp. 525–552. DOI: https://doi.org/10.1007/s11831-019-09388-y.

[105] Kanglin Liu et al. "Spectral regularization for combating mode collapse in gans". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6382–6390. DOI: https://doi.org/10.48550/arXiv.1807.04015.

[106] Hoang Thanh-Tung and Truyen Tran. "Catastrophic forgetting and mode collapse in GANs". In: *2020 international joint conference on neural networks (ijcnn)*. IEEE. 2020, pp. 1–10. DOI: 10.1109/IJCNN48605.2020.9207181.

[107] Asith Abeysinghe et al. "Data augmentation on convolutional neural networks to classify mechanical noise". In: *Applied Acoustics* 203 (2023), p. 109209. DOI: https://doi.org/10.1016/j.apacoust.2023.109209.

[108] Hung-Chi Chu, Young-Lin Zhang, and Hao-Chu Chiang. "A cnn sound classification mechanism using data augmentation". In: *Sensors* 23.15 (2023), p. 6972. DOI: https://doi.org/10.3390/s23156972.

[109] Tauhidul Islam et al. "A systematic review of deep learning data augmentation in medical imaging: Recent advances and future research directions". In: *Healthcare Analytics* (2024), p. 100340. DOI: https://doi.org/10.1016/j.health.2024.100340.

[110] Matthew C McCallum et al. "On the effect of data-augmentation on local embedding properties in the contrastive learning of music audio representations". In: *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2024, pp. 671–675. DOI: https://doi.org/10.48550/arXiv.2401.08889.

[111] Shing-Tai Pan and Han-Jui Wu. "Performance improvement of speech emotion recognition systems by combining 1D CNN and LSTM with data augmentation". In: *Electronics* 12.11 (2023), p. 2436. DOI: https://doi.org/10.3390/electronics12112436.

[112] Jonathan Christian Setyono and Amalia Zahra. "Data augmentation and enhancement for multimodal speech emotion recognition". In: *Bulletin of Electrical Engineering and Informatics* 12.5 (2023), pp. 3008–3015. DOI: https://doi.org/10.11591/eei.v12i5.5031.

[113] Mikko-Ville Laitinen, Sascha Disch, and Ville Pulkki. "Sensitivity of human hearing to changes in phase spectrum". In: *Journal of the Audio Engineering Society* 61.11 (2013), pp. 860–877. URL: https://aes2.org/publications/elibrary-page/?id=17068.

[114] William M Hartmann. "Pitch, periodicity, and auditory organization". In: *The Journal of the Acoustical Society of America* 100.6 (1996), pp. 3491–3502. DOI: https://doi.org/10.1121/1.417248.

[115] Stanley Smith Stevens, John Volkmann, and Edwin Broomell Newman. "A scale for the measurement of the psychological magnitude pitch". In: *The journal of the acoustical society of america* 8.3 (1937), pp. 185–190. DOI: https://doi.org/10.1121/1.1915893.

[116] Vijay Konda and John Tsitsiklis. "Actor-critic algorithms". In: *Advances in neural information processing systems* 12 (1999). URL: https://proceedings.neurips.cc/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[117] Tuomas Haarnoja et al. "Soft actor-critic algorithms and applications". In: *arXiv preprint arXiv:1812.05905* (2018). DOI: https://doi.org/10.48550/arXiv.1812.05905.

[118] Max Morrison et al. "Neural pitch-shifting and time-stretching with controllable lpcnet". In: *arXiv preprint arXiv:2110.02360* (2021). DOI: https://doi.org/10.48550/arXiv.2110.02360.

[119] Noussa Amer and Hisham Othman. "Time Stretching and Pitch Shifting for Anomalous Sound Detection in Fan Sound Signal using MobileNetV2". In: *2023 Eleventh International Conference on Intelligent Computing and Information Systems (ICICIS)*. IEEE. 2023, pp. 80–86. DOI: 10.1109/ICICIS58388.2023.10391190.

[120] Md Raffael Maruf et al. "Comparative Study of Effective Augmentation Method for Bangla ASR Using Convolutional Neural Network". In: *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2020, Volume 1*. Springer. 2021, pp. 203–213. DOI: https://doi.org/10.1007/978-981-15-9927-9_20.

[121] Marina Eskander and Hisham Othman. "Anomalous Sound Detection for Slider Machine Using Contrastive Learning Under Domain-Shifted Conditions". In: *2024 IEEE 3rd International Conference on Computing and Machine Intelligence (ICMI)*. IEEE. 2024, pp. 1–5. DOI: 10.1109/ICMI60790.2024.10585806.

[122] Sumit Ranjan, Rupayan Chakraborty, and Sunil Kumar Kopparapu. "Reinforcement Learning based Data Augmentation for Noise Robust Speech Emotion Recognition". In: *Proc. Interspeech 2024*. 2024, pp. 1040–1044. DOI: 10.21437/Interspeech.2024-921.

[123] Yanjie Sun et al. "Automated Data Augmentation for Audio Classification". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2024). DOI: 10.1109/TASLP.2024.3402049.

[124] Connor Shorten, Taghi M Khoshgoftaar, and Borko Furht. "Text data augmentation for deep learning". In: *Journal of big Data* 8.1 (2021), p. 101. DOI: https://doi.org/10.1186/s40537-021-00492-0.

[125] Markus Bayer, Marc-André Kaufhold, and Christian Reuter. "A survey on data augmentation for text classification". In: *ACM Computing Surveys* 55.7 (2022), pp. 1–39. DOI: https://doi.org/10.1145/3544558.

[126] Steve Tingiris and Bret Kinsella. *Exploring GPT-3: An unofficial first look at the general-purpose language processing API from OpenAI*. Packt Publishing Ltd, 2021. URL: https://ieeexplore.ieee.org/document/10163179/authors#authors.

[127] Katikapalli Subramanyam Kalyan. "A survey of GPT-3 family large language models including ChatGPT and GPT-4". In: *Natural Language Processing Journal* (2023), p. 100048. DOI: https://doi.org/10.1016/j.nlp.2023.100048.

[128] Som Biswas. "The Function of chat GPT in social media: According to chat GPT". In: *Available at SSRN 4405389* (2023). DOI: http://dx.doi.org/10.2139/ssrn.4405389.

[129] Tianyi Zhang et al. "Bertscore: Evaluating text generation with bert". In: *arXiv preprint arXiv:1904.09675* (2019). DOI: https://doi.org/10.48550/arXiv.1904.09675.

[130] Chaobo Zhang, Jie Lu, and Yang Zhao. "Generative pre-trained transformers (GPT)-based automated data mining for building energy management: Advantages, limitations and the future". In: *Energy and Built Environment* 5.1 (2024), pp. 143–169. DOI: https://doi.org/10.1016/j.enbenv.2023.06.005.

[131] Dinesh Kalla et al. "Study and analysis of chat GPT and its impact on different fields of study". In: *International journal of innovative science and research technology* 8.3 (2023). DOI: 10.5281/zenodo.7767675.

[132] Asep Ridwan Lubis. "Balancing the Equation: Investigating AI Advantages, Challenges, and Ethical Considerations in the Context of GPT-3, Natural Language Processing, and Researcher Roles". In: *SAR Journal-Science and Research* 6.4 (2023), pp. 257–262. DOI: 10.18421/SAR64-05.

[133] Augustin Lecler, Loic Duron, and Philippe Soyer. "Revolutionizing radiology with GPT-based models: current applications, future possibilities and limitations of ChatGPT". In: *Diagnostic and Interventional Imaging* 104.6 (2023), pp. 269–274. DOI: https://doi.org/10.1016/j.diii.2023.02.003.

[134] Nimit Jagdishbhai and Krishna Yatin Thakkar. "Exploring the Capabilities and Limitations of GPT and ChatGPT in Natural Language Processing". In: *J. Manag. Res. Anal* 10.1 (2023), pp. 18–20. DOI: 10.18231/j.jmra.2023.004.

[135] Shangying Hua, Shuangci Jin, and Shengyi Jiang. "The limitations and ethical considerations of chatgpt". In: *Data intelligence* 6.1 (2024), pp. 201–239. DOI: https://doi.org/10.1162/dint_a_00243.

[136] Dah-Chin Luor. "A comparative assessment of data standardization on support vector machine for classification problems". In: *Intelligent Data Analysis* 19.3 (2015), pp. 529–546. DOI: 10.3233/IDA-150730.

[137] Murali Shanker, Michael Y Hu, and Ming S Hung. "Effect of data standardization on neural network training". In: *Omega* 24.4 (1996), pp. 385–397. DOI: https://doi.org/10.1016/0305-0483(96)00010-2.

[138] Alexander E Curtis et al. "The mystery of the Z-score". In: *Aorta* 4.04 (2016), pp. 124–130. DOI: 10.12945/j.aorta.2016.16.014.

[139] JiGuan G Lin. "On min-norm and min-max methods of multi-objective optimization". In: *Mathematical programming* 103.1 (2005), pp. 1–33. DOI: https://doi.org/10.1007/s10107-003-0462-y.

[140] Tianyang Lin et al. "A survey of transformers". In: *AI open* 3 (2022), pp. 111–132. DOI: https://doi.org/10.1016/j.aiopen.2022.10.001.

[141] Hongming Xu et al. "Vision transformers for computational histopathology". In: *IEEE Reviews in Biomedical Engineering* (2023). DOI: 10.1109/RBME.2023.3297604.