Improving Classification Performance under Imbalanced Data Conditions using Generative Adversarial Networks

Thesis approved by
the Department of Computer Science
Technische Universität Kaiserslautern
for the award of the Doctoral Degree
Doctor of Engineering (Dr.-Ing.)

to

Sungho Suh

Date of Defense:	October 21st, 2021
Dean:	Prof. Dr. Jens Schmitt
Reviewer:	Prof. Dr. Paul Lukowicz
Reviewer:	Prof. Dr. Bernhard Sick

D386

Abstract

Deep learning has achieved significant improvements in a variety of tasks in computer vision applications with an open image dataset which has a large amount of data. However, the acquisition of a large number of the dataset is a challenge in real-world applications, especially if they are new eras for deep learning. Furthermore, the distribution of class in the dataset is often imbalanced. The data imbalance problem is frequently bottlenecks of the neural network performance in classification. Recently, the potential of generative adversarial networks (GAN) as a data augmentation method on minority data has been studied.

This dissertation investigates using GAN and transfer learning to improve the performance of the classification under imbalanced data conditions. We first propose a classification enhancement generative adversarial networks (CEGAN) to enhance the quality of generated synthetic minority data and more importantly, to improve the prediction accuracy in data imbalanced condition. Our experiments show that approximating the real data distribution using CEGAN improves the classification performance significantly in data imbalanced conditions compared with various standard data augmentation methods.

To further improve the performance of the classification, we propose a novel supervised discriminative feature generation method (DFG) for minority class dataset. DFG is based on the modified structure of Generative Adversarial Network consisting of four independent networks: generator, discriminator, feature extractor, and classifier. To augment the selected discriminative features of minority class data by adopting attention mechanism, the generator for class-imbalanced target task is trained while feature extractor and classifier are regularized with the pre-trained ones from large source data. The experimental results show that the generator of DFG enhances the augmentation of label-preserved and diverse features, and classification results are significantly improved on the target task.

In this thesis, these proposals are deployed to bearing fault detection and diagnosis of induction motor and shipping label recognition and validation for logistics. The experimental results for bearing fault detection and diagnosis conclude that the proposed GAN-based framework has good performance on the imbalanced fault diagnosis of rotating machinery. The experimental results for shipping label recognition and validation also show that the proposed method achieves better performance than many classical and state-of-the-art algorithms.

Acknowledgements

I am deeply grateful to my advisor, Prof. Paul Lukowicz for supervising my thesis and giving me the opportunity of working at the German Research Center for Artificial Intelligence (DFKI). He always gives me insightful comments and motivation.

I would also like to thank Dr. Yong Oh Lee for supervising me at KIST Europe and providing essential support and advice. I have learned so much from Dr. Yong Oh Lee. I've learned how to write papers and how to proceed with research from him.

I would like to give a warm thank towards the members of my PhD committee: Prof. Deßloch and Prof. Sick. I am grateful to Dr. Jongwoon Hwang for giving me a chance to start working at KIST Europe and pursuing PhD degree in Germany. I am thankful to all other co-authors for working together: Dr. Jun Jo, Haebom Lee, Prof. Mayank Shekhar Jha, Joel Jang, Jihun Kim, Sojung Cheon, and Seungjae Won. I wish to express my thankfulness towards all the colleagues at the Embedded Intelligence of DFKI: Jane Bensch, Bo Zhou, Hymalai Bello, Vitor Fortes Rey, Sizhen Bian, Juan Felipe Vargas Colorado, Matthias Tschöpe, Agnes Grünerbl, Hamraz Javaheri, and Marco Hirsch.

Thanks to my friends in Germany who were always around with their support: Sunmi, Peter, Juna, Hana, Hyo Seon, Jae Hyung, Sia, Jae Sang, Friederike, Changgyun, Hyunki, Woojin, Younghun, Minji. Also, thanks to my friends in Korea for being my best friends and support: Junhyuk, Jeongmin, Jungkyu, Hyekyung, Younghwan, Sungho, Jungeun, Hyeoksung, Sunhwa, Minsik.

Most importantly, thanks to my family. To my Mom and Dad, I could not have done this far without their support and they have taught me how to move forward while facing obstacles in my life. It has been really happy and proud to live as your son. To my brother, Sungmin, thanks for always being a supportive brother and staying with our parents. To my mother-in-law and brother-in-law, thanks for supporting us to come to Germany and during my PhD course in Germany. To my cousin, Wonhee, and his girlfriend, Eva, thanks for always supporting us and loving our kids. To my aunt's family in Paris, thanks for being a great supporter of my family in Europe.

Especially to my wife, Jieun, for accompanying me together in my life. I could not have done without your support and sacrifice. Thank you very much and love you. To my sons, Jun and Yule, you guys are my treasures. Thank you for coming into my life.

Contents

Lists

Figures

Tables

1 Introduction

1.1 Motivation

Deep learning has achieved significant improvements in many machine learning and computer vision applications, such as object detection, image classification, medical diagnosis, autonomous vehicles, image restoration, and image segmentation. It is also widely applied in natural language processing and speech recognition where they outperform conventional machine learning models [2]. Convolutional neural networks (CNN) which is a representative deep learning model, is composed of a number of convolution, pooling, and activation layers and fully connected layers. The complex architecture of CNN requires significant computational power for training the networks and plenty of data. In other words, the significant improvement by deep learning technology has been achieved with the power of data, using a large number of images, instances, and examples to train to fit complex and huge models. However, the acquisition of large datasets is a challenge in real-world applications, especially if they pertain to new areas in deep learning. For training deep neural networks, the quantity of data is as important as the quality because training with only a small dataset often results in degradation of the neural network performance or introduces an over-fitting problem. Attempts have been made to establish a large training dataset in various fields; however, the annotation of a dataset still remains expensive, laborious, and time-consuming. Learning from small datasets is still a challenge and becomes a more difficult problem when it is augmented with other challenging problems, e.g., data imbalance, variations within the class, and similarities between classes among others. Especially, the distribution of classes in the dataset is often imbalanced. Training a network under class-imbalanced conditions can produce a detrimental effect on neural network performance and a biased classification result [3, 4, 5]. The performance of machine learning algorithms, such as CNN, deteriorates under imbalanced data conditions as the classification results show a bias in the majority data class.

The data imbalance problem has occurred in a wide range of applications. Most of them are classification and recognition problems, where the minority classes are the interested target to recognize or classify from the other classes. In this dissertation, the proposals are deployed to fault detection and diagnosis for rotating machinery, and shipping label recognition and validation for logistics, as real industrial applications.

1.1.1 Application: Fault Detection and Diagnosis for Rotating Machinery

Fault detection and diagnosis (FDD) in manufacturing facilities is very important for (1) improving productivity by preventing undesired downtime and (2) guaranteeing safe working conditions [6]. Bearing is widely used in various mechatronic systems, such as induction motors, turbines, aircraft machines, and vehicles. More than 50 % of the failures of induction motors are due to the degradation of the bearings [7]. Thus, it is important to predict future health conditions and to warn of failure to prevent catastrophic accidents by the corresponding maintenance. Prognostics and health management (PHM) technology collects status information from industrial systems, such as manufacturing machines, facilities, and power plants, to detect failures of the system and enable maintenance schedule in advance by predicting the point of failure through analysis and predictive verification [8]. One of the PHM technologies in the rolling element bearings is to predict the remaining useful life (RUL) to prevent unexpected failures and improve reliability [9, 10]. A large number of studies have been developed for RUL prediction, which can be categorized into model-based [11] and data-driven methods [12, 13].

Model-based methods detect the bearing faults and estimate the RUL through analytical models based on physical laws and mathematical functions [14, 15, 16]. These methods include physics-based methods, empirical-based methods [8], Kalman filter [17], and particle filter [18]. However, they require expert domain knowledge and it is difficult to build a precise model in the industrial systems becoming increasingly complex [19]. In addition, because physical models are highly dependent on individual specifications, user configuration is needed when the model is used in a specific facility.

To overcome the problems of physical models, assorted data-driven FDD methods that use machine learning and statistics, such as support vector machine [20, 21] and fuzzy logic [22], have been proposed. However, these conventional data-driven FDDs still require a complicated pre-processing step before training the models. Recently, data-driven methods have become more and more attractive with the significant development of machine learning and little requirement of expert knowledge. the data-driven methods capture the direct relationship between collected machine data and degradation status with machine learning techniques. In [23, 24, 25, 26, 27], time-domain, and frequency-domain features are extracted in data processing, and then an FDD model is applied for motor status classification. In [28, 29], vibration image generation with signal analysis was utilized for feature extractions. The feature extraction method itself can be automated using these approaches, but establishing such models requires complicated signal processing steps. Automatic feature extraction using an auto-encoder has been proposed [30], but the computation cost of the DNN model itself is quite high. To realize the wide adoption of data-driven FDD to industry, simpler and more efficient methods are required in both data-processing and DNN models. Furthermore, these data-driven methods have shown great performance with low domain knowledge requirements, but problems related

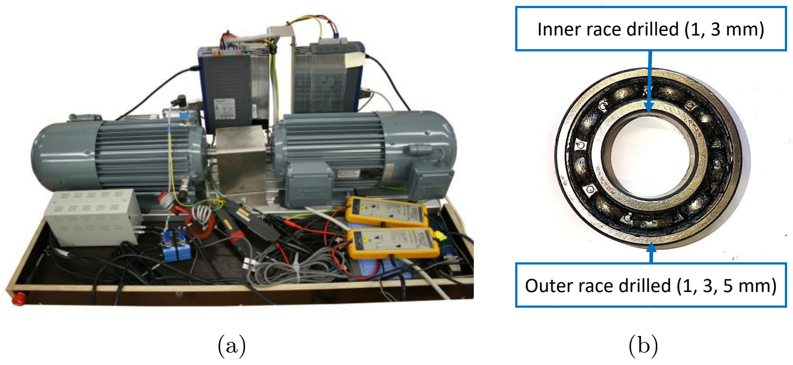


Figure 1.1: Test bench and bearings. (a) test bench (model: EMOD FKFIE2100LA, Pmech: 3 kW, nnom: 1440 /min, Vnom: 400V, Inom: 6.4A, cosϕ: 0.78), (b) bearings with different faulty conditions

to the quantity and quality of data still remain.

Data imbalance is common in FDD because normal condition data are more common than faulty condition data in real manufacturing environments [31]. Such imbalanced conditions degrade data-driven FDD, especially for the convolutional neural network (CNN)-based classifiers. Among oversampling [24, 32], down-sampling [33], and ensemble learning [34], all of which have been proposed to solve the data imbalance issue, oversampling is most suitable for industrial FDD because of severe data imbalance ratios.

In this study, the data-driven FDD of bearing faults in an induction motor under data imbalance conditions is investigated. The purpose of the fault detection method is to detect bearing faults and diagnose the fault types. Among the various fault types, inner race, outer race, and contaminant faults are considered. As an initial step, we collected two channels (the horizontal and vertical axes) of vibration signals for normal and faulty conditions using the apparatus shown in Figure 1.1 (a). To measure faulty condition data, bearings were artificially damaged by the following methods. For inner and outer race faults, we drilled into the middle of the inner and outer raceway in the bearings after removing the metal shield and the grease in the bearing. The drilling diameter was 1, 3, and 5 mm for low, medium, and high severity, respectively. For the contaminant, we inserted metal chips in the cage. The bearings with different faulty conditions are shown in Figure 1.1 (b).

1.1.2 Application: Shipping Label Recognition and Validation for Logistics

Shipping labels are widely used in logistics. A common shipping label contains barcodes and the addresses of the sender and receiver which are important to identify a package and its destination. However, with the rapid rise of the logistics delivery market, the annual cost associated with failed delivery has also increased. In 2015, this annual cost was estimated to be approximately 1.5 billion dollars for the postal service and 20 billion dollars for the mailing industry in the United States of America [35]. One of the main causes of misdelivery is incorrect shipping labels, which may contain falsely given addresses or may be damaged during the logistics process. Packages with problematic labels lead to not only cost loss during unnecessary delivery to undeliverables and return but also damage to the manufacturer's and seller's reputations. A process for screening undeliverable addresses and error-containing barcodes in the shipping label that is used for identifying packages and their destinations is required. Recently, an automatic inspection system with high speed and accuracy to recognize barcodes and verify the addresses is highly desirable to the logistics and manufacturing industries.

To verify the information in the shipping label, the commonly used approaches are optical character verification (OCV) and optical character recognition (OCR). These approaches capture an image using a vision system or smartphone, detect regions of interest (ROIs), and recognize optical characters in the ROIs. Several traditional image processing methodologies have been applied for OCV and OCR, but deep-learning-based approaches have been shown to be effective for these tasks [36, 37, 38, 39, 40]. However, unlike the food package application, the quality of the barcode also has to be inspected for the shipping label. The printed address not only has to be compared to the user-typed address, but its validity also has to be verified. Like OCV, there exist several barcode quality and printed label inspection systems, but they are mostly off-the-shelf and black-box commercial ones [41] and the systems usually cost high and support only a limited number of shipping label formats. Though the address verification is available in [42], their service is to verify the delivery address requested by customers, not the address in the printed shipping label. Therefore, it could not be inspected in the process after packaging. Furthermore, the performance of OCR engines and text detection engines is sensitive to image quality and defects on target objects. The quality of the captured image can be affected by many degradations caused during image acquisition. During the packaging and delivery processes, the shipping label can be damaged. Thus, proper assessment and classification of the captured image and object are required to improve the text detection and recognition processes. Four poor conditions of captured images are defined, and example images of collected datasets are shown in Fig. 1.2.

The shipping label inspection system [43] involves six steps: input image quality verification, calibration, salient region detection, image enhancement, text recognition, and address validation. The flowchart of the shipping label inspection system is demonstrated in Fig. 1.3.

In the first step, the proposed input image quality verification process [44] classifies the five image types: normal, contaminated address, unreadable image, handwritten address, and damaged shipping label. The unreadable image and the damaged label are reported to the user to acquire the image properly or check the condition of the shipping label. The contaminated and handwritten addresses provide information on the address area for image enhancement and/or text recognition processes. In the

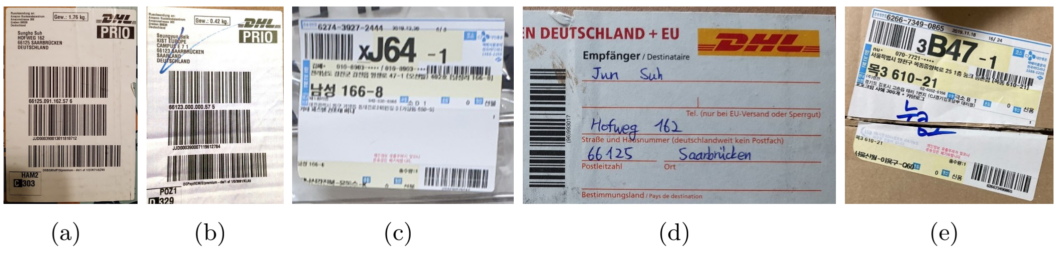


Figure 1.2: Examples of the annotated dataset of the shipping label. (a) normal, (b) contaminated, (c) unreadable, (d) handwritten, (e) damaged.

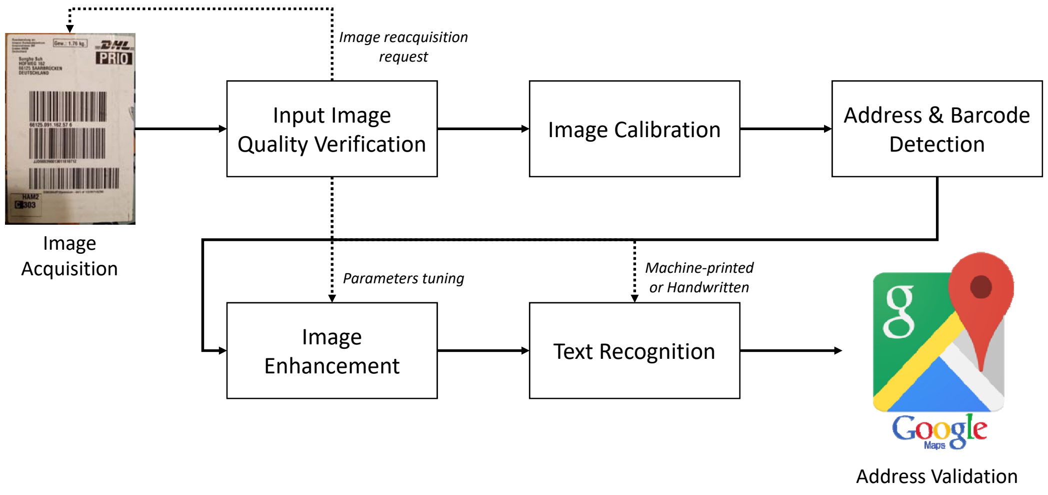


Image reacquisition request
Image Acquisition
Input Image Quality Verification
Image Calibration
Address & Barcode Detection
Parameters tuning
Machine-printed or Handwritten
Image Enhancement
Text Recognition
Address Validation

Figure 1.3: Flowchart of the shipping label inspection system

collected dataset, the number of normal images is by far more than other types and it is severely imbalanced. In the second step, the shipping label images have to be enhanced to improve the performance of text recognition. However, for a shipping label image having a large background region portion, prediction can sometimes misclassify parts of the background region as foreground. In the final step, the address in the shipping label needs to be recognized and validated.

1.2 Data Imbalance Problem

The data imbalance problem is frequently the bottleneck of the neural network performance in classification. The data imbalance is a terminology used to describe the situation where the class distributions are not equal, i.e., when one class, which calls majority class, exceeds by far the other classes, which calls minority classes. It is the case when different classes have a significantly different number of samples. Due to the data imbalance problem, the training algorithm gives more attention to the class(es) with the majority of samples, which results in biased classifiers, i.e., classifiers that give more attention to samples belonging to the majority classes. If the empirical training and test distributions are the same as the true data distribution, it is in the right direction that the training process minimizes errors on the training set. In this case, Minimizing errors on the training dataset improves the accuracy of the test dataset. Data imbalance is not an issue in the three cases as follows. First, classes are easy to be separated under the data imbalance condition. Second, the number of available samples is quite large enough to cover all aspects of the true data distribution, in which case undersampling can be used to balance the data distribution and improve the performance of the classification. Last, the variation within the majority class is very low, which means that most of the samples in the majority class are distributed within a certain space. Thus, the data imbalance can be an issue only if all classes are essential or the minority classes are more important. However, the data imbalance problem is easily unveiled in numerous fields including computer vision [45, 46, 47, 48, 49], medical diagnosis [50, 51, 52], fault detection [31, 53], and others [54, 55, 56].

Figure 1.4 provides an example of classification under four different data imbalance conditions expressed with four different colors. The data samples by class were marked with circular points of four different colors, and the classification results were expressed in the background colors of different areas. Case 1 falls under the balanced data condition and a trained classifier predicts the four classes well. However, as imbalanced conditions worsen from case 2 to case 4, the classifier tends to classify samples of the minority classes as instances of the majority class. In particular, as the number of samples in classes C and D decreases from case 2 to case 4, the classification rates for classes C and D decline considerably. In other words, the classifier trained with imbalanced data tends to predict a class belonging to the majority ones. Thus, it ignores the effect of the minority or blends it with the majority.

1.3 Overview of State-of-the-art Methods

Among the main approaches to deal with the data imbalance problem, the rebalancing of the class distribution at the data level, such as oversampling [24, 53, 32], undersampling [33], and ensemble learning [34], is a general solution without the dependency of the classifier. Between them, the oversampling technique, which results in the generation of artificial data for the minority class, has proven to be the most effective way to handle the class imbalance for the CNN model on image classification [5].

In this dissertation, the data imbalance ratio (IR) and the adjusted IR (A-IR) are frequently used.

- IR = (# of majority class data) : (# of minority class data),

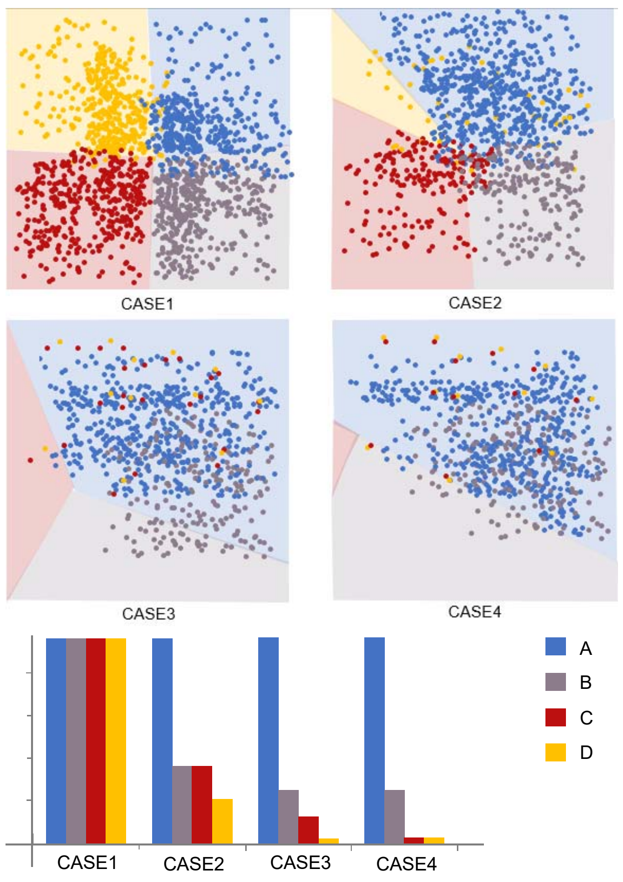


Figure 1.4: The example of the effect of the imbalanced data conditions for the classifier

- Adjusted IR (A-IR) = (# of majority class data) : (# of minority class data) + (# of augmented data of minority class)

A traditional oversampling method in the computer vision domain to augment the training dataset and reduce overfitting consists of geometric transformations such as rotation, image cropping, flipping, and color conversions [57, 58]. However, the images generated with these methods are merely simple and redundant copies of the original data in many cases. Additionally, the geometric transformations do not improve the data distribution determined by the high-level features as they only lead to an image-level transformation through depth and scale [57]. Thus, the oversampling technique is needed to estimate the data distribution and generate data, not just to augment the training set.

The standard oversampling algorithm is the synthetic minority oversampling technique (SMOTE) [59], which generates new data samples for minority classes based on the similarities between the original minority class samples in the feature space. It is one of the most widely used algorithms to generate synthetic minority class samples to improve the performance of the classification [60]. However, it can produce noisy results when the appearances of the majority and minority classes are ambiguous [61]. For this reason, extensions including borderline SMOTE [62] and ADASYN [63] have emerged, attempting to increase classification accuracy by sharpening the boundary between the two classes. However, these oversampling techniques can fail to generate new samples that are similar to the original data at first glance but different in detail, especially when extracting features in a regularized way from the imbalanced dataset is difficult. When these oversampling techniques target high-dimensional imbalanced data, such as images and audio, they cannot reduce the classification bias towards the majority class [64]. Moreover, the Euclidean distance used in SMOTE is not a suitable metric to measure the similarity between samples in high dimensional spaces [65]. In certain cases, the Euclidean distance between the target sample and the nearest neighbor is larger than the distance between the sample and the furthermost neighbor.

Recently, generative adversarial networks (GAN) [66] have emerged as a class of generative models approximating the real data distribution. Conditional GAN (CGAN) [67] and auxiliary classifier GAN (ACGAN) [68] also extend GANs by conditioning the training procedure on the class labels for the classifier. Douzas et al. [69] demonstrate that the data generated by CGAN improves the performance of the classification.

However, GAN and conditional GANs, such as CGAN and ACGAN, have limitations. The instability of the training process remains these GANs as a challenge in practice. The other problem is the influence of noise. As the generator of the GANs takes as input a random noise vector and outputs a synthetic image, we need the classifier for the conditional GANs to consider the influence of noise. Also, conditional GANs perform well in the hypothesis that the class boundaries are clear. In a real-world dataset, this hypothesis does not hold because the boundaries between classes are often unclear and ambiguous. In this dissertation, a classification enhancement generative adversarial networks (CEGAN) [70] is proposed to generate synthetic minority data to enhance the classification under the data imbalanced conditions. The proposed method deploys the objective formulation of Wasserstein generative adversarial network with gradient penalty (WGAN-GP) [71], which improves the stability and performance of the classification under imbalanced data conditions [53, 72].

Furthermore, one of the general approaches to improve neural network performance under small and biased datasets is transfer learning. The weights of deep neural networks are firstly pre-trained on a large-scale dataset, which is called source task, and then fine-tuned using the data from the target task with a small number of data [73]. In the case of fine-tuning in CNN, the weights in the first few convolution layers are fixed by pre-train the last convolution layers are fine-tuned by the target task. During fine-tuning, the parameters of the target model can be driven far away from the pre-trained parameter values, and it leads to incorporating information relevant for the targeted problem, and overfitting to the target task, so-called catastrophic forgetting [74].

During the past few years, several transfer learning approaches with regularizations have been proposed to constrain parameters on preserving pre-trained knowledge on the source dataset. Li et al. [75] proposed L^2-SP that weight parameters are constrained to be driven to the pre-trained weight parameters by minimizing the distance of weights between source and target networks. Li et al. [1] proposed DELTA, which utilizes discriminative knowledge in feature maps from outer layers outputs through re-weighting the feature maps by the attention mechanism. These regularization approaches achieved significant improvement and alleviate the catastrophic forgetting problem by drawing weight parameters close to pre-trained values or aligning transferable channels in feature maps. To further improve the performance of the classification with a small training dataset or class imbalanced dataset, a novel discriminative feature generation (DFG) method [76] using attention maps in the feature space is proposed in this dissertation. The proposed method is a combination of transfer learning and adversarial feature augmentation to complement their drawbacks. The baseline is transfer learning with regularizations, and the features are augmented using a GAN with the weight of the activation level of the feature maps in each class.

1.4 Contributions of This Thesis

Several important research issues in imbalanced data learning are investigated in this dissertation. The data imbalance problem is common in a wide range of applications and GAN can be utilized in many various applications. The proposals have been deployed to fault detection and diagnosis for rotating machinery [53, 77, 78], shipping label recognition [43, 44, 79], medical CT image synthesis and segmentation [80, 81], and text classification [82]. The representative two applications are selected from the publications in the dissertation, and the already published contributions of this work are summarized as follows.

- **Classification Enhancement Generative Adversarial Networks (CEGAN):** The CEGAN is proposed to enhance the quality of generated synthetic minority data and to improve the performance of the classification under the data imbalanced condition. A novel GAN structure containing a classifier, that induces the generated data, has more features for classification. The classifier has the functionality of reducing the impact of noise input and the ambiguity between classes. This study is demonstrated with five benchmark datasets. The results indicate that approximating the real data distribution using CEGAN improves the classification performance significantly in data imbalanced conditions compared with various standard data augmentation methods. The CEGAN method was published in the journal *Neural Networks* [70].

- **Discriminative Feature Generation (DFG):** The DFG method is proposed using attention maps in the feature space. This method is a combination of transfer learning and adversarial feature augmentation to complement their drawbacks. The main contribution is a novel feature generation and augmentation using the proposed GAN structure to unravel the data imbalanced problem and improve neural network performance. For the generation of meaningful features for the classification of small-sized target data, transfer learning with regularization and class-wise attention is adopted. The proposed method is evaluated with various pairs of source and target datasets to show general applicability to classification in the class-imbalanced conditions. The DFG method was published in the journal *Pattern Recognition* [76].

- **Generative Oversampling Method on Bearing Fault Detection and Diagnosis (FDD):** A novel data-driven FDD method for bearing faults in induction motors where the fault condition data are imbalanced is proposed. The bearing fault detection method is based on CNN, in which the vibration signals from a test bench are used as inputs after an image transformation procedure. Furthermore, a novel data-driven health state division method based on CNN and a generalized feature extraction method for remaining useful life prediction of bearing based on GAN are also presented in this dissertation. The CNN-based FDD model and the health state division method were published in the journal *Applied Sciences* [53] and the journal *Sensors* [77], respectively. In addition, the generalized multiscale feature extraction method is available online [78].

- **Robust Shipping Label Recognition and Validation for Logistics:** a verification and recognition system for various types of shipping labels by using deep neural networks was developed. To effectively detect barcode and address areas, a popular deep learning-based object detection algorithm, You Only Look Once (YOLO) [83], is adopted for the shipping label dataset and the angle of the image is calibrated by the angle estimated from the barcode. To inspect the quality of an input image, an input image quality verification method combining global and local features is also presented in this study. Finally, a two-stage color document image enhancement and binarization method using GAN is proposed to improve the accuracy and efficiency of text recognition in the shipping label. The verification and recognition system of shipping labels is published in *ICIP 2019* [43] and the input image quality verification method is published in *ICPR 2020* [44]. The color document image binarization method is available online [79].

1.5 Thesis structure

This dissertation is organized as follows. Chapter 2 explains the background techniques related to imbalanced data classification, GAN, and transfer learning. The advantages and limitations of related work and state-of-the-art research are summarized and discussed. Chapter 3 introduces the CEGAN method. The experimental results on five benchmark datasets indicate that approximating the real data distribution using CEGAN improves the classification performance significantly in data imbalanced conditions compared with various standard data augmentation methods. Chapter 4 presents the discriminative feature generation method for the classification of imbalanced data. In Chapter 5, the generative oversampling method for imbalanced data on bearing FDD is presented and the evaluation shows that the GAN-based method improves the performance of the FDD and predicts the remaining useful life well. Chapter 6 shows another application in shipping label recognition and validation. It shows the CNN-based image quality inspection method and the GAN-based document image binarization method. Finally, Chapter 7 summarizes the achievements and contributions and proposes future research directions.

2 Background and Related Work

2.1 Synthetic Oversampling Methods

The oversampling method adds the minority class samples to an imbalanced training set. The traditional oversampling methods can be categorized into random oversampling and synthetic oversampling methods. Random oversampling is a non-heuristic method that adds samples through the random replication of the minority class samples [84, 85]. In the computer vision domain, the random oversampling consists of geometric transformations such as rotation, image cropping, flipping, and color conversions [57, 58] to augment the training dataset and reduce overfitting. However, the random oversampling methods create very specific rules, leading to oversampling [86]. Additionally, the geometric transformations do not improve the data distribution determined by the high-level features as they only lead to an image-level transformation through depth and scale [57]. Thus, the random oversampling technique is needed to estimate the data distribution and generate data, not just to augment the training set.

As an alternative to the random oversampling method, the synthetic oversampling methods add new samples to balance the training dataset by generating synthetic minority class samples. The generated samples add essential information to the original data set that may help to improve the performance of the classification. A standard synthetic oversampling method is the synthetic minority oversampling technique (SMOTE) [59], which aims to create synthetic examples based on original samples according to the similarities among existing minority instances. It is one of the most widely used algorithms to generate synthetic minority class samples to improve the performance of the classification [60]. The procedure of SMOTE is as follows. The first step is to select example samples from a minority class in the training dataset. Then, SMOTE finds the k nearest neighbors of the selected samples. Lastly, one of the k instances is randomly chosen to generate a synthetic sample by interpolation, which calculates the euclidean distance between the sample the selected nearest neighbor. The difference is then multiplied by a random number between 0 and 1. SMOTE can be expressed as follows. Given two samples from a minority class, such as $x_i, x_j \in R^d$, new samples are generated.

$$x_{new} = x_i + r(x_j - x_i), \text{where} \quad r \in [0, 1] \tag{2.1}$$

where x_{new} is the generated sample, and r is a random number ranging between 0 and 1. Compared with the random oversampling method, SMOTE causes the decision boundaries for the minority class to be spread further into the majority class space, prevents the overfitting problem to a certain degree, and improves the performance of the classification. However, it can produce noisy results when the appearances of the majority and minority classes are ambiguous [61].

For this reason, many extensions to the SMOTE algorithm have been developed. borderline SMOTE [62] generates synthetic samples along the borderline between minority and majority classes for better classification results. The borderline SMOTE assumes that the samples close to the borderline are more significant for classification. The first step of the borderline SMOTE is to find the k nearest neighbors of every sample in the entire training set, not only from a minority class. For every sample x_i, let k' be the number of majority samples in K nearest neighbors of x. If $k' = k$, all k nearest neighbors are majority samples, which means this sample can be regarded as noise. If $k/2 \leq k' < k$, the number of majority nearest neighbors is larger than the number of the minority one, so x_i can be easily misclassified and put into a set $DANGER$. The algorithm finds k nearest neighbors from the minority class in the $DANGER$ set and generates s new synthetic minority samples between x_i and its nearest neighbors.

$$y_j = x_i + r_j(\tilde{x}_j - x_i), j = 1, 2, ..., s \tag{2.2}$$

where y_j is the generated sample, $\tilde{x}_j$ is j-th nearest neighbor, and r_j is a random number between 0 and 1. Similarly, safe level SMOTE [87] accounts for the nearest neighbors for both the minority and majority classes and defines the number of positive instances in k nearest neighbors. If the safe level of an instance is close to zero, the sample is considered as noise. Only samples with sufficient safe levels are used to generate more samples. Each synthetic instance is generated in a safe position with the safe level ratio of instances.

SMOTE using support vector machine algorithm (SVM-SMOTE) [88] is proposed using an alternative of borderline SMOTE where an SVM algorithm is used instead of a k nearest neighbor to establish the decision boundary between classes. The Majority Weighted Minority Oversampling Technique (MWMOTE) [89] is a complex oversampling algorithm to handle overfitting problems. MWMOTE comprises three stages to make synthetic data: identification of the hard-to-learn minority class samples on datasets, importance weighting for each hard-to-learn minority class sample, and synthesis of the new samples following a strategy similar to SMOTE. The results of the MWMOTE can reduce the degree of bias or noise and produce synthetic data with better accuracy.

Another SMOTE extension ADASYN [63] is proposed to balance the skewed data distribution by generating data samples adaptively based on the data distribution and using k nearest neighbors. ADASYN uses the data distribution to decide about the number of synthetic samples to be generated for each minority sample by adaptively changing the weights of the different minority samples to compensate for the skewed distributions, whereas SMOTE generates the same number of samples for each minority sample. This inspired other similar techniques to incorporate mechanisms that control the number of synthetic data [90, 91].

However, most of the synthetic oversampling methods were designed for the imbalanced datasets in two-class, the majority, and minority classes. Additionally, the synthetic oversampling methods seldom showed their superiority when the target data contains more complex or high-dimensional features. In the other words, when these oversampling techniques target high-dimensional imbalanced data, such as images and audio, they cannot reduce the classification bias towards the majority class. The synthetic oversampling methods cannot reduce the classification bias towards the majority class for the classifier and are even less effective than the random undersampling method, dealing with complex high-dimensional imbalanced data [64].

Moreover, the Euclidean distance used in SMOTE is not a suitable metric to measure the similarity between samples in high dimensional spaces [65]. In certain cases, the Euclidean distance between the target data and the nearest neighbor is larger than the distance between the data and the furthermost data.

Furthermore, the introduced synthetic oversampling methods can fail to generate new samples that are similar to the original data at first glance but different in detail, especially when extracting features in a regularized way from the imbalanced dataset is complex.

2.2 Data Augmentation using GAN

2.2.1 Generative Adversarial Networks

Vanilla Generative Adversarial Networks

Generative adversarial networks (GAN) [66] represent a class of generative models based on a game theory scenario in which a generator network G competes against an adversary, D, a discriminator. GAN aims to approximate the probability distribution function that certain data is assumed to be drawn from. The objective function of the min-max game between the generator and the discriminator is expressed as follows:

$$\min_G \max_D E_{x \sim P_r(x)}[\log D(x)] + E_{z \sim P_z(z)}[\log(1 - D(G(z)))], \tag{2.3}$$

where x is real data sampled from the real data distribution $P_r(x)$, z is the noise vector sampled from a uniform distribution $P_z(z)$, and the generator G generates a synthetic image $G(z)$. The equation shows that treating the discriminator as a classifier minimizes the Jensen-Shannon (JS) divergence between the real data distribution and the one assumed by the generator.

In other words, in each iteration of the training process, the discriminator is trained to fit to maximize the JS divergence to distinguish real and synthetic fake data while the generator is given. On the other hand, the generator is trained to fit to minimize the JS divergence to generate fake data looks real when the discriminator is fixed. Thus the training of GAN is a min-max game between the generator and the discriminator. However, in practice, the training process of GANs is extremely unstable and struggles to have the problem of mode collapse, where the generator generates simple modes of images, which cannot be distinguished by the discriminator since the generated image looks real. Thus, it is difficult to train in practice and it cannot improve the performance of the classification vastly. For this reason, much of the recent work has been focused on improving the stability and promising diversity of generated image mode.

Conditional Generative Adversarial Networks

The conventional vanilla GAN is an unsupervised model. Conditional GAN (CGAN) [67] is the first conditional variant of GAN to control the generated data samples, which applies the generated antagonistic network concept to the supervised learning method and achieve a corresponding effect between labels and the generated data samples. CGAN takes conditional label inputs for the generator and the discriminator to train the networks under the conditions. The objective functions of conditional GAN are as follows:

$$\min_G \max_D E_{x\sim P_r(x)}[\log D(x|y)] + E_{z\sim P_z(z)}[\log(1 - D(G(z|y)))], \tag{2.4}$$

where y is the corresponding class labels of input data x.

Isola et al. [92] proposed Pix2Pix GAN for the general purpose of image-to-image translation using CGAN. The generator of the Pix2Pix GAN model is trained via adversarial loss, which encourages the generation of plausible images in the target domain and minimizes the measured L1 loss between the generated image and the expected output image. The discriminator is provided with both the source image and a target image and determines whether the target image is a plausible transformation of the source image. Pix2Pix GAN was demonstrated on a range of image-to-image translation tasks such as conversions of maps to satellite photographs, black-and-white photographs to color, and product sketches to product photographs.

Auxiliary classifier GAN (ACGAN)[68] is a class-conditional extension of GANs and offers a simple method for providing varying amounts of control in the image generation process by adding a classification layer to the discriminator. The objective function consists of two parts: source and classifier loss. The source loss with class labels is identical to the objective function in GANs, Eq. (2.3), generating synthetic images indistinguishable from the real ones and distinguishing real and synthetic data. The source loss is expressed as follows:

$$L_{src} = E_{x\sim P_r(x)}[\log D(x)] + E_{z\sim P_z(z), c_g\sim P(c_g)}[\log(1 - D(G(z, c_g)))], \tag{2.5}$$

where c_g is the sampled class label from the conditional data vector found in the training dataset and represented by the categorical distribution $P(c_g)$.

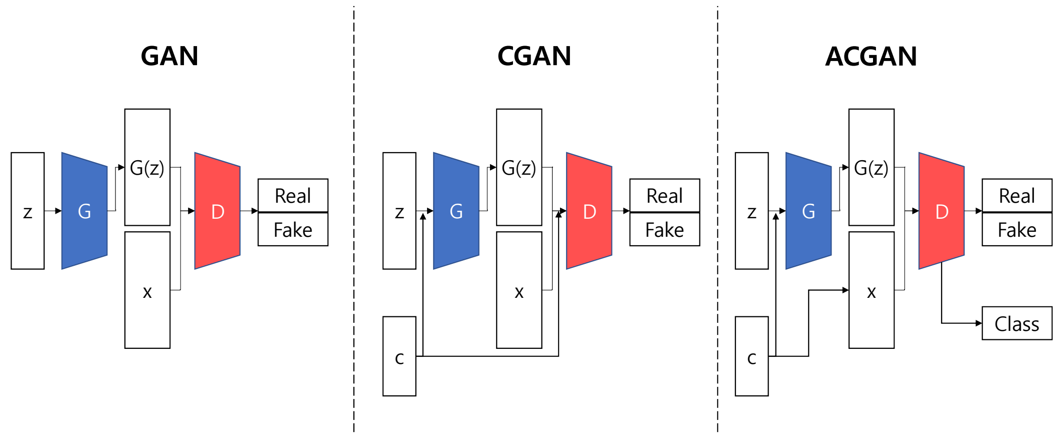


Figure 2.1: Overall framework comparison of ACGAN to a vanilla GAN and CGAN

The classifier loss is employed to ensure that the images are classified into the right categories and at the same time forces the generator to produce synthetic data belonging to the target class.

The classifier loss is defined as

$$\begin{aligned} L^r_{cls} &= E_{x \sim P_r(x), c_r \sim P(c_r)}[-\log C(c = c_r | x)] \\ L^g_{cls} &= E_{z \sim P_z(z), c_g \sim P(c_g)}[-\log C(c = c_g | G(z, c_g))], \end{aligned} \tag{2.6}$$

where $C(c|x)$ represents a probability distribution over the class labels computed by an auxiliary classifier C in the discriminator, x and c_r are a pair from the real data and class labels of the training dataset. In Eq.(2.6), the auxiliary classifier C is trained to classify real data with the corresponding class label c_r and the generator G is optimized to generate synthetic images classified as the target class c_g by the auxiliary classifier C.

Finally, the full objective functions involved in training the ACGAN are expressed as follows.

$$\begin{aligned} L_D &= -L_{src} + \lambda_r L^r_{cls} + \lambda_g L^g_{cls} \\ L_G &= L_{src} + \lambda_r L^r_{cls} + \lambda_g L^g_{cls}. \end{aligned} \tag{2.7}$$

λ_r and λ_g are hyper-parameters of the classifier losses for real and synthetic data. The discriminator D is trained to minimize L_D and the generator G is trained to minimize L_G. Figure 2.1 compares the ACGAN network architecture with that of vanilla GAN and CGAN and Figure 2.2 displays the ACGAN architecture.

Nevertheless, the reliability of ACGAN still needs to be improved and the quality of the data generated is incomparable with that of the real training data, although

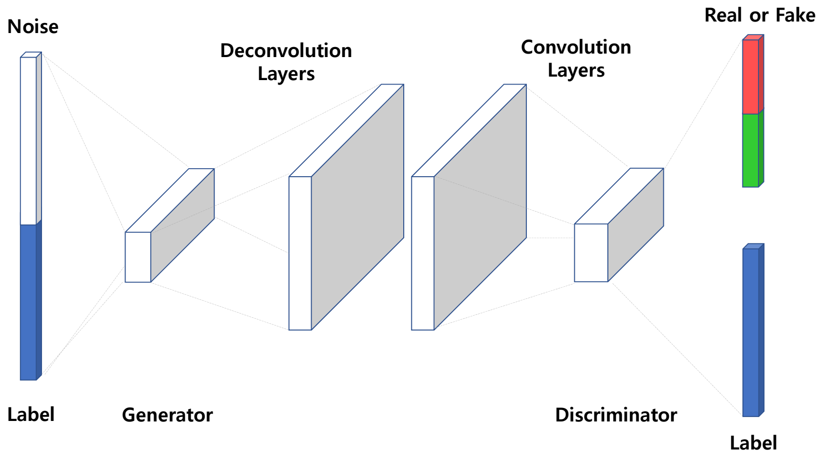


Figure 2.2: Network architecture of ACGAN

the images generated are both diverse and discriminable [68]. The main cause of the unstable training process is that the JS divergence in the GAN formation is discrete; therefore, it tends to result in a stationary discriminator and to suffer from the gradient vanishing problem during the training. Furthermore, the quality of each model trained is measured in a subjective manner as smaller losses do not guarantee better results.

Wasserstein Generative Adversarial Networks

To solve the stability problem mentioned above, the Wasserstein GAN (WGAN) [93] uses the Wasserstein-1 distance, also known as the earthmover (EM) distance, which is a continuous and more appropriate metric for measuring the distance between two distributions. The EM distance does not suffer from vanishing gradients; by contrast, the JS divergence in the GAN does not supply useful gradients to the generator [93]. The objective function of the WGAN is provided in Eq.(2.8).

$$\max_{w \in W} E_{x \sim P_r(x)}[f_w(x)] - E_{z \sim p_z(z)}[f_w(G(z))] \tag{2.8}$$

Eq.(2.8) is the EM distance between the probability distribution of the real data and one of the generated data points. W denotes a set of K-Lipschitz functions and a function f_w is modeled as the discriminator and one of the K-Lipschitz functions. To ensure the discriminator f_w belongs to the K-Lipschitz functions, the weights w for each layer of the discriminator are confined within the range $[-c, c]$ [93]. Thus, the training procedure for the WGAN first trains the weights w of the discriminator in Eq.(2.8) to yield the exact distance between two probability distributions. As a next step, the generator is trained by estimating the gradients of the objective function $-E_{P_z}(\nabla f_w G(z)))$ to minimize the distance between two distributions while the discriminator f_w is fixed. Consequently, the losses of the generator and the discriminator by the WGAN correlate well with the image quality and the WGAN provides stable updates to the generator and the discriminator, in contrast with the original GAN.

However, this method is still not optimal in terms of convergence and high-quality data generation because of the weight clipping procedure needed to satisfy the Lipschitz constraint. As a consequence of weight clipping, the WGAN cannot fit a complex distribution and the confinement of the weights to a small range in each layer can lead to gradient vanishing or exploding. To satisfy the Lipschitz constraint without clipping the weights in each layer of the discriminator, the Wasserstein GAN with gradient penalty (WGAN-GP) is proposed [71]. The loss function of the WGAN-GP is defined as follows.

$$L = E_{z \sim p_z(z)}[D(G_\theta(z))] - E_{x \sim P_r(x)}[D(x)] + \lambda E_{\hat{x} \sim P_{\hat{x}}(\hat{x})}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2], \quad (2.9)$$

where λ is the penalty coefficient and $P_{\hat{x}}$ is the uniform sampling along straight lines between pairs of points from the real data distribution P_r and the distribution generated. The motivation for this is that the constraint is enforced uniformly along the line as the optimal discriminator consists of straight lines connecting the two distributions. The WGAN-GP provides a training procedure that is faster and more stable than the WGAN.

2.2.2 Data Augmentation at Input Level

Data augmentation enhances the size and quality of the training dataset. Techniques range from simple data augmentation, such as flip, shift, and rotation, to deep generative models based on generative adversarial networks (GAN) [66], where a small training dataset is augmented at the input data level. Because GANs can approximate the distribution of the real input data and generate realistic samples from a generative model, many recent studies have shown that augmenting a small training dataset using a GAN can improve the classification performance in real applications. Huang et al. [94] proposed stacked GAN (SGAN), which is trained to invert the hierarchical representations of a bottom-up discriminative network. Guo et al. [95] proposed a discriminative variational autoencoding adversarial network, which learns a latent two-component mixture distributor and alleviates the class imbalance for deep imbalanced learning. Cui et al. [96] proposed a class-balanced loss for long-tailed distributions. The class-balanced loss re-weights losses inversely with the effective number of samples per class. However, such data augmentation at the input level showed limited improvement in the performance, because balancing the data distribution has weak relation to the enhancement of feature extraction in the minority dataset. Although the generated synthetic data can balance the distribution between the classes, the classification performance is limited due to no guarantee enhancing the feature extraction ability of the classifier, when it is an independent network of the GAN. In Chapter 3, the CEGAN method is proposed to generate synthetic data at the input level to improve the performance of the classification. Even if GAN is trained with the classifier together, GAN has the limitation to generate real-looking synthetic data on large-scale images. According to [97, 98], the state-of-art GAN has the limited generating capability on large-scale images. When the size of the input image is large, this augmentation method is not applicable. In other words, not-qualified synthetic data is not able to improve the performance of the classification significantly.

2.2.3 Feature Augmentation

Unlike GAN-based data augmentation at the input data level, Volpi et al. [99] firstly proposed to perform data augmentation using the GAN scheme in the feature space. The adversarial feature augmentation generates domain-invariant features and the size of the minority classes in the feature space without regarding the modality of the input data. A feature extractor is trained with the source dataset under supervised learning, and then a feature generator for the unlabeled target dataset is trained in the CGAN framework [67] against the feature extractor. Zhang et al. [100] developed a more general feature generation framework for imbalanced classification, inspired by the adversarial feature augmentation approach. These methods can generate domain-invariant features without considering the modality of the data. Consequently, they achieved better performance than unsupervised domain adaptation methods.

However, the improvement of the classification under the class-imbalanced condition is still not significant. The dimension of the features in the feature extractor is considerably lower than the input data. Such a low dimension may not be sufficient to present the data distribution of a small amount of minority class data. The generated features should be domain-invariant and modality-free.

2.3 Transfer Learning

Transfer learning is a machine learning method that transfers knowledge learned in a source task to a target task [101, 73]. Transfer learning is useful when the volume of the target task is much smaller than that of the source task. The weights of deep neural networks are first pre-trained on a large-scale dataset, which is called the source task, and then fine-tuned using the data from the target task with a small number of data [73]. In the case of fine-tuning in convolutional neural networks, the weights in the first few convolution layers are fixed by pre-training. The last convolution layers are fine-tuned by the target task. During fine-tuning, the parameters of the target model can be driven far away from the pre-trained parameter values, leading to the incorporation of information relevant to the targeted problem and overfitting to the target task, so-called catastrophic forgetting [74]. This simple method cannot guarantee good performance because it may in many cases burden the network of the target task with irrelevant information. Donahue et al. [102]

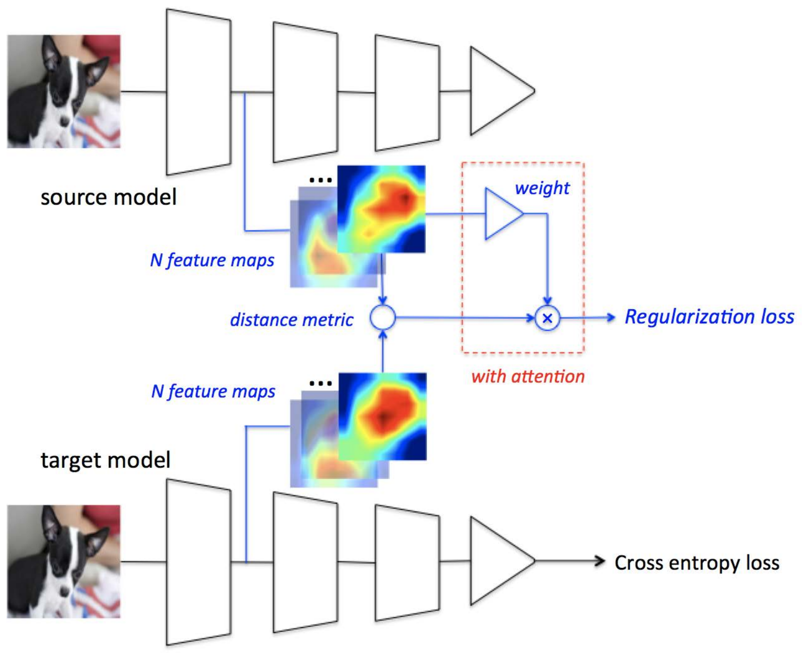


Figure 2.3: DELTA Framework [1]

employed a classifier trained with features extracted from a pre-train, and a large number of parameters of the source task are reused in the target task. This simple method cannot guarantee performance, because it may aggravate the network of the target task with irrelevant information in some cases. Yosinki et al. [103] quantified the transferability of features from each layer to learn transferable representations. For constraining catastrophic forgetting in inductive transfer learning, L^2-norm regularization was proposed in [75]. The key concept of L^2-SP is "starting point as reference" optimization, which tries to drive weight parameters to pre-trained values by regularizing the distance between the parameters of source and target tasks.

Instead of regularizing the weight parameters, deep learning transfer using a feature map with attention (DELTA), which is a regularized transfer learning framework, was proposed by Li et al. [1]. Figure 2.3 illustrates the framework of DELTA. Inspired by knowledge distillation for model compression [104, 105, 106], DELTA employs the ideas of "inactivated channel re-usage" and feature map regularization with attention. These regularization approaches achieved significant improvement and alleviated the catastrophic forgetting problem by drawing weight parameters close to pre-trained values or aligning transferable channels in feature maps. It constrains the difference between the feature maps generated by the convolution layers of the source and target networks with attention. DELTA selects the discriminative features from the outer layer outputs using a supervised attention mechanism. The overall loss functions are expressed as follows.

$$\min_{\theta_T} \sum_i L(C(x_i, \theta_T), y_i) + \Omega(\theta_T, \theta_S, x_i, y_i, C) \tag{2.10}$$

where θ_S and θ_T are weight parameter vectors of the source and target networks, (x_i, y_i) is the input tuple, $L(C(x_i, \theta_T), y_i)$ refers to the classification loss, and Ω is the regularization loss.

$$\Omega(\theta_T, \theta_S, x, y, C) = \alpha\Omega'(\theta_T, \theta_S, x, y, C) + \beta\Omega''(\theta_T \backslash \theta_S) \tag{2.11}$$

where α, β are two non-negative tuning parameters, Ω' is behavioral regularizer, Ω'' constrains the L^2-norm of the private parameters in θ_T. Behavioral regularizer is defined as follows.

$$\Omega'(\theta_T, \theta_S, x_i, y_i, C) = \sum_j (W_j(C, \theta_S, x_i, y_i) \cdot \|FM_j(C, \theta_T, x_i) - FM_j(C, \theta_S, x_i)\|_2^2 \tag{2.12}$$

where $FM_j(C, \theta_T, x_i)$, $FM_j(C, \theta_S, x_i)$ are feature maps extracted from the j-th filter and the i-th image in target and source, respectively, and the behavioral difference Ω' is measured using Euclid distance. $W_j(C, \theta_S, x_i, y_i)$ refers to the weight assigned to the j-th filter and the i-th image for a supervised attention mechanism and is defined as follows.

$$W_j(C, \theta_S, x_i, y_i) = softmax(L(C(x_i, \theta_{T\backslash j}), y_i) - L(C(x_i, \theta_S), y_i)) \tag{2.13}$$

where $\theta_{T\backslash j}$ refers to the modification of the original parameter with all elements of the j-th filter set to zero. In other words, the weight $W_j(C, \theta_S, x_i, y_i)$ is characterized by the potential performance loss as removing the features from the trained network.

3 Classification Enhancement Generative Adversarial Networks (CEGAN)

3.1 Framework of CEGAN

3.1.1 Data Augmentation using CEGAN with Noise Reduction

It is necessary to ensure the performance of the classifier in the training procedure of ACGAN because the discriminator and the generator are trained by using the classification loss. However, because the auxiliary classifier in ACGAN shares network structure and weight parameters with the discriminator, the performance of the auxiliary classifier in ACGAN cannot lead to generating high-quality images. In order to generate data for minority classes to improve the performance of the classifier and train the generative model stably, we employ the structure of the classifier to the independent network in the proposed GAN structure. We propose a classification enhancement GAN (CEGAN) which is composed of three independent networks: a discriminator, a generator, and a classifier. As indicated in Figure 3.1, the training procedure comprises of two steps: Step 1 shows the architecture of CEGAN. Unlike the framework of ACGAN in Figure 2.2, the class label is classified by an independent classifier. Step 2 involves training a classifier with the real and augmented dataset to verify the improvement of the classification performance. C_1 is the modified classifier reducing the influence of noise input and C_2 is the original classifier classifying the input dataset.

For the stability of the training procedure and the quality of the generated data, the objective functions are defined as follows.

$$\begin{aligned}
\mathbb{L}_D = &- \mathop{\mathbb{E}}_{z\sim p_z(z)}[D_{\theta_D}(G(z))] \\
&+ \mathop{\mathbb{E}}_{x\sim P_r(x)}[D_{\theta_D}(x)] + \lambda \mathop{\mathbb{E}}_{\hat{x}\sim P_{\hat{x}}(\hat{x})}[(\|\nabla_{\hat{x}} D_{\theta_D}(\hat{x})\|_2 - 1)^2], \quad (3.1a)\\
\mathbb{L}_G = &- \mathop{\mathbb{E}}_{z\sim p_z(z)}[D(G_{\theta_G}(z))], \quad (3.1b)\\
\mathbb{L}_C^r = &\mathop{\mathbb{E}}_{x\sim P_r(x),\ c_r}[-\log C_{\theta_C}(c = c_r|x)], \quad (3.1c)\\
\mathbb{L}_C^g = &\mathop{\mathbb{E}}_{z\sim P_z(z),\ c_g}[-\log C_{\theta_C}(c = c_g|G_{\theta_G}(z, c_g))], \quad (3.1d)\\
\mathbb{L}_C = &\rho\, \mathbb{L}_C^r + (1-\rho)\, \mathbb{L}_C^g. \quad (3.1e)
\end{aligned}$$

Note that $\mathbb{L}_D$, $\mathbb{L}_G$, $\mathbb{L}_C^r$, $\mathbb{L}_C^g$, and $\mathbb{L}_C$ are the loss functions of the discriminator,

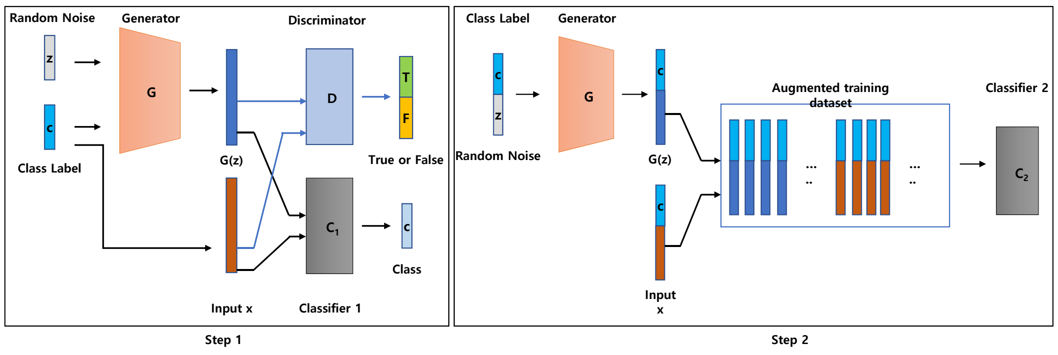


Figure 3.1: Overview of training procedure, of CEGAN (Step 1), of the classifier with augmented data (Step 2)

the generator, and the classifier with real and generated images, and the classifier, respectively. θ_D, θ_G, and θ_C are parameters of the discriminator, the generator, and the classifier, respectively. c_g is a sampled class label from the conditional data in the training set. w is a hyper-parameter that controls the importance of the classification for the real and generated data. Whereas the discriminator D is trained to minimize $\mathbb{L}_D$ as same as WGAN-GP, the generator G is trained to minimize $\mathbb{L}_G$ and $\mathbb{L}_C^g$ simultaneously. Furthermore, the classifier C is trained to minimize $\mathbb{L}_C^r$ with the real data and $\mathbb{L}_C$ with both the real data and the generated data. ρ is the hyper-parameter for the weighted sum of the classifier loss. The reason why the classifier C is optimized not only with the real data is to prevent the classifier from overfitting to the real data because the performance of the classifier with the real data from the minority classes is not enough to improve the performance of the classification under the data imbalanced conditions. The training details of CEGAN are summarized in Algorithm 1.

Additionally, we use the DCGAN architecture model, which is an extended model of the GAN that uses deconvolution layers in the generator and convolution layers in the discriminator to extract features [107]. Details on the architecture of the DCGAN in the proposed CEGAN can be found in Section 3.2.

The classifier in our CEGAN is trained following Algorithm 1. However, the classifier was originally designed for the real dataset, not for the synthetic data generated by GANs. As the generator takes as input a random noise vector and produces synthetic data, the classifier in the proposed CEGAN should consider the influence of noisy input. To actually reduce the impact of the noise in the training procedure of the CEGAN, we alternate the pooling method and the activation function of the classifier, such as LeNet5 [108] and VGGNet [109], in CEGAN.

First, we focus on the pooling layer to reduce the impact of noise. If some of the pixels in the sliding window are noisy, the max-pooling preserves noise and discards other meaningful pixels. However, we propose to use the average pooling in each

Algorithm 1 The training of CEGAN. We use default values of $\lambda = 10$, $n_D = 5$, $n_{C1} = 2$, $n_{C2} = 10$

```
Require: The batch size $m$, Adam hyperparameters $\alpha$, $\beta_1$, $\beta_2$, hyper-parameter for the weight sum $\rho$.
 1: for number of training iteration do
 2:     for $t = 1, ..., n_D$ do
 3:         Sample $\{x^{(i)}\}_{i=1}^{m} \sim P_r$ a batch from the real data.
 4:         Sample $\{z^{(i)}\}_{i=1}^{m} \sim P_z$ a generated batch.
 5:         $\hat{x} \leftarrow G_{\theta_G}(z)$
 6:         $\hat{x} \leftarrow \epsilon x + (1-\epsilon)\hat{x}$
 7:         $\theta_D \leftarrow Adam(\nabla_{\theta_D} \frac{1}{m} \sum D_{\theta_D}(x^{(i)}) - D_{\theta_D}(\hat{x}^{(i)}) + \lambda(\|\nabla_{\hat{x}^{(i)}} D_{\theta_D}(\hat{x}^{(i)})\|_2 - 1)^2, \theta_D, \alpha_D, \beta_1, \beta_2)$    ▷ Eq. (3.1a)
 8:     end for
 9:     Sample $\{z^{(i)}\}_{i=1}^{m} \sim P_z$ a generated batch.
10:     $\theta_G \leftarrow Adam(\nabla_{\theta_G} \frac{1}{m} \sum -D_{\theta_D}(G_{\theta_G}(z^{(i)})), \theta_G, \alpha_G, \beta_1, \beta_2)$    ▷ Eq. (3.1b)
11:     iter ← iter +1
12:     if iter $\equiv 0 (\mod n_{C1})$ then
13:         Sample $\{z^{(i)}\}_{i=1}^{m} \sim P_z$ a generated batch and labels $\{c_g^{(i)}\}_{i=1}^{m}$.
14:         $\theta_G \leftarrow Adam(\nabla_{\theta_G} - \frac{1}{m} \sum \log C_{\theta_C}(c = c_g^{(i)} | G_{\theta_G}(z, c_g^{(i)})), \theta_G, \alpha_G, \beta_1, \beta_2)$    ▷ Eq. (3.1d)
15:         Sample $\{x^{(i)}\}_{i=1}^{m} \sim P_r$ a batch from the real data and labels $\{c_r^{(i)}\}_{i=1}^{m}$.
16:         $\theta_C \leftarrow Adam(\nabla_{\theta_C} - \frac{1}{m} \sum \log C_{\theta_C}(c = c_r | x), \theta_C, \alpha_{C1}, \beta_1, \beta_2)$    ▷ Eq. (3.1c)
17:     end if
18:     if iter $\equiv 0 (\mod n_{C2})$ then
19:         Sample $\{z^{(i)}\}_{i=1}^{m} \sim P_z$ a generated batch and labels $\{c_g^{(i)}\}_{i=1}^{m}$.
20:         Sample $\{x^{(i)}\}_{i=1}^{m} \sim P_r$ a batch from the real data and labels $\{c_r^{(i)}\}_{i=1}^{m}$.
21:         $\theta_C \leftarrow Adam(\nabla_{\theta_C} - [\rho \frac{1}{m} \sum \log C_{\theta_C}(c = c_r | x) + (1-\rho) \frac{1}{m} \sum \log C_{\theta_C}(c = c_g^{(i)} | G_{\theta_G}(z, c_g^{(i)}))], \theta_C, \alpha_{C2}, \beta_1, \beta_2)$    ▷ Eq. (3.1e)
22:     end if
23: end for
```

pooling layer for the classifier because the averaging can yield statistics for the small sliding windows and smooth out noise information. Thus, in the case of input data containing noise, the average pooling can not only lower the computation complexity but also reduce the effect of the speckle noise. Figure 3.2 shows the examples of the max pooling and average pooling.

Secondly, we set the activation function of the nodes in our CNN to reduce the effect of noise. Normally, a Rectified Linear Unit (ReLU) or a leaky ReLU is used as the activation function in a CNN as they both provide faster computation and better convergence. To reduce the influence of noise, we propose the use of the hyperbolic tangent (tanh) as the activation function for the classifier of the conditional GANs. The activation function tanh saturates big values, and we can expect it to mitigate

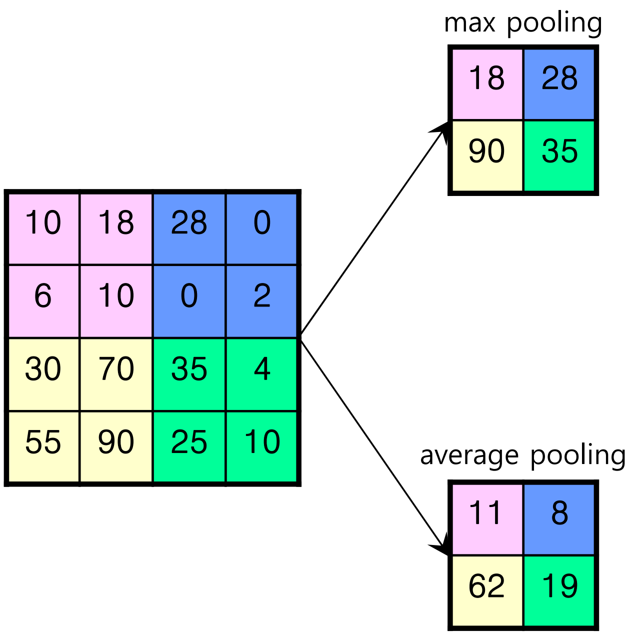


Figure 3.2: The example of the max pooling and average pooling

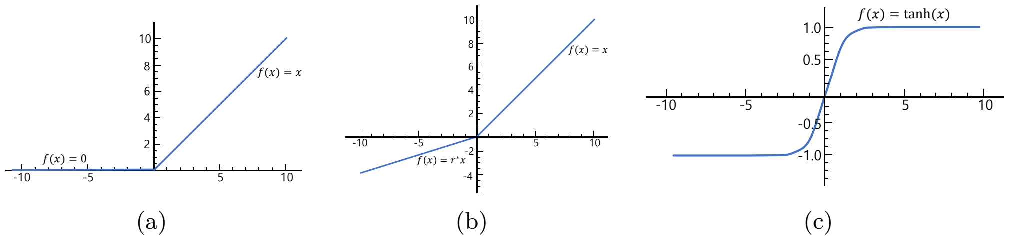


Figure 3.3: Activation functions, (a) ReLU, (b) Leaky ReLU, (c) Hyperbolic tangent (tanh)

the impact of noise, although ReLU and leaky ReLu are known to accelerate the convergence of the stochastic gradient descent compared with tanh. As shown in Figure 3.3, tanh saturates big values, and it is expected to prevent the influence of noise, though ReLU and leaky ReLu are known that it is to accelerate the convergence of stochastic gradient descent compared to tanh. In the next section, the experimental results show that the proposed modification reduces the effect of noise and improves the accuracy of the classifier.

Such a concept of the GAN with an independent classifier is similar to that of controllable GAN (ControlGAN) [110]. ControlGAN is also composed of three neural network structures, a discriminator, a generator/decoder, and an independent classifier/encoder. It employs an equilibrium parameter to balance the learning of

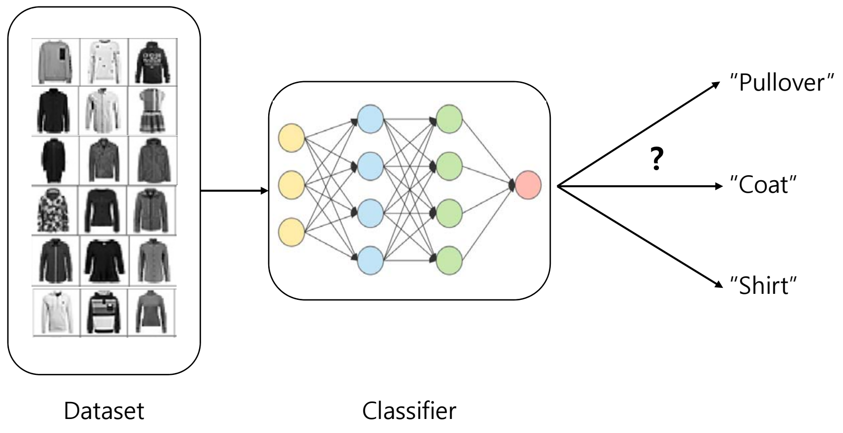


Figure 3.4: Example of ambiguous classes in fashion-MNIST benchmark dataset

the GAN structure and the decoder-encoder structure. Another noteworthy point of ControlGAN is that the classification loss with generated data is used only for training the generator. In contrast, the classifier of CEGAN is trained with both real and generated data, preventing it from overfitting the imbalanced data. Furthermore, the simple modification of the classifier architecture was proposed to reduce the impact of the noise in the training procedure.

3.1.2 CEGAN-AR: CEGAN for Ambiguity Reduction

Typical conditional GANs generate a synthetic image by assuming that class boundaries are clear. However, the assumption is not applicable in a real-world dataset as the boundaries between classes are very often unclear and ambiguous. For example, it is very difficult to distinguish classes such as "Pullover," "Coat," and "Shirt" in fashion-MNIST [111], as shown in Figure 3.4. In many cases, conditional GANs train the model only to fit discrete labels and do not consider ambiguous relationships beneath the data.

To overcome this limitation, we develop CEGAN using multiple subsets of class labels for ambiguous classes and call the developed model classification enhancement GAN for ambiguity reduction (CEGAN-AR). To this end, we propose a three-step subset extraction method where the resulting n-subsets include the ambiguous classes. Firstly, in order to extract the n-subsets, we train the classifier with a dataset that consists of the number of data in all classes equal to the number of data in the minority class. From the trained classifier network, features are extracted after the first fully connected (FC) layer. The dimension of the features is reduced to 100-dimensions by using principal component analysis (PCA) [112] and further reduced to two-dimensions by using t-Distributed Stochastic Neighbor Embedding (t-SNE) [113]. By this procedure, the features of the dataset are mapped into a two-dimensional plane. As a second step, we cluster the features into the number of the classes by applying KMeans++ algorithm [114]. This allows similar features to group together and the clustered features can be matched with the corresponding class. From each corresponding class, we calculate the class ambiguity it has with each other class by counting the number of misclassified features. We can extract highly misclassified combinations, and n-subsets are defined by grouping these combinations. Examples and details on the method of the subset extraction, applying to the fashion-MNIST and CIFAR-10 datasets, are presented in Section 3.2.

Using the defined subsets, the classifier of the proposed CEGAN is trained based on the ambiguous relationship in accordance with the following loss functions:

$$\begin{aligned}
\mathbb{L}_C^r &= \mathop{\mathbb{E}}_{x \sim P_r(x), c_r}[-\log C_{\theta_C}(c = c_r|x)] + \sum_k^{n_s} \lambda_k^r \mathop{\mathbb{E}}_{x \sim P_r(x), c_r}[-\log C_{\theta_C}^k(c = c_r|x) s_{kc_r}] \\
\mathbb{L}_C^g &= \mathop{\mathbb{E}}_{z \sim P_z(z), c_g}[-\log C_{\theta_C}(c = c_g|G(z, c_g))] \\
&\quad + \sum_k^{n_s} \lambda_k^g \mathop{\mathbb{E}}_{x \sim P_z(z), c_g}[-\log C_k(c = c_g|G_{\theta_G}(z, c_g)) s_{kc_r}],
\end{aligned} \tag{3.2}$$

where λ_k^r and λ_k^g are the hyper-parameters for the weighted sum of the defined subsets. $\mathbf{S} \in R^{n_s \times n_c}$ is a binary matrix with elements equal to either 0 or 1 determining the defined subsets, n_s is the number of subsets, and n_c is the total number of class labels in a given dataset. For example, in the fashion-MNIST dataset, three subsets are defined from the result of the proposed subset extraction method as follows.

Subset 1 = Pullover, Coat, Shirt

Subset 2 = Sandal, Sneaker, Ankle Boot

Subset 3 = T-Shirt, Dress, Shirt

Table 3.1 shows the correspondence for class labels for fashion-MNIST, CIFAR-10, and CINIC-10. Referring to the correspondence index, the matrix $\mathbf{S}$ for the three subsets is defined as

$$\mathbf{S} = \begin{bmatrix} 0\,0\,1\,0\,1\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0\,1\,0\,1\,0\,1 \\ 1\,0\,0\,1\,0\,0\,1\,0\,0\,0 \end{bmatrix} \tag{3.3}$$

Note that $\mathbf{S}$ is dependent on the definition of the subsets. In Eq.(3.2), the classifier C and the generator G are optimized similarly as the CEGAN classifier. However, upon closer inspection, the classifier C undergoes deep training to classify the ambiguous classes defined by $\mathbf{S}$ and the generator G attempts to generate synthetic data according to the classification of C.

Table 3.1: Class labels for fashion-MNIST and CIFAR-10/CINIC-10

Labels	**Fashion-MNIST**	**CIFAR-10/CINIC-10**
0	T-shirt/top	Airplane
1	Trouser	Automobile
2	Pullover	Bird
3	Dress	Cat
4	Coat	Deer
5	Sandal	Dog
6	Shirt	Frog
7	Sneaker	Horse
8	Bag	Ship
9	Ankle Boot	Truck

Table 3.2: Summary of the benchmark datasets

	MNIST	EMNIST	F-MNIST	CIFAR-10	CINIC-10
No.classes	10	10	10	10	10
No.training	5,000	24,000	6,000	5,000	9,000
No.test	1,000	4,000	1,000	1,000	9,000
Classifier model	LeNet5	LeNet5	LeNet5	VGGNet-16	VGGNet-16
Depth	1	1	1	3	3

3.2 Experimental Results

3.2.1 Datasets and Models

For the evaluation of the proposed method, we used five benchmark datasets: MNIST [115], extended MNIST digits [116], fashion-MNIST [111], CIFAR-10 [117], and CINIC-10 [118]. All benchmark datasets provide labeled training and a test set. Example images of the benchmark datasets are displayed in Figure 3.5. The LeNet5 [108] classifier model is used for MNIST, EMNIST, and fashion-MNIST, while the VGGNet-16 [109] classifier model is used for the CIFAR-10 and CINIC-10. A summary of the five benchmark datasets is shown in Table 3.2.

The datasets provide class-balanced data for training and testing. To generate an imbalance for evaluation, we adopt the step imbalance type introduced in [5]. The step imbalance assumes that the classes are divided into two groups: minority and majority. Then classes in the same group have the same number of data points, making a step in the data distribution plot. Because each of the five benchmark datasets is composed of ten classes, we set the number of majority classes as two and the other as eight to evaluate the worst case. Note that we set the test datasets under balanced conditions using all test data as shown in Table 3.2.

The experiments were all implemented using Python scripts in the Tensorflow framework and tested on a Linux system. Classifier models for the same dataset

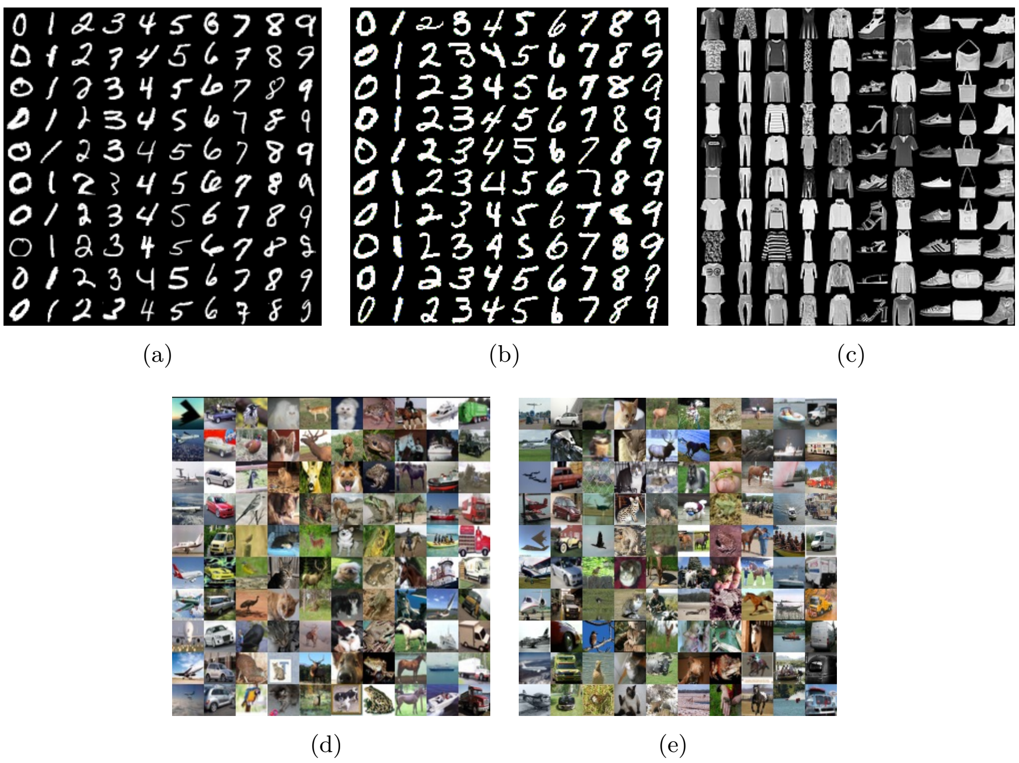


Figure 3.5: Example images of each dataset, (a) MNIST, (b) EMNIST, (c) fashion-MNIST, (d) CIFAR-10, (e) CINIC-10

were all trained with an equal epoch number of iterations even under data imbalance conditions.

The classifier model, used for MNIST, EMNIST, and fashion-MNIST, is LeNet5. The architecture of the classifier is shown in Table 3.3. As a preprocessing step, we normalized the pixel values to the range $[0, 1]$. The model is optimized using Adam with a learning rate of 0.001. The architectures of the generator and discriminator for MNIST, EMNIST, and fashion-MNIST are also shown in Table 3.3. To train these networks, we use the Adam optimizer [119] with parameters $\beta_1 = 0$, $\beta_2 = 0.9$, and a learning rate of $\alpha = 1.0 \times 10^{-4}$. As with the classifier model, two types of generator and discriminator models are used.

The classifier model, used for CIFAR-10 and CINIC-10, is VGGNet-16. The architecture of the model is presented in Table 3.4. CIFAR-10 and CINIC-10 are significantly more complex to classify than MNIST, EMNIST, and fashion-MNIST as the dataset contains color images, in contrast with the other three datasets, and the images have various backgrounds in addition to the target object. To improve the classification accuracy on the CIFAR-10 dataset, we normalized using global contrast normalization and ZCA whitening [120], and did not use any data augmentation method for preprocessing. For the CINIC-10 dataset, we did not normalize and

Table 3.3: Architecture of the LeNet5 classifier, the generator, and the discriminator models used for MNIST, EMNIST, and fashion-MNIST

Network	**Layers**	**Act. func.**	**Dimension**
LeNet5 Classifier	Input Image, pad=2	-	32 x 32 x 1
	Conv 5 x 5	ReLU	28 x 28 x 6
	Max Pooling 2 x 2	-	14 x 14 x 6
	Conv 5 x 5	ReLU	10 x 10 x 16
	Max Pooling 2 x 2	-	5 x 5 x 16
	Fully Connected	ReLU	120
	Fully Connected	ReLU	84
	Fully Connected	-	10
	SoftMax	-	10
Generator	Input	-	7 x 7x 128
	Upsample	-	14 x 14 x 128
	Conv 4 x 4	lReLU	14 x 14 x 128
	Conv 4 x 4	lReLU	14 x 14 x 64
	Upsample	-	28 x 28 x 64
	Conv 4 x 4	lReLU	28 x 28 x 32
	Conv 4 x 4	lReLU	28 x 28 x 16
	Upsample	-	56 x 56 x 16
	Conv 4 x 4	lReLU	56 x 56 x 4
	Conv 4 x 4, st=2	tanh	28 x 28 x 1
Discriminator	Input	-	28 x 28 x 1
	Conv 5 x 5, st=2	lReLU	14 x 14 x 32
	Conv 5 x 5, st=2	lReLU	7 x 7 x 64
	Conv 5 x 5, st=2	lReLU	4 x 4 x 128
	Conv 5 x 5, st=2	lReLU	4 x 4 x 256
	Conv 5 x 5, st=2	lReLU	4 x 4 x 256

applied augmentation methods such as horizontal flip and shift. Network weights were initialized using a Xavier procedure and the biases set to zero [121]. The model was optimized using stochastic gradient descent with a momentum value of $\mu = 0.9$ [122]. The base learning rate for the optimizer was multiplied by a fixed decay factor of 0.1 after 40, 80, and 120 epochs iteration, and the model was trained for 160 epochs on the CIFAR-10 and 300 epochs on the CINIC-10. In addition, the architectures of the generator and discriminator for CIFAR-10 and CINIC-10 are shown in Table 3.4.

Table 3.4: Architecture of VGGNet-16 classifier, the generator, and the discriminator models used for CIFAR-10 and CINIC-10

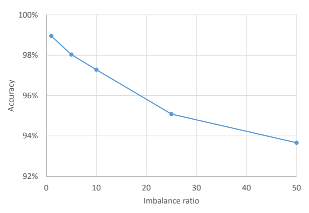
Network	Layers	Act. func.	Dimension
VGGNet-16 Classifier	Input Image	-	32 x 32 x 3
	2×Conv 3 x 3	ReLU	32 x 32 x 64
	Dropout 0.7	-	32 x 32 x 64
	Max Pooling 2 x 2, st=2	-	16 x 16 x 64
	2×Conv 3 x 3	ReLU	16 x 16 x 128
	Dropout 0.6	-	16 x 16 x 128
	Max Pooling 2 x 2, st=2	-	8 x 8 x 128
	3×Conv 3 x 3	ReLU	8 x 8 x 256
	Dropout 0.6	-	8 x 8 x 256
	Max Pooling 2 x 2, st=2	-	4 x 4 x 256
	3×Conv 3 x 3	ReLU	4 x 4 x 512
	Dropout 0.6	-	4 x 4 x 512
	Max Pooling 2 x 2, st=2	-	2 x 2 x 512
	3×Conv 3 x 3	ReLU	2 x 2 x 512
	Dropout 0.6	-	2 x 2 x 512
	Max Pooling 2 x 2, st=2	-	1 x 1 x 512
	Fully Connected	ReLU	512
	Fully Connected	ReLU	512
	Fully Connected	-	10
	SoftMax	-	10
Generator	Input	-	4 x 4 x 512
	DeConv 5 x 5, st=2	lReLU	8 x 8 x 256
	DeConv 5 x 5, st=2	lReLU	16 x 16 x 128
	DeConv 5 x 5, st=2	tanh	32 x 32 x 3
Discriminator	Input	-	32 x 32 x 3
	Conv 5 x 5, st=2	lReLU	16 x 16 x 128
	Conv 5 x 5, st=2	lReLU	8 x 8 x 256
	Conv 5 x 5, st=2	lReLU	4 x 4 x 512

3.2.2 Classification Results under Imbalanced Data Conditions and High-Variance Noise Input Conditions

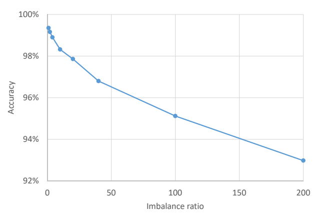
Before evaluating the proposed method, we verify the following two points: (1) that the data imbalance ratio (IR) defined below affects the performance of the classifier and (2) that the noise in the input data affects the performance of the classifier.

a) IR = (# of majority class data) : (# of minority class data),

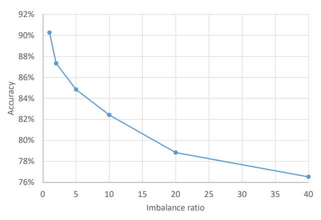
b) Adjusted IR (A-IR) = (# of majority class data) : (# of minority class data) + (# of augmented data of minority class)



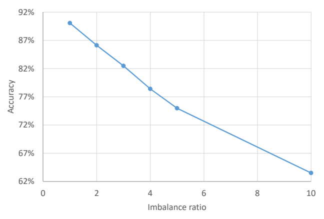
(a)



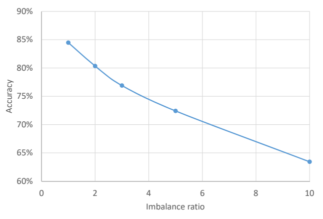
(b)



(c)



(d)



(e)

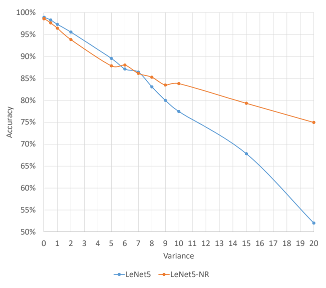
Figure 3.6: The accuracy of classification under various IR on each dataset, (a) MNIST, (b) EMNIST, (c) fashion-MNIST, (d) CIFAR-10, (e) CINIC-10

Table 3.5: The results of the simple exponential curve fitting for relationships between the accuracy of the classification and the IR

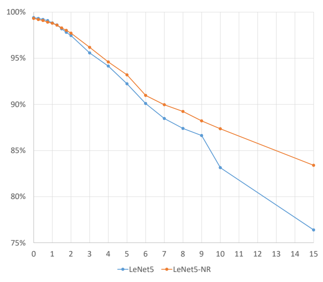
$y = c + ae^{-\lambda x}$	λ	a	c
MNIST	0.0382	0.0657	0.9265
EMNIST	0.0091	0.0729	0.9189
fashion-MNIST	0.0874	0.1397	0.7625
CIFAR-10	0.1022	0.4947	0.4563
CINIC-10	0.1334	0.3385	0.5461

To check the relationship between classification performance and data imbalance conditions, we train a model by setting the number of majority class data to the maximum number, and changing the IR from 1:1 to 50:1 on the MNIST, 200:1 on the EMNIST, 40:1 on the fashion-MNIST, 5:1 on the CIFAR-10, and 10:1 on the CINIC-10. Figure 3.6 displays the overall accuracy of the classification for various IRs on benchmark datasets. For example, the accuracy of the classifier on the MNIST, EMNIST, fashion-MNIST, CIFAR-10, and CINIC-10 under the data balanced condition was 98.95%, 99.35%, 90.26%, 90.06%, and 84.47%, respectively. However, as the IR increased, the accuracy of the classifiers on all datasets decreased dramatically. The results of the simple exponential curve fitting $y = c + ae^{-\lambda x}$ for relationships between the accuracy of the classification and the IR are shown in Table 3.5.

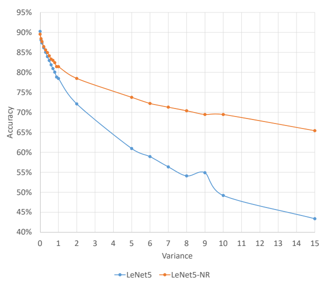
As the level of complexity of the classification increased, the accuracy tended



(a)



(b)



(c)

Figure 3.7: The accuracy of the classification on each dataset for various values of the variance of the Gaussian noise, (a) MNIST, (b) EMNIST, (c) fashion-MNIST

to decrease further. For example, the exponential decay constant for the CIFAR-10, $\lambda = 0.1022$, is almost three times larger than for the MNIST, $\lambda = 0.0382$. Consequently, the increasing IR has a negative impact on the performance of the classifiers.

The second experiment was to analyze the impact of the noise in the input data on the performance of the classifier. To evaluate the classifier in the noisy conditions, we added various variance values for the Gaussian noise into the training datasets. We used the EMNIST and fashion-MNIST benchmark datasets and compared the LeNet5 classifier to the LeNet5 with the noise reduction method (LeNet5-NR) that we introduced in Section 3.1. Figure 3.7 shows the accuracy trends of the classifier on the MNIST, EMNIST, and fashion-MNIST as the variance of the Gaussian noise increase. For the EMNIST dataset, the accuracy of LeNet5 is almost identical to that of LeNet5-NR in the absence of noise in the training dataset, or when the variance of the Gaussian noise is lower than 1.5. The accuracy of LeNet5 and LeNet5-NR in the absence of noise is 99.40% and 99.32%, respectively. The difference is close to 0.1%. However, the gap in accuracy increases as the variance exceeds 1.5. The accuracy of LeNet5-NR also decreases but remains at or above 83.41% even for the most severe variance value of 15, whereas the accuracy of the original LeNet5 falls to approximately 76.41% under the same condition. Regarding the fashion-MNIST dataset, the accuracy of the original LeNet5 classifier is higher than that of the LeNet5-NR only for the noise-free dataset. The difference in accuracy between LeNet5 and LeNet5-NR increases with the variance in the Gaussian noise. For the severe variance value of 15, the accuracy of the LeNet5-NR is close to 65%, while the accuracy of the LeNet5 falls below 45%. In summary, the proposed noise reduction method is capable of classifying images even when the variance of the Gaussian noise is high, and can potentially reduce the influence of the noise in the training images.

3.2.3 Evaluations of the Data Augmentation Methods

In this subsection, we evaluate the proposed method by comparing it to others, including the ACGAN [68], SMOTE [59], SVM-SMOTE [88], MWMOTE [89], VAE-GANs [123], BAGAN [124], and WGAN-GP [71], on the five benchmark datasets. In the experiments, two metrics are used to evaluate the data augmentation methods. One is the overall accuracy, which is the proportion of test data correctly classified. It informs us of the contribution of the data augmentation method to the improvement in the performance of the classifier. To calculate the exact accuracy with no influence from the majority class selection, the experiments were performed for all possible combinations, ${}_{10}C_2 = 45$, and the average accuracy was calculated. Another metric is the multiscale structural similarity (MS-SSIM) [125, 126]. MS-SSIM is a multiscale variant of the structural similarity metric designed to emulate the perception process in the human visual system [125]. MS-SSIM values vary from 0.0 to 1.0, where a higher value indicates higher similarity. Thus, a lower MS-SSIM score means that the images generated using data augmentation vary widely. To measure the image diversity, we calculate the MS-SSIM scores for 1000 pairs of images within a class of the generated dataset.

Experimental Results on MNIST

To evaluate the performance of the data augmentation methods, we generated synthetic images using the proposed method and various other data augmentation techniques and trained the classifier with the real and generated data. For the MNIST dataset, we oversampled over 100 images, randomly chosen for training the generative model, from each class using the proposed method and other various data augmentation techniques. As the maximum number of data points in each class is 5000 in the MNIST dataset, 100 images per class yield an IR of 50:1 and we oversample the 100 images chosen until the maximum number of data points in each class reaches 200, 500, 1000, and 5000 (yielding A-IR = 25:1, 10:1, 5:1, and 1:1, respectively). The sample images that were generated using the proposed method and various others are presented in Appendix B. As shown there, the ACGAN generated similar types of images. By contrast, the proposed method, CEGAN, provided a higher variety and better quality than the others.

As mentioned above, the classifier was trained for 45 cases and the average accuracy was calculated. The comparison results of the performance of the classifier are shown in Figure 3.8 and Table 3.6. The lower bound of the accuracy under IR = 50:1 with no oversampling was 93.66%. The upper bound of the accuracy under IR = 10:1 with no oversampling was 97.28%.

Among the data augmentation methods, CEGAN achieved the highest classification accuracy, and the accuracy at A-IR = 1:1 improved by 2.73% over that at IR = 50:1. Thus, the images generated by CEGAN can help improve the performance of the classification on the MNIST dataset. Furthermore, the proposed method reduces the complexity of the training procedure and is faster than WGAN-GP. Our network

Table 3.6: Percentile accuracy of real and generated data using data augmentation methods on the MNIST dataset. All the numbers are percentages. (IR = 50:1)

A-IR	**25:1**	**10:1**	**5:1**	**1:1**
ACGAN	93.52	93.46	94.01	93.84
SMOTE	93.85	94.13	94.42	94.60
SVM-SMOTE	93.92	94.46	94.23	95.06
MWMOTE	**94.80**	95.02	95.07	95.54
VAE-GANs	93.71	94.02	93.69	92.79
BAGAN	93.70	94.62	93.99	94.71
WGAN-GP	94.76	**95.57**	**96.07**	96.10
CEGAN	94.65	95.40	95.96	**96.39**

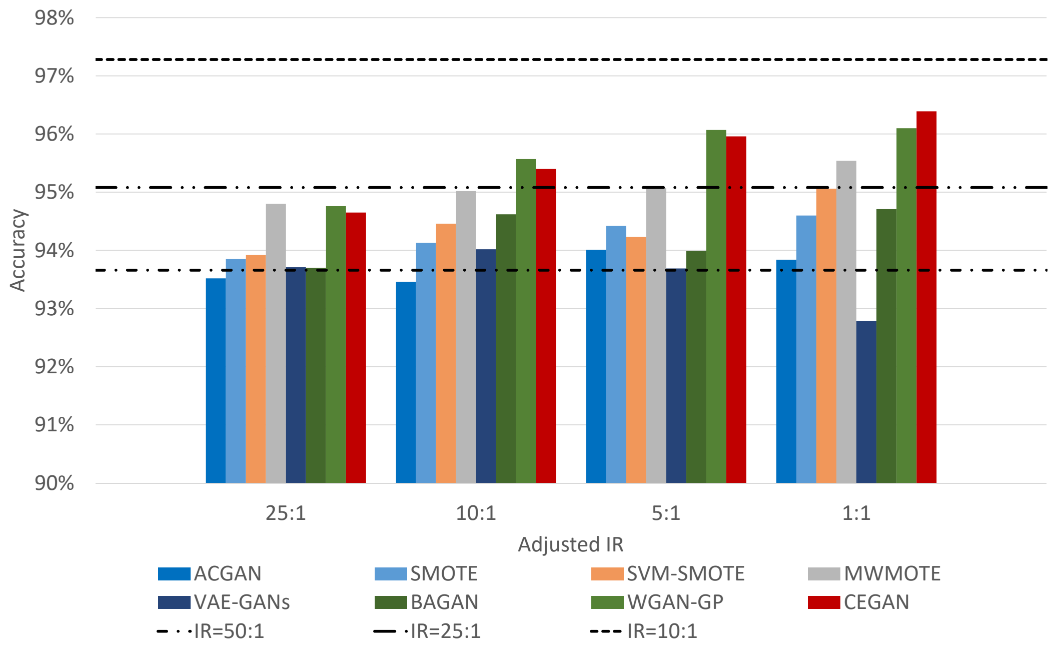


Figure 3.8: Comparison results of the classifier accuracy on the MNIST dataset

has a bigger structure than the one in WGAN-GP to handle the entire training data simultaneously. However, the WGAN-GP had to be trained using each class, while the proposed method was trained all at once. Moreover, the proposed method does not have to be ten times slower than the WGAN-GP as the filters in the convolution layers effectively capture the features.

Additionally, we measured and compared the MS-SSIM scores for the dataset generated by data augmentation. Figure 3.9 plots the mean MS-SSIM values for

Table 3.7: Mean MS-SSIM scores between the images generated by data augmentation on the MNIST dataset

Method	**mean MS-SSIM**
Training data	0.5680 ± 0.1015
ACGAN	0.6766 ± 0.0774
SMOTE	0.6828 ± 0.0908
SVM-SMOTE	0.6458 ± 0.0825
MWMOTE	0.6445 ± 0.0818
VAE-GANs	0.5959 ± 0.0853
BAGAN	0.5805 ± 0.1200
WGAN-GP	0.5660 ± 0.1108
CEGAN	**0.5574 ± 0.1062**

Table 3.8: MS-SSIM scores with the training data MS-SSIM score on the MNIST dataset

Method	**# of lower score classes**
ACGAN	0
SMOTE	0
SVM-SMOTE	0
MWMOTE	0
VAE-GANs	3
BAGAN	4
WGAN-GP	**7**
CEGAN	**7**

the images generated by data augmentation and the training data is broken up by class (i.e., ten scatter points for each method). The red line is the line of equality. For example, if a point is placed above the line of equality, the images generated for the class are not as diverse as the training data. Thus, when the points are overly distributed below the equality line, it means that the data augmentation method generated a wider variety of datasets than the training dataset. As shown in Table 3.8, most of the alternative data augmentation techniques only recorded three or four classes with a lower mean sample MS-SSIM score than the training data. By contrast, the number of classes below the equality line was seven for WGAN-GP and CEGAN. Table 3.7 provides a comparison of the overall mean MS-SSIM scores. Only the WGAN-GP and CEGAN provided lower mean MS-SSIM scores than the training data. Moreover, among all the data augmentation methods, the CEGAN achieved the lowest MS-SSIM score. In other words, the CEGAN generated more diverse synthetic images than other methods.

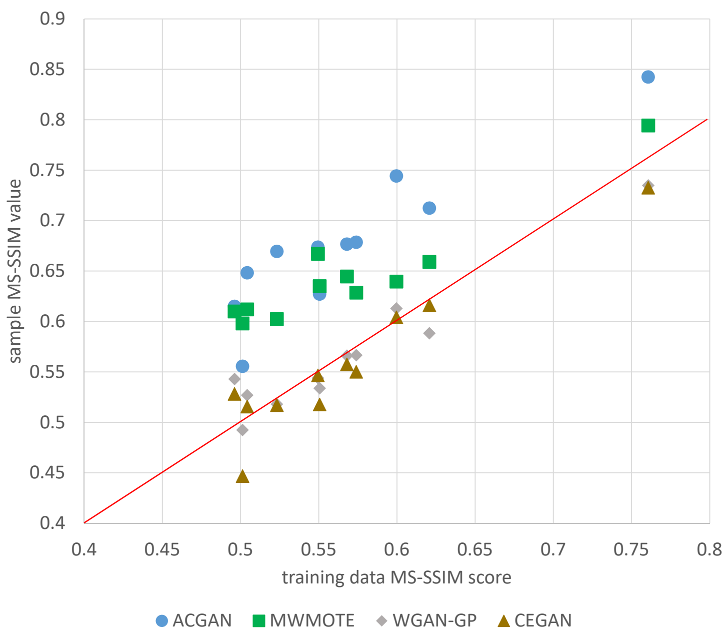


Figure 3.9: Comparison results of the MS-SSIM scores between the images generated by data augmentation on the MNIST dataset (the red line indicates equality)

Experimental Results on EMNIST

As with the MNIST dataset, we oversampled over 100 images, which were randomly chosen, from each class of the EMNIST digit dataset by using the proposed method and various others. Although the number of data points in each class in the training dataset is 24000 for the EMNIST dataset, we used 20000 images to readily calculate the IR, which is still four times as many as in the MNIST dataset. Then, 100 images per class yield an IR of 200:1, and we oversample the chosen 100 images until the maximum number of data points in each class reaches 2000, 5000, 10000, and 20000 (yielding A-IR = 10:1, 4:1, 2:1, and 1:1, respectively). The example images generated with the proposed method and various others are presented in Appendix B. As shown there, the mode collapse problem [127] occurs in the image set generated with ACGAN and VAE-GANs. However, CEGAN provides more varied types and better quality than other methods.

Similar to the MNIST dataset, we compare the classification accuracy according to the data augmentation method, as per Figure 3.10 and Table 3.9. The lower bound of the accuracy under IR = 200:1 with no oversampling was 92.98%. The upper bound of the accuracy under IR = 40:1 with no oversampling was 96.80%. Among data augmentation methods, our proposal, CEGAN, yielded the highest accuracy for the classification, improving the accuracy by 3.13% at A-IR = 1:1 over that at IR = 200:1. Thus, the images generated with the CEGAN can help improve the

Table 3.9: Percentile accuracy of real and synthetic data on the EMNIST dataset. All the numbers are percentages. (IR = 200:1)

A-IR	**10:1**	**4:1**	**2:1**	**1:1**
ACGAN	92.21	92.04	91.62	91.81
SMOTE	93.12	93.48	93.46	93.48
SVM-SMOTE	93.55	94.79	94.99	95.58
MWMOTE	94.68	94.58	94.83	95.33
VAE-GANs	92.55	91.82	92.28	92.33
BAGAN	94.40	93.74	94.08	94.03
WGAN-GP	**95.49**	**96.08**	**96.07**	96.32
CEGAN	95.36	95.66	**96.07**	**96.48**

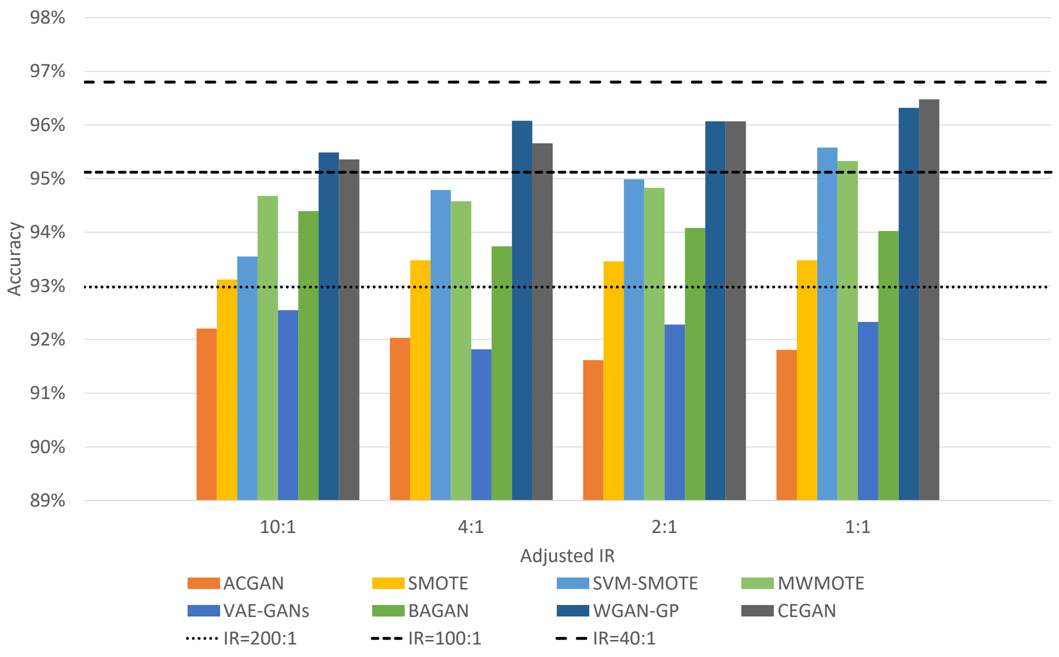


Figure 3.10: Comparison results of the classifier accuracy on the EMNIST dataset

performance of the classification on the EMNIST dataset as well.

We measured and compared the MS-SSIM scores in the same way as in Figure 3.9. As presented in Figure 3.11, we found that CEGAN is in a lower position than the other data augmentation methods. In Table 3.11, we found that the other data augmentation methods only have zero to three classes more diverse than the training data. By contrast, our method scored seven points below the equality line. Table 3.10 shows a comparison of the overall mean MS-SSIM scores. Only the WGAN-GP and CEGAN achieved a lower mean MS-SSIM score than the training

Table 3.10: Mean MS-SSIM scores among images generated by data augmentation on the EMNIST dataset

Method	**mean MS-SSIM**
Training data	0.4597 ± 0.1184
ACGAN	0.6098 ± 0.0742
SMOTE	0.5979 ± 0.1070
SVM-SMOTE	0.5834 ± 0.1014
MWMOTE	0.5777 ± 0.0934
VAE-GANs	0.5658 ± 0.1040
BAGAN	0.4832 ± 0.1235
WGAN-GP	**0.4462 ± 0.1252**
CEGAN	0.4471 ± 0.1200

Table 3.11: MS-SSIM scores with the training data on the EMNIST dataset

Method	**# of lower score classes**
ACGAN	0
SMOTE	0
SVM-SMOTE	0
MWMOTE	0
VAE-GANs	0
BAGAN	3
WGAN-GP	**8**
CEGAN	7

data. Moreover, among all data augmentation methods, the WGAN-GP achieved the lowest MS-SSIM score and has eight classes below the equality line. However, the mean MS-SSIM score with CEGAN is only slightly different from the score with the WGAN-GP. Consequently, we argue that CEGAN also outperforms other techniques on the EMNIST dataset.

Experimental Results on Fashion-MNIST

As in previous experiments, we oversampled over 150 images randomly chosen from each fashion-MNIST class. The images selected yielded IR = 40:1 and we oversampled them until the maximum number of data points in each class reached 300, 600, 1200, 3000, and 6000 (yielding A-IR = 10:1, 5:1, 2:1, and 1:1, respectively). In contrast with the MNIST and EMNIST datasets, the accuracy of the classification on the fashion-MNIST dataset was around 90% in Figure 3.6. The low accuracy stems from unclear boundaries between ambiguous classes, as mentioned in the previous section. Figure 3.12 shows the results of the feature extraction and clustering, that were described in the previous section, for 150 samples of each class from the fashion-MNIST dataset. In the merged result of the feature extraction and cluster-

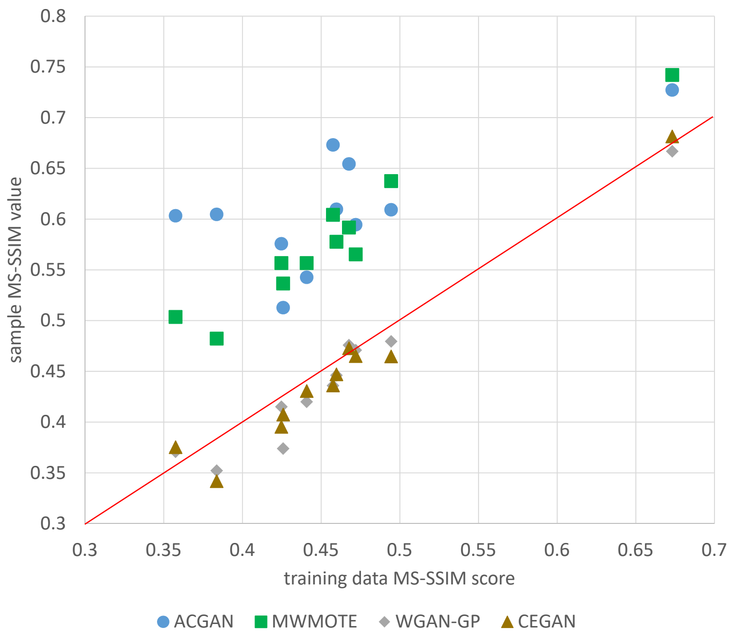


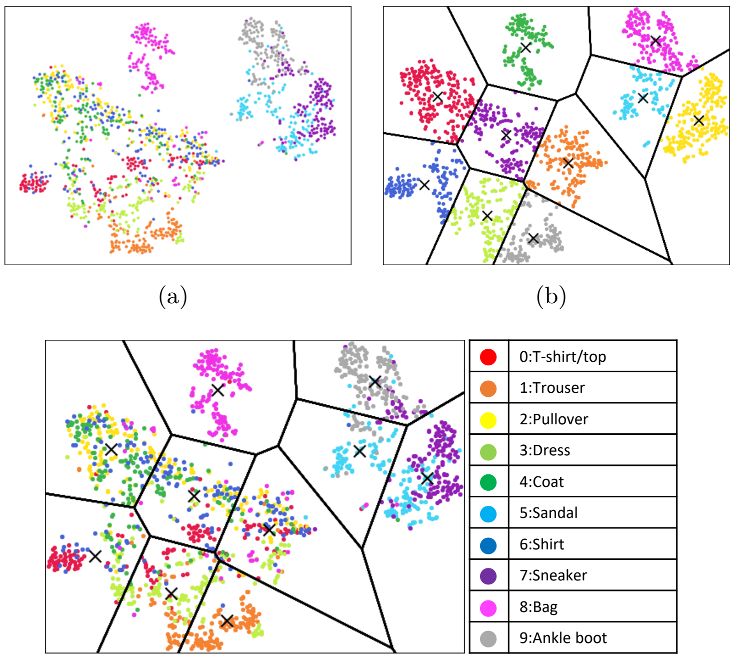
Figure 3.11: Comparison results of the MS-SSIM scores between the images generated by data augmentation on the EMNIST dataset (red line is equality)

Table 3.12: Highly correlated classes extracted by feature extraction and clustering for the fashion-MNIST dataset

Class 1	**Class 2**	**Score**
Pullover	Coat	110
Sandal	Sneaker	82
Pullover	Shirt	67
Coat	Shirt	64
T-shirt	Dress	55
T-shirt	Coat	54
T-shirt	Shirt	51
Dress	Coat	34
Sandal	Ankle Boot	26
Sneaker	Ankle Boot	26

ing from the fashion-MNIST dataset, several classes are mixed in some clusters. To derive subsets of ambiguous classes, we calculate an ambiguity score by counting the number of features among confused classes in the feature clustering map. The list of the high score class relationship and the score are shown in Table 3.12.

As a result, we define three subsets, which were mentioned in the previous section, from Table 3.12 for the CEGAN-AR in this experiment. Note that we use $\lambda_k^r = \lambda_k^g = 0.2$ for this experiment.



(a) (b)

(c)

Figure 3.12: Results of the feature extraction and clustering for 150 samples of each class from the fashion-MNIST dataset, (a) result of PCA and t-SNE, (b) result of KMeans++ algorithm, (c) merged result of feature extraction and clustering.

Table 3.13: Percentile accuracy of real and synthetic data on the fashion-MNIST dataset. All the numbers are percentages. (IR = 40:1)

A-IR	**10:1**	**5:1**	**2:1**	**1:1**
ACGAN	77.90	77.55	77.81	77.83
SMOTE	79.40	79.55	79.50	79.62
SVM-SMOTE	79.21	79.28	79.54	79.98
MWMOTE	76.66	77.19	80.02	80.95
WGAN-GP	**80.91**	**80.95**	80.95	81.03
CEGAN	79.96	80.59	80.89	81.44
CEGAN-AR	80.38	80.85	**81.47**	**81.72**

The example images generated with the proposed method and various others are presented in Appendix B. As shown in the figure in Appendix B, the mode collapse problem occurs in the image set generated with ACGAN. Meanwhile, SMOTE, MWMOTE, CEGAN, and CEGAN-AR generated high-quality images with no noise and presented various types simultaneously.

Experimental results are shown in Figure 3.13 and Table 3.13. The lower bound of

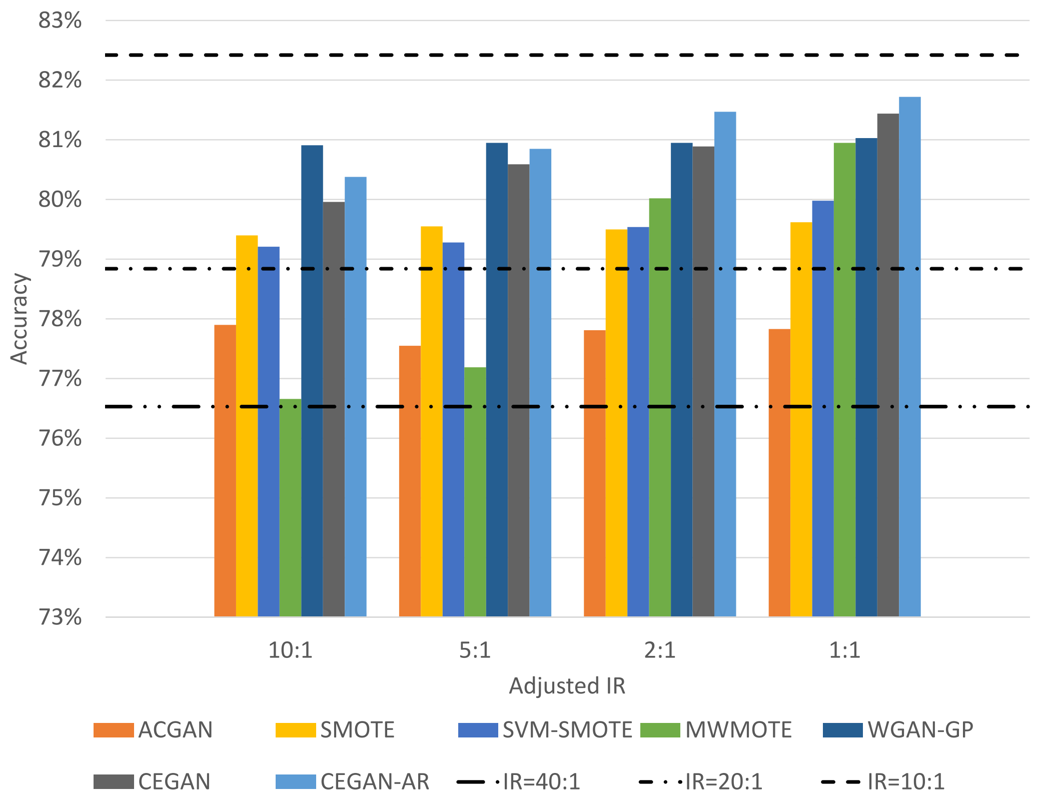


Figure 3.13: Comparative results of classifier accuracy on the fashion-MNIST dataset.

Table 3.14: Mean MS-SSIM scores among images generated by data augmentation on the fashion-MNIST dataset

Method	**mean MS-SSIM**
Training data	0.4563 ± 0.1306
ACGAN	0.8918 ± 0.1306
SMOTE	0.5575 ± 0.1217
SVM-SMOTE	0.5138 ± 0.1207
MWMOTE	0.5473 ± 0.1080
WGAN-GP	0.4536 ± 0.1343
CEGAN	0.4427 ± 0.1309
CEGAN-AR	**0.4358 ± 0.1208**

the accuracy under IR = 40:1 with no oversampling was 76.53%. The upper bound of the accuracy under IR = 10:1 with no oversampling was 82.42%.

Among data augmentation methods, the CEGAN-AR provided the highest ac-

Table 3.15: MS-SSIM scores with the training data on the fashion-MNIST dataset

Method	# of lower score classes
ACGAN	0
SMOTE	1
SVM-SMOTE	2
MWMOTE	1
WGAN-GP	4
CEGAN	4
CEGAN-AR	**5**

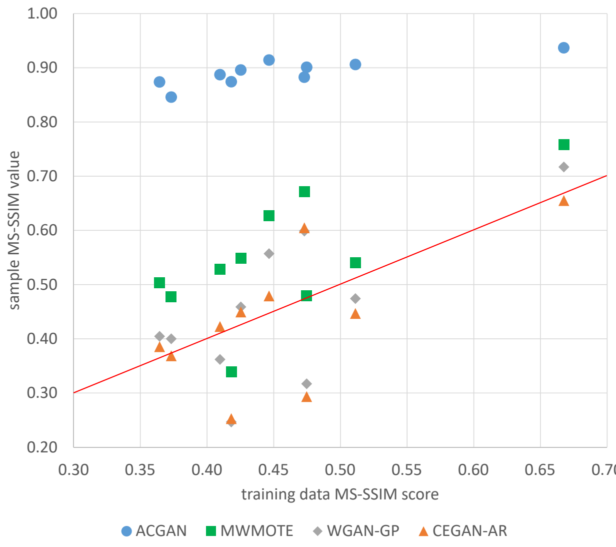


Figure 3.14: Comparative results of the MS-SSIM scores among images generated by data augmentation on the fashion-MNIST dataset (red line is equality)

curacy, improving it by 5.19% at A-IR = 1:1 over that at IR = 200:1. Additionally, the CEGAN-AR achieved higher accuracy than the CEGAN. Thus, we contend that CEGAN-AR can help improve classification performance on the fashion-MNIST dataset and achieve better classification results for ambiguous classes.

Once again we measured and compared the MS-SSIM scores in Figure 3.14 and Table 3.15. In the case of the fashion-MNIST dataset, there were zero to four classes for other methods and five classes for the CEGAN-AR with a lower mean MS-SSIM score than the training data. Based on the comparative results presented in Table 3.14, only the WGAN-GP, CEGAN, and CEGAN-AR achieved a lower mean MS-SSIM score than the training data.

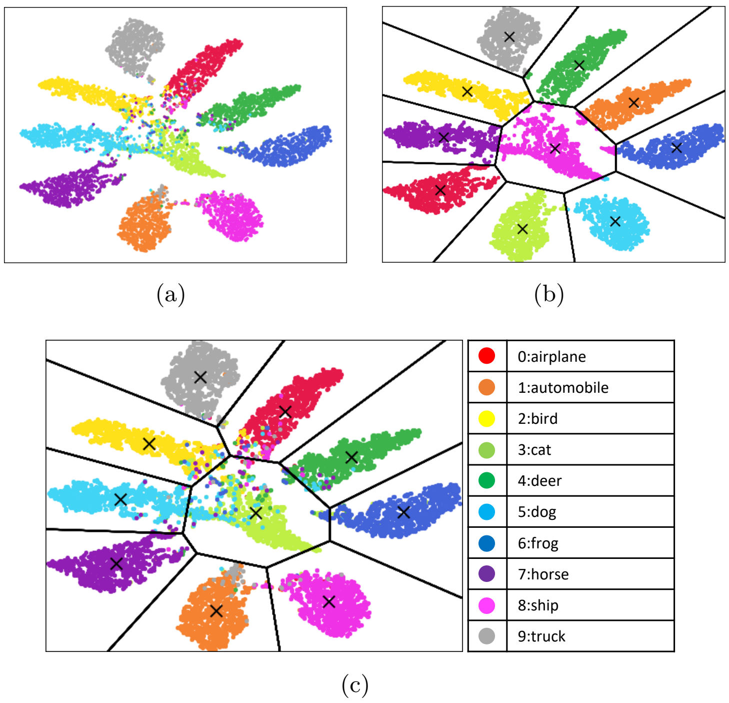


Figure 3.15: Results of the feature extraction and clustering for 1000 samples of each class from the CIFAR-10 dataset, (a) result of PCA and t-SNE, (b) result of KMeans++ algorithm, (c) merged result of feature extraction and clustering.

Experimental Results on CIFAR-10

The classification on the CIFAR-10 dataset is more difficult than on the other datasets. The accuracy of the classification even under the balanced condition was around 90% and the accuracy decreased dramatically as the IR increased. Unlike previous experiments, we sampled 1000 images from each class in the CIFAR-10 dataset for data augmentation (IR = 5:1). This is mainly because the quality of the images generated can also be affected and the performance of the classification cannot improve when its accuracy is too low. We oversampled over the 1000 images chosen until the maximum number of data points in each class reached 1250, 1667, 2500, 5000, yielding A-IR of 4:1, 3:1, 2:1, and 1:1, respectively.

In this experiment, we also defined subsets from the result of the feature extraction and clustering on the CIFAR-10 dataset for the CEGAN-AR. The results of the feature extraction and clustering for 1000 images from the CIFAR-10 dataset are shown in Figure 3.15. To derive subsets of ambiguous classes, we calculate an ambiguity score by counting the number of features among confused classes in the feature clustering map. The list of the high score class relationship and the score are shown in Table 3.16.

Table 3.16: Highly correlated classes extracted by feature extraction and clustering for the CIFAR-10 dataset

Class 1	**Class 2**	**Score**
Dog	Cat	144
Bird	Cat	81
Cat	Deer	69
Truck	Automobile	48

Table 3.17: Percentile accuracy of real and synthetic data on the CIFAR-10 dataset. All the numbers are percentages. (IR = 5:1)

A-IR	**4:1**	**3:1**	**2:1**	**1:1**
ACGAN	76.01	77.08	78.62	80.15
SMOTE	76.24	77.41	78.03	79.05
SVM-SMOTE	76.35	77.68	78.67	80.24
MWMOTE	76.55	**78.03**	79.27	80.50
WGAN-GP	76.26	77.91	**79.56**	79.56
CEGAN	76.58	78.01	78.81	80.77
CEGAN-AR	**76.76**	**78.03**	79.41	**81.75**

As a result, the three subsets are defined, referring to Table 3.16 for the CEGAN-AR in this experiment, as follows.

Subset 1 = Bird, Cat, Deer

Subset 2 = Cat, Dog

Subset 3 = Automobile, Truck

Note that we use $\lambda_k^r = \lambda_k^g = 0.2$ for this experiment. The example images generated by data augmentation are shown in Appendix B. The mode collapse problem occurs in the image set generated with ACGAN as well. The example images generated with WGAN-GP are difficult to recognize and contain noise as well. On the contrary, SMOTE, MWMOTE, CEGAN, and CEGAN-AR generated high-quality images with no noise and the generated images are comparatively easy to recognize. As shown in Figure 3.24, among the four example images from SMOTE, MWMOTE, CEGAN, and CEGAN-AR, the images generated with SMOTE seem to have the best quality, because the object is clear and the background is not noisy.

Experimental results are shown in Figure 3.16 and Table 3.17. The lower bound of the accuracy under IR = 5:1 with no oversampling was 74.97%. The upper bound of the accuracy under IR = 3:1 with no oversampling was 82.48%.

Among data augmentation methods, the CEGAN-AR achieved the highest accuracy, which improved by 6.78% at A-IR = 1:1 over that at IR = 5:1. Additionally, the CEGAN-AR led to a higher accuracy than the CEGAN. Thus, we contend that

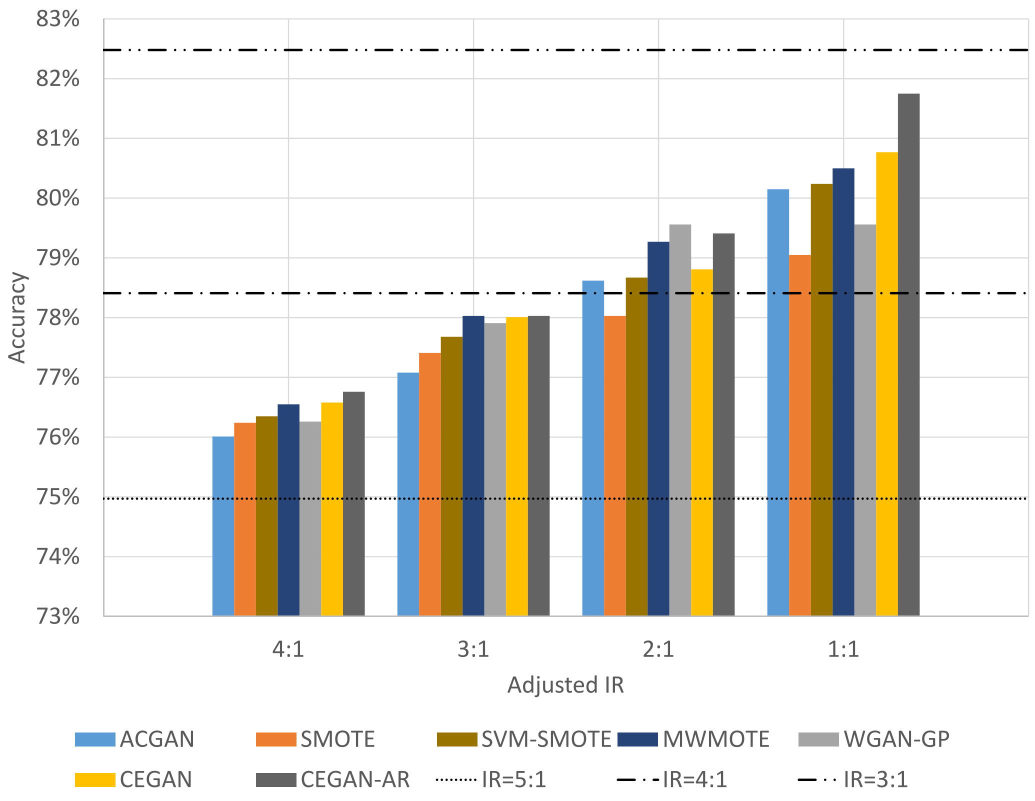


Figure 3.16: Comparison results of the classifier accuracy on CIFAR-10 dataset.

CEGAN-AR can help to improve the classification performance on the CIFAR-10 dataset and achieve a better classification of the ambiguous classes.

We measured and compared the MS-SSIM scores in Figure 3.17 and Table 3.19. In contrast with the other benchmark datasets, as the CIFAR-10 dataset has three-channel images and we oversampled over 1000 images, which is much more than for other datasets, the mean MS-SSIM score of the training data is far lower than the other datasets. In Table 3.19, there were zero to one class for the other methods and five classes for the WGAN-GP and CEGAN-AR with a lower mean MS-SSIM score than the training data. Based on the comparative results presented in Table 3.18, WGAN-GP, CEGAN, and CEGAN-AR achieved a lower mean MS-SSIM score than the training data.

Experimental Results on CINIC-10

CINIC-10 dataset contains 9000 training images in each class and the number of training images is 1.8 times as many as in the CIFAR-10 dataset. To show the effectiveness of the proposed method with more training images and under the severe

Table 3.18: Mean MS-SSIM scores among images generated by data augmentation on the CIFAR-10 dataset

Method	**mean MS-SSIM**
Training data	0.1230 ± 0.0653
ACGAN	0.2041 ± 0.0642
SMOTE	0.1715 ± 0.0827
SVM-SMOTE	0.1697 ± 0.1316
MWMOTE	0.1553 ± 0.0760
WGAN-GP	0.1187 ± 0.0607
CEGAN	0.1196 ± 0.0646
CEGAN-AR	**0.1180 ± 0.0644**

Table 3.19: MS-SSIM scores with the training data on the CIFAR-10 dataset

Method	**# of lower score classes**
ACGAN	1
SMOTE	0
SVM-SMOTE	1
MWMOTE	0
WGAN-GP	**5**
CEGAN	4
CEGAN-AR	**5**

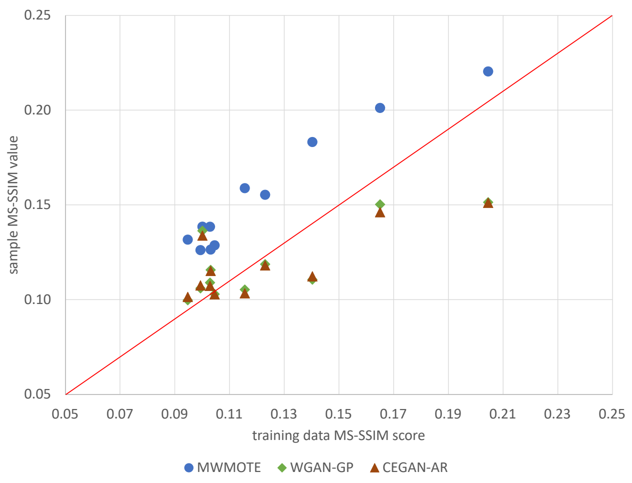


Figure 3.17: Comparative results of the MS-SSIM scores among images generated the data augmentation on the CIFAR-10 dataset (red line is equality)

IR condition, we sampled 900 images from each class in the CINIC-10 dataset for data augmentation (IR = 10:1). We oversampled over the 900 images chosen until

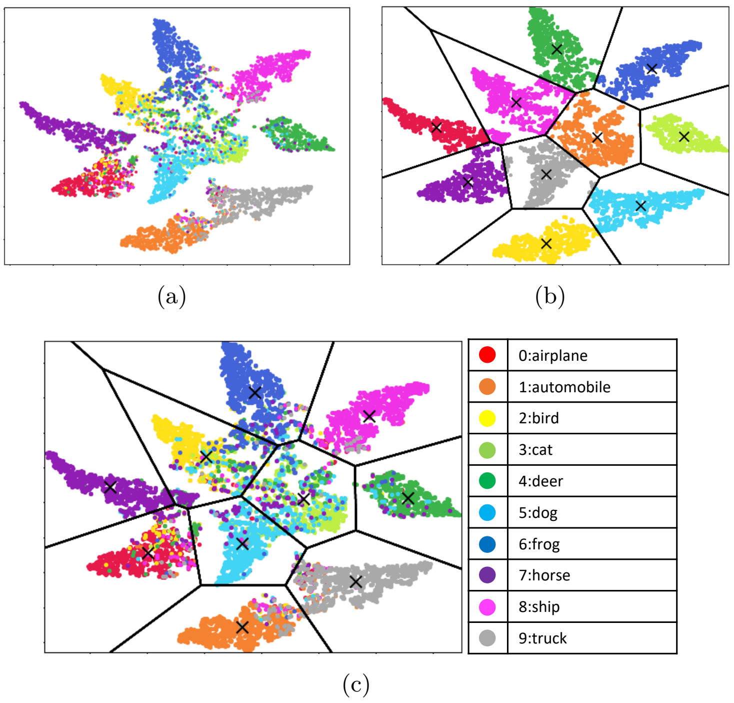


Figure 3.18: Results of the feature extraction and clustering for 900 samples of each class from the CINIC-10 dataset, (a) result of PCA and t-SNE, (b) result of KMeans++ algorithm, (c) merged result of feature extraction and clustering.

the maximum number of data points in each class reached 1800, 3000, 4500, 9000, yielding A-IR of 5:1, 3:1, 2:1, and 1:1, respectively.

In this experiment, we also defined subsets from the result of the feature extraction and clustering on the CINIC-10 dataset for the CEGAN-AR. The results of the feature extraction and clustering for 900 images from the CINIC-10 dataset are shown in Figure 3.18. To derive subsets of ambiguous classes, we calculate an ambiguity score by counting the number of features among confused classes in the feature clustering map. The list of the high score class relationship and the score are shown in Table 3.20.

As a result, the three subsets are defined, referring to Table 3.20 for the CEGAN-AR in this experiment, as follows.

Subset 1 = Cat, Dog

Subset 2 = Bird, Cat, Frog

Subset 3 = Automobile, Truck

Note that we use $\lambda_k^r = \lambda_k^g = 0.2$ for this experiment. The example images gen-

Table 3.20: Highly correlated classes extracted by feature extraction and clustering for the CINIC-10 dataset

Class 1	**Class 2**	**Score**
Dog	Cat	223
Truck	Automobile	100
Bird	Frog	97
Cat	Frog	95
Bird	Cat	91

Table 3.21: Percentile accuracy of real and synthetic data on the CINIC-10 dataset. All the numbers are percentages. (IR = 10:1)

A-IR	**5:1**	**3:1**	**2:1**	**1:1**
SMOTE	63.67	63.60	63.49	64.13
SVM-SMOTE	63.85	63.21	62.77	62.52
MWMOTE	65.79	66.04	65.90	65.41
CEGAN	65.76	65.86	66.14	66.62
CEGAN-AR	**66.09**	**66.12**	**66.28**	**66.74**

erated by data augmentation are shown in Appendix B. SMOTE, MWMOTE, CEGAN, and CEGAN-AR generated high-quality images with no noise and the generated images are comparatively easy to recognize. Among the four example images from SMOTE, MWMOTE, CEGAN, and CEGAN-AR, the images generated with SMOTE seem to have the best quality as the object in the images is clear.

Experimental results are shown in Figure 3.19 and Table 3.21. The lower bound of the accuracy under IR = 10:1 with no oversampling was 63.43%. The upper bound of the accuracy under IR = 5:1 with no oversampling was 72.41%. Because the performance of the classification under IR = 10:1 was low, the quality of the generated images can be affected and the improvement of the classification is little. Especially, the performance of the classification by SVM-SMOTE is lower than the baseline (IR=10:1), since the classification using SVM under the imbalanced condition in the CINIC-10 dataset was poor.

Among data augmentation methods, the CEGAN-AR achieved the highest accuracy, which improved by 3.31% at A-IR = 1:1. Additionally, the CEGAN-AR led to a higher accuracy than the CEGAN. Thus, we contend that CEGAN-AR can help to improve the classification performance on the CINIC-10 dataset and achieve a better classification of the ambiguous classes.

We measured and compared the MS-SSIM scores in Figure 3.20 and Table 3.23. In Table 3.23, there was zero class for the other methods and three classes for the CEGAN and CEGAN-AR with a lower mean MS-SSIM score than the training data. Unlike other experiments, the proposed method provided a higher mean MS-SSIM score than the training data. Based on the results of the accuracy and MS-SSIM

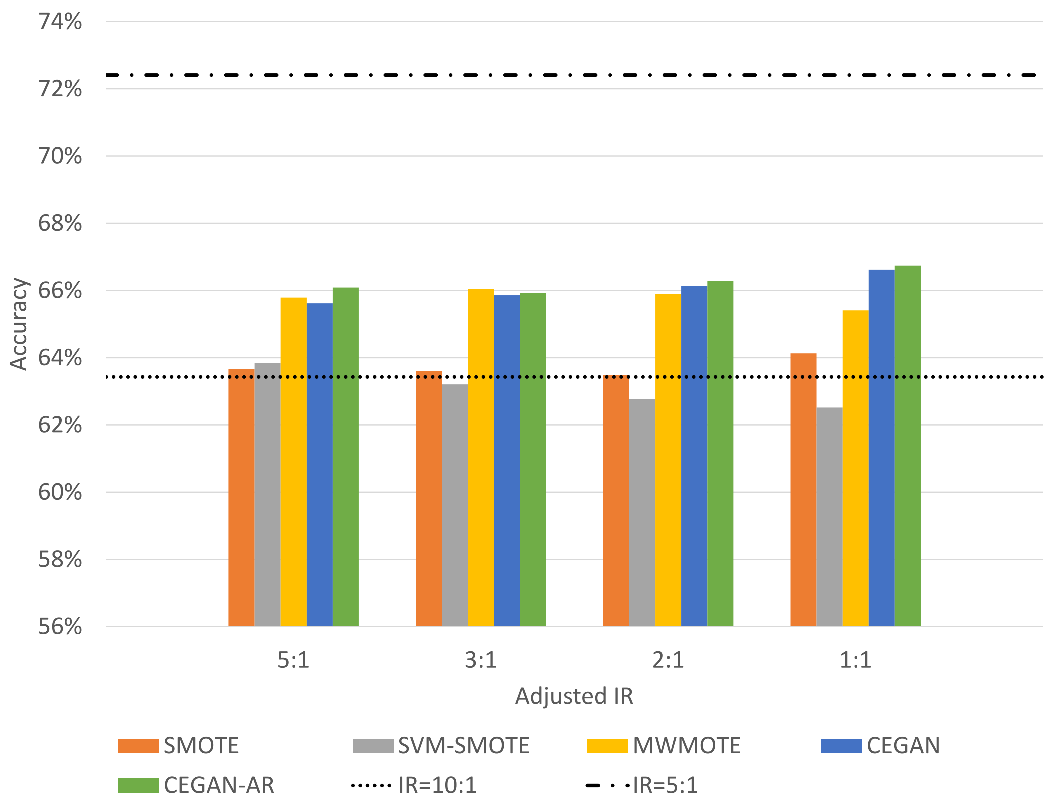


Figure 3.19: Comparison results of the classifier accuracy on CINIC-10 dataset.

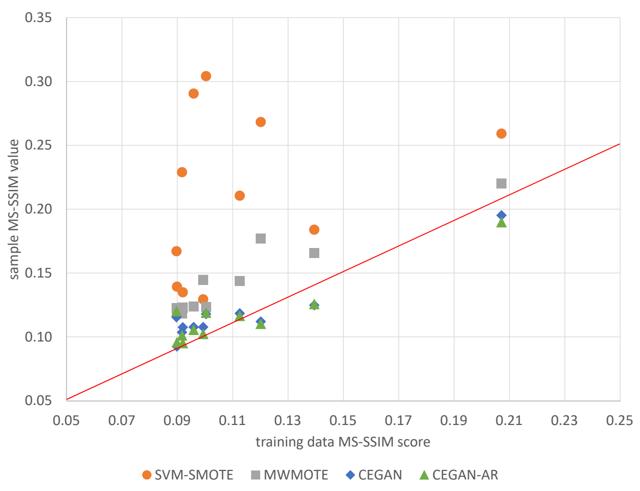


Figure 3.20: Comparative results of the MS-SSIM scores among images generated the data augmentation on the CINIC-10 dataset (red line is equality)

Table 3.22: Mean MS-SSIM scores among images generated by data augmentation on the CINIC-10 dataset

Method	**mean MS-SSIM**
Training data	0.1126 ± 0.0625
SMOTE	0.1616 ± 0.0798
SVM-SMOTE	0.2106 ± 0.1620
MWMOTE	0.1437 ± 0.0723
CEGAN	0.1184 ± 0.0813
CEGAN-AR	0.1164 ± 0.0800

Table 3.23: MS-SSIM scores with the training data on the CINIC-10 dataset

Method	**# of lower score classes**
SMOTE	0
SVM-SMOTE	0
MWMOTE	0
CEGAN	3
CEGAN-AR	3

Table 3.24: Average runtime of data augmentation methods in seconds

Method	**time**
SMOTE	35
SVM-SMOTE	22727
MWMOTE	25432
CEGAN	19287
CEGAN-AR	21580

score, we concluded that the performance of the classification and the diversity of the generated images have a limitation when the classification accuracy is low.

To measure the efficiency, the runtime on the CINIC-10 dataset is computed. The CINIC-10 dataset has three-dimensional images and the size of the training dataset is 90000 images. By using CINIC-10, the runtime of the proposed and other methods under the conditions where the computational load is high and the more severe data imbalance ratio is given. The experiment results show that the proposed method took much more time than SMOTE (600 times approximately), but less than SVM-SMOTE and MWMOTE. The runtime comparison results are shown in Table 3.24.

Examples of Generated Images

Figures 3.21, 3.22, 3.23, 3.24, and 3.25 show examples of the images generated on MNIST, EMNIST, fashion-MNIST, and CIFAR-10 datasets using various data augmentation methods, respectively. Note that other methods suffer from the mode

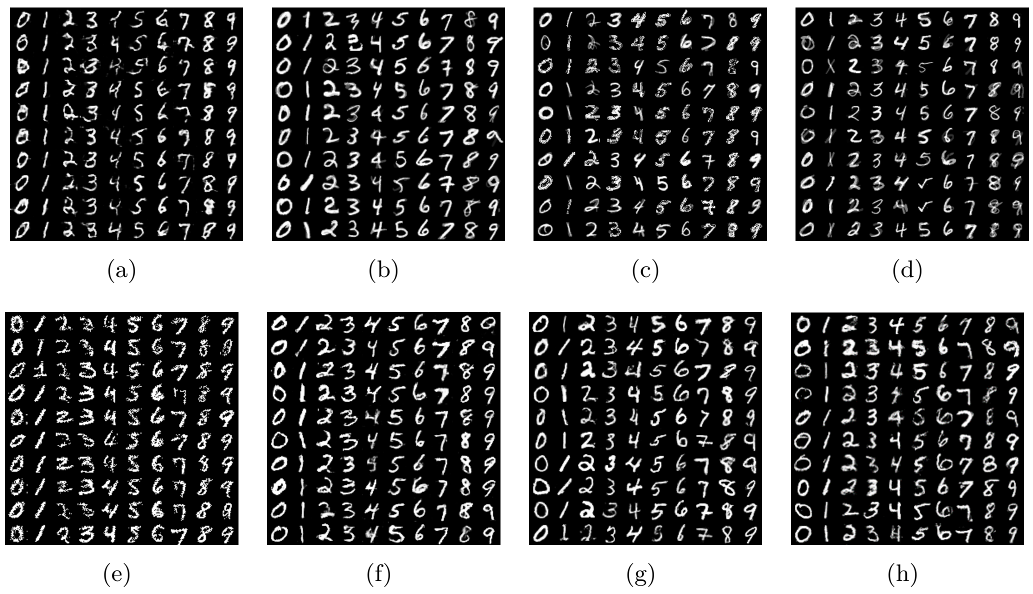


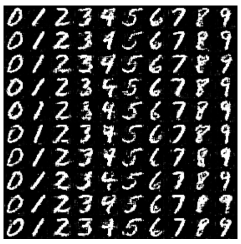
Figure 3.21: Examples of the generated images on MNIST dataset, By (a) ACGAN, (b) SMOTE, (c) SVM-SMOTE, (d) MWMOTE, (e) VAE-GANs, (f) BAGAN, (g) WGAN-GP, (h) CEGAN

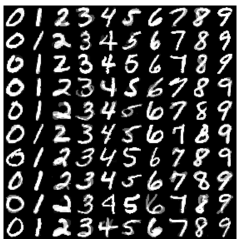
collapse problem or fail to distinguish confusing classes. By contrast, the technique proposed generates diverse synthetic data successfully while being effective at recognizing ambiguous classes.

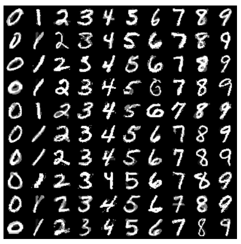
3.2.4 Overall Analysis

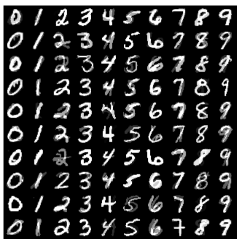
The experimental results presented above indicate that the proposed method improved the classification performance more than other oversampling methods and created an assorted yet appropriate set of synthetic images. Additionally, we observed that the lower the mean MS-SSIM score, the larger the improvement in classification accuracy. This demonstrates that the generative model improves the performance of the classification when the probability distribution is approximated well by the model and a diverse set of synthetic data is generated. Another interesting phenomenon we discovered is that the better performance CEGAN achieves, the more synthetic images are generated. The alternative techniques such as MWMOTE or WGAN-GP lead to slightly better outcomes than the proposed model in lower A-IRs.

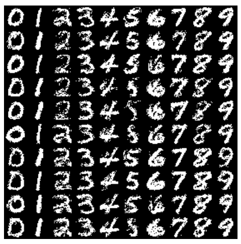
Conversely, when A-IR reaches 1:1, our method achieved the best performance for all five datasets. We interpret this result as follows: when the imbalance is severe, increasing the number of data points is more important than generating more diverse images. This explains why naive data augmentation such as SMOTE achieves better performance at higher A-IR. In this scenario, the data generated

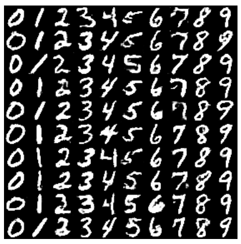

(a)

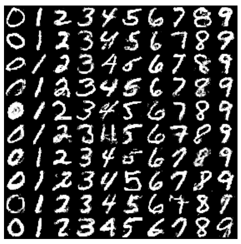

(b)


(c)


(d)


(e)


(f)


(g)

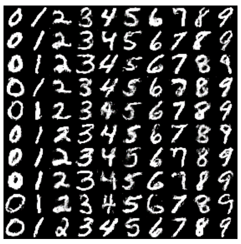
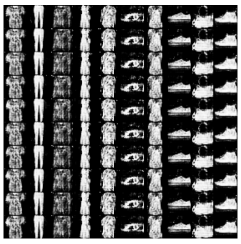
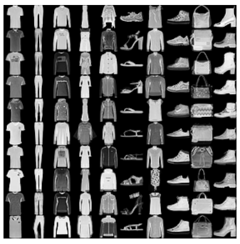
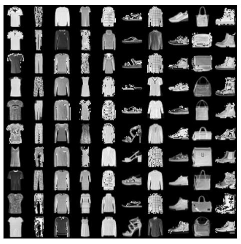
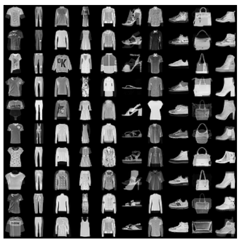

(h)

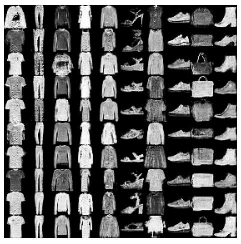
Figure 3.22: Examples of the generated images on EMNIST dataset, By (a) AC-GAN, (b) SMOTE, (c) SVM-SMOTE, (d) MWMOTE, (e) VAE-GANs, (f) BAGAN, (g) WGAN-GP, (h) CEGAN

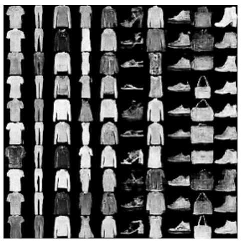

(a)


(b)


(c)


(d)


(e)


(f)

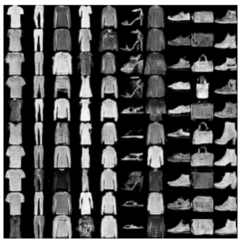

(g)

Figure 3.23: Examples of the generated images on fashion-MNIST dataset, By (a) ACGAN, (b) SMOTE, (c) SVM-SMOTE, (d) MWMOTE, (e) WGAN-GP, (f) CEGAN, (g) CEGAN-AR

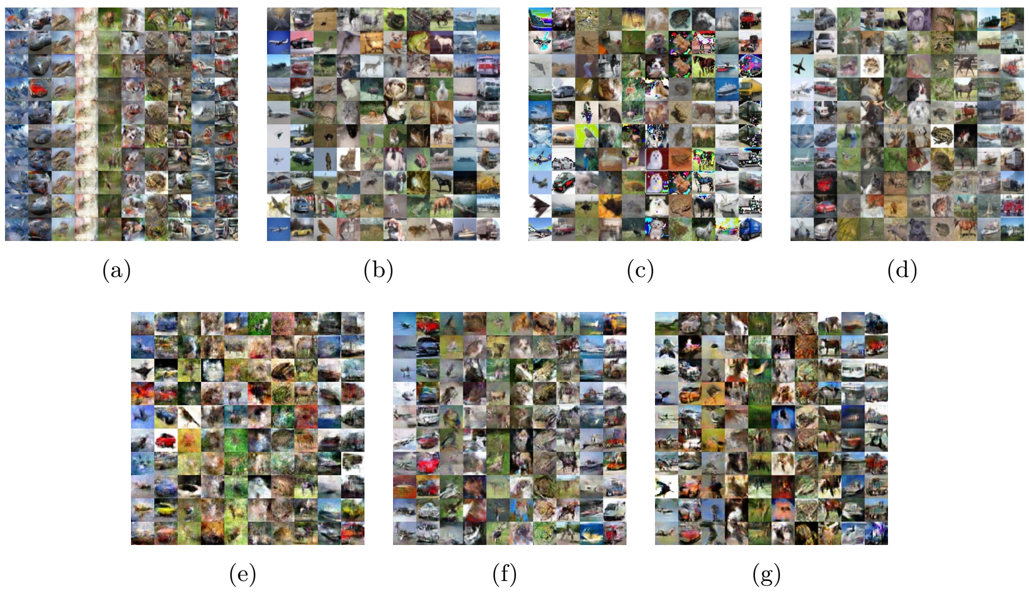


Figure 3.24: Examples of the generated images on CIFAR-10 dataset, By (a) AC-GAN, (b) SMOTE, (c) SVM-SMOTE, (d) MWMOTE, (e) WGAN-GP, (f) CEGAN, (g) CEGAN-AR

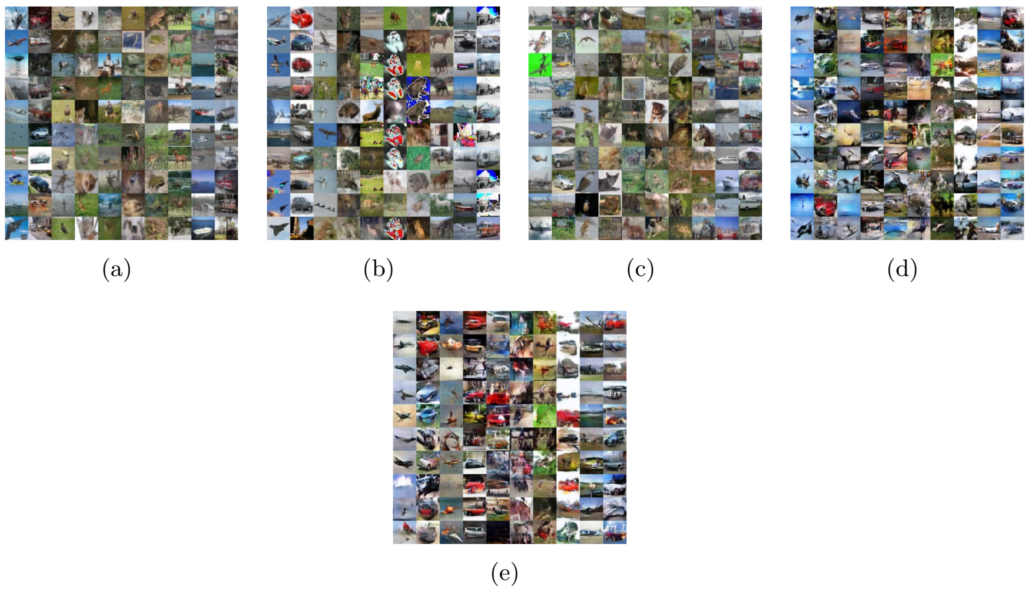


Figure 3.25: Examples of the generated images on CINIC-10 dataset, By (a) SMOTE, (b) SVM-SMOTE, (c) MWMOTE, (d) CEGAN, (e) CEGAN-AR

might be monotonous but is helpful enough to compensate for the bias caused by the imbalance. However, this effect decreases when A-IR approaches 1:1. As the generated data is within a certain boundary of diversity, it no longer insulates the classifier from the various cases. The method we propose soothes such a bias subject to the boundary of diversity.

3.3 Conclusions

In this chapter, we proposed a data augmentation method, named CEGAN, composed of three independent networks and employed the objective formulation of WGAN-GP, for classification under imbalanced data conditions. As the generator of GANs takes a random noise vector for input and outputs a synthetic image, we proposed a modified classifier architecture for generated images considering the effect of noise input. We also proposed a conditional generative model with class subsets, which were determined by feature extraction and clustering, to classify ambiguous classes more precisely, the CEGAN-AR. The experimental results showed that the proposed method improves the accuracy of the classification by 2.73%, 3.13%, 5.19%, 6.78%, and 3.31% on the MNIST, EMNIST, fashion-MNIST, CIFAR-10, and CINIC-10 datasets, respectively. Compared to others, our method provided the largest improvement in classification performance. In addition, we compared the diversity of the data generated by data augmentation using MS-SSIM scores. The proposed approach generated image samples with the lowest MS-SSIM score among all data augmentation methods.

4 Discriminative Feature Generation (DFG)

4.1 Framework of DFG (Application of CEGAN framework in the feature space with the supervised attention model)

The goal is to generate discriminative features to improve the performance of the neural network on a small number of training data and the class-imbalanced dataset. Unlike CGAN and ACGAN[67, 68], the structure of the feature classifier to the independent network is employed in the proposed GAN structure. The proposed GAN structure is composed of four independent networks: a feature generator, feature discriminator, feature extractor, and feature classifier. The structure of the proposed GAN model is shown in Figure 4.1. The feature extractor represents the first convolution layer or block in a neural network classifier, and the feature classifier represents the rest of the convolution layers and fully connected layers after the feature extractor in the neural network classifier. The feature extractor and classifier are regularized with transferred weights from the source model, similar to Li et al. [1]. We deploy the knowledge distillation technique to the training procedure of GAN and train the feature generator with supervised attention models to generate discriminative features.

4.1.1 Training GAN Model with Four Independent Networks

For the stability of the training procedure and the quality of the generated data, we deploy the objective formulation of WGAN-GP for the feature discriminator and the feature generator. The loss functions of the feature discriminator and the feature generator are denoted by $\mathbb{L}_D$ and $\mathbb{L}_G$, respectively.

$$\mathbb{L}_D(x, z, \hat{y}; \theta_D) = -\underset{(z,\hat{y})\sim(P_z,\hat{Y})}{\mathbb{E}}[D(G(z, \hat{y}))] + \underset{x\sim X}{\mathbb{E}}[D(E(x))] + \lambda \underset{\hat{x}\sim P_{\hat{x}}}{\mathbb{E}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2], \tag{4.1}$$

$$\mathbb{L}_G(z, \hat{y}; \theta_G) = -\underset{(z,\hat{y})\sim(P_z,\hat{Y})}{\mathbb{E}}[D(G(z, \hat{y}))], \tag{4.2}$$

where z is the noise vector sampled from uniform distribution P_z, and feature generator G generates synthetic feature $G(z)$. λ is the penalty coefficient, and $P_{\hat{x}}$ is

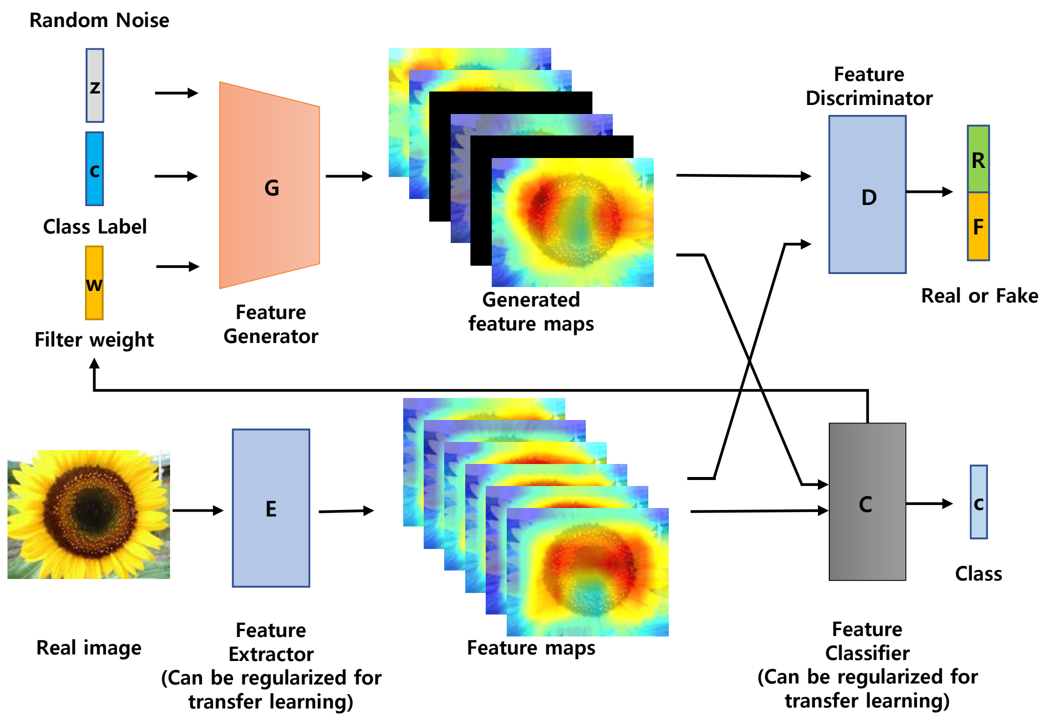


Figure 4.1: Structure of the proposed model.

the uniform sampling along straight lines between pairs of points from the real data distribution, P_r, and the generated data distribution. θ_D and θ_G are the parameters of the feature discriminator and the feature generator, respectively. Feature discriminator D is trained to minimize $\mathbb{L}_D$ to distinguish between real and generated features. Generator G is trained to minimize $\mathbb{L}_G$.

To generate the distribution of features similar to the real features from the feature extractor, a total loss function of the feature classifier contains a loss function of the feature classifier on the features concatenating the real and generated features as well. The objective functions of the feature extractor and classifier are represented as follows:

$$\mathbb{L}_E(x, y; \theta_E) = \mathop{\mathbb{E}}_{(x,y)\sim(X,Y)}[-y \log C(E(x))], \tag{4.3}$$

$$\begin{aligned}\mathbb{L}_C(x, y, z, \hat{y}; \theta_C) =& \alpha\, \mathbb{L}^r_C(x, y; \theta_C) + \beta\, \mathbb{L}^r_C(x, y; \theta_C) + \gamma L^c_C \\ & (\alpha + \beta + \gamma = 1).\end{aligned} \tag{4.4}$$

In (4.4), the loss functions of L^r_C, L^g_C, and L^c_C are defined as

$$\mathbb{L}^r_C(x, y; \theta_C) = \mathop{\mathbb{E}}_{(x,y)\sim(X,Y)}[-y \log C(E(x))], \tag{4.4a}$$

$$\mathbb{L}_C^g(z, \hat{y}; \theta_C) = \mathop{\mathbb{E}}_{(z,\hat{y})\sim(p_z(z),\hat{Y})} [-\hat{y} \log C(G(z, \hat{y}))], \tag{4.4b}$$

$$\begin{aligned}\mathbb{L}_C^c(x, y, z, \hat{y}; \theta_C, \theta_G) = &\mathop{\mathbb{E}}_{(\tilde{x},\tilde{y})\sim(\tilde{X},\tilde{Y})} [-\tilde{y} \log C(\tilde{x})], \\ &\text{where } \tilde{x} = E(x) \oplus G(z, \hat{y}),\ \tilde{y} = y \oplus \hat{y}\end{aligned} \tag{4.4c}$$

where $\mathbb{L}_E$ and $\mathbb{L}_C$ are the loss functions of the feature extractor and feature classifier, respectively. θ_E and θ_C are the parameters E and C, respectively. $\oplus$ denotes a concatenation operation. $\mathbb{L}_C^c$ is the loss function of the feature classifier on the feature maps concatenated with the real and generated features. α, β, and γ are hyperparameters that control the importance of the classification for the real and generated data. In $\mathbb{L}_E$ and $\mathbb{L}_C^r$, the regularization term, which characterizes the differences between the source and target network for transfer learning, can be added.

Feature generator G (to minimize (4.2) and (4.4c)) and feature classifier C (to minimize (4.4) and (4.4a)) are trained simultaneously, whereas feature discriminator D and feature extractor E are trained to minimize (4.1) and (4.3), respectively. The reason why feature classifier C is optimized not only with the real data is to prevent the classifier from overfitting to the real data because the performance of the classifier with the real data is not sufficient to improve the performance of the classification under the class-imbalanced condition.

In the training procedure, we define two training parameters. The generator learning parameter controls the ratio between $\mathbb{L}_G$ and $\mathbb{L}_C^g$ while optimizing the generator parameter, θ_G. The classifier learning parameter is used to control the balance of how much the classifier is trained from the feature extracted from the real data $\mathbb{L}_C^r$ or from the real features combined with the generated features, $\mathbb{L}_C$. The training details of the proposed method using (4.1), (4.2), (4.3), and (4.4) with these parameters are summarized in Algorithm 2.

4.1.2 Discriminative Feature Generation by using the Supervised Attention Model

To generate discriminative features, we adopt supervised attention mechanisms for each class label. To obtain the weights for feature maps, we propose a supervised attention method adopted from [1] for the generator network. Whereas the supervised attention method in [1] is calculated by averaging the filter weight for each filter, we utilize the fact that the importance of each filter varies from class to class. We transform the filter weights for a single data into class-wise filter weights and deactivate channels with low filter weights.

In the supervised attention method from [1], the weights of the features are characterized by the performance loss when removing the convolutional filter for each feature from the feature extractor network. For a conv2d layer in the feature extractor and generator, the parameter form is a four-dimensional tensor with the shape

Algorithm 2 Training procedure of the proposed DFG method. We use the default values of $\lambda = 10$, $n_D = 5$, $n_{C1} = 2$, $n_{C2} = 10, and n_W = 2000$

```
Require: Batch size $m$, learning rate $\eta$, hyperparameter for weight sum $\rho$, hyper-
    parameters $\alpha$, $\beta$, and $\gamma$, and a threshold value for filter weight $\delta$.
 1: Initialize: $\theta_E$, $\theta_C$ from pre-trained source networks $\theta_{E_S}$, $\theta_{C_S}$.
 2: $W_j(E_S; \theta_{E_S}) \leftarrow \mathrm{softmax}(-y \log C_S(E_{\theta^{\setminus j}_{E_S}}(x)) + y \log C_S(E_{\theta_{E_S}}(x)))$
 3: for $k$ = 1,..., $n_l$ number of class labels do
 4:     $W^*_j(E_S, y = k; \theta_{E_S}) \leftarrow W_j$ (if $W_j > \delta/n_l$), $W^*_j(E_S, y = k; \theta_{E_S}) \leftarrow 0$ (else)
 5: end for
 6: for $step = 1, ...,$ number of training iteration do
 7:     for $t = 1, ..., n_D$ do
 8:         Sample $\{x^{(i)}\}^m_{i=1} \sim P_r$ a batch from the real data.
 9:         Sample $\{z^{(i)}\}^m_{i=1} \sim P_z$ a generated batch and labels $\{\hat{y}^{(i)}\}^m_{i=1}$.
10:         Update feature discriminator $D$ using Eq. (4.1):
11:         $\theta_D \leftarrow \theta_D - \eta_D \nabla_{\theta_D} \mathbb{L}_D(x, z, \hat{y}; \theta_D)$,
12:     end for
13:     Sample $\{z^{(i)}\}^m_{i=1} \sim P_z$ a generated batch and labels $\{\hat{y}^{(i)}\}^m_{i=1}$.
14:     Sample $\{x^{(i)}\}^m_{i=1} \sim P_r$ a batch from the real data and labels $\{y^{(i)}\}^m_{i=1}$.
15:     Update feature generator $G$ using Eq. (4.2):
16:     $\theta_G \leftarrow \theta_G - \eta_G \nabla_{\theta_G} \mathbb{L}_G(z, \hat{y}, W^*_j; \theta_G)$,
17:     if $step \equiv 0 (\mod n_{C1})$ then
18:         Concatenate real and generated features following Eq. (4.4c): $\tilde{x}$, $\tilde{y}$,
19:         Update $G$ using Eq. (4.4c): $\theta_G \leftarrow \theta_G - \eta_G \nabla_{\theta_G} \mathbb{L}^c_C(x, y, z, \hat{y}, W^*_j; \theta_G)$,
20:         Update feature extractor $E$ using Eq. (4.3):
21:         $\theta_E \leftarrow \theta_E - \eta_E \nabla_{\theta_E} \mathbb{L}_E(x, y; \theta_E)$,
22:         Update feature classifier $C$ using Eq. (4.4a)
23:         $\theta_C \leftarrow \theta_C - \eta_C \nabla_{\theta_C} \mathbb{L}^r_C(x, y; \theta_C)$,
24:     end if
25:     if $step \equiv 0 (\mod n_{C2})$ then
26:         Concatenate real and generated features following Eq. (4.4c): $\tilde{x}$, $\tilde{y}$,
27:         Update $C$ using Eq. (4.4):
28:         $\theta_C \leftarrow \theta_C - \eta_C \nabla_{\theta_C} \mathbb{L}_C(x, y, z, \hat{y}, W^*_j; \theta_C)$,
29:     end if
30:     if iter $\equiv 0 (\mod n_W)$ then
31:         Update filter weight $W_j(x, y; \theta_E)$ using Eq.(4.5).
32:         for $k$ = 1,..., $n_l$ number of class labels do
33:             $W^*_j(y = k; \theta_E) \leftarrow \rho W_j(y = k; \theta_{E_S}) + (1 - \rho) W_j(y = k; \theta_E)$ Eq.(4.7) (if
    $W_j(c; \theta_E) > \delta/n_l$),
34:             $W^*_j(y = k; \theta_E) \leftarrow 0$ (else)
35:         end for
36:     end if
37: end for
```

of (c_{i+1}, c_i, k_h, k_w), where c_i denotes the number of channels of the i-th layer and (k_h, k_w) represents the size of the kernel. At the last convolution layer of the feature extractor, the dimension of the output feature maps is $(c_{i+1}, h_{i+1}, w_{i+1})$, which is matched with the dimension of the output of the feature generator. We can measure significance through the performance reduction of the feature extractor and classifier when the filter in the last convolution layer is disabled in the feature extractor network. In other words, because it usually causes higher performance loss to remove a filter with a greater capacity for discrimination, we can improve the performance of the classification by generating feature maps focusing on channels with high filter weights. The filter weight is calculated through the feature extractor and classifier and is input into the feature generator. The feature generator generates weighted feature maps with deactivated feature maps using the filter weight. The deactivated feature maps are denoted in black in Figure 4.1.

The filter weight is expressed in (4.5), which is used to calculate the gap between the classification losses of the feature extractor and classifier on data (x, y) with and without the j-th filter in the last convolution layer of the feature extraction network.

$$W_j(x, y; \theta_E) = \mathrm{softmax}[\underset{(x,y)\sim(X,Y)}{\mathbb{E}}[-y \log C(E_{\theta_E^{\setminus j}}(x))] - \underset{(x,y)\sim(X,Y)}{\mathbb{E}}[-y \log C(E_{\theta_E}(x))]], \tag{4.5}$$

where $\theta_E^{\setminus j}$ denotes the modified parameter from θ_E with all elements of the j-th filter set to zero.

Because each class usually has different importance of the filter channel, the calculated filter weights for a single data point can be transformed into class-wise filter weights.

$$W_j(c; \theta_E) = n_f \times \frac{W_j(x_i, y_i | y_i = c; \theta_E)}{n_c} \tag{4.6}$$

where $W_j(c; \theta_E)$ $(W \in \mathcal{R}^{n_c \times n_f})$ is a class-wise filter weight, c is a class label, n_c is the number of images in the class, and n_f is the number of filters. By multiplying the number of filters n_f, we set the average value of the weights to 1. When transfer learning is used for the small training dataset and class imbalance problem, a filter weight on the source model can be obtained through feature extractor E_S with parameter θ_{E_S} and classifier C_S with parameter θ_{C_S} trained on the source dataset. Whereas the filter weight in [1] was calculated on the source model only, the proposed method updates the filter weight on the target model. In this case, the filter weight can be expressed in the form of a weighted sum between the filter weight on the source dataset and the target dataset.

$$W_j(c; \theta_E) = \rho W_j(c; \theta_{E_S}) + (1 - \rho) W_j(c; \theta_E), \tag{4.7}$$

By applying the weight to the filters of the last layer in the feature generator, the feature generator can generate discriminative features by following the supervised attention models. The operation of the last layer in the original feature generator is expressed as follows.

$$f^l(z, \hat{y}) = \text{norm}[\tanh(\text{TransConv}(f^{l-1}(z, \hat{y}), \theta_G^l) + b^l)] \tag{4.8}$$

Note that f^l denotes the features of the l-th layer in the feature generator, norm() performs batch normalization, TransConv(f, θ) is a transpose convolution that takes f as the input with θ as a weight, and b^l is a bias vector of the l-th layer. To generate discriminative features by using the class-wise weights of a filter, the operation can be rewritten as follows.

$$\hat{f}_j^l(z, \hat{y}) = \text{norm}[W_j^*(c = \hat{y}; \theta_E) \cdot \tanh(\text{TransConv}_j(f^{l-1}(z, \hat{y}), \theta_G^l) + b_j^l)] \tag{4.9}$$

$\hat{f}^l(z, \hat{y})$ is a discriminative feature generated by the supervised attention model. To maximize the difference in the weights and inactivate unimportant filter channels, we set the weight values that are smaller than a threshold, δ, to zero and represent as W_j^*. δ is set to 0.95 of the average value of all filters for each class.

4.2 Experimental Results

4.2.1 Datasets and architectures

To evaluate the proposed method, we used the following six benchmark datasets: Street View House Numbers (SVHN) [128], Fashion-MNIST (F-MNIST) [111], STL-10 [129], CINIC-10 [118], Caltech-256 [130], and Food-101 [131]. As a classifier model, LeNet5 [108] for SVHN and F-MNIST, VGGNet-16 [109] for STL-10 and CINIC-10, and ResNet-50 [132] for Caltech-256 and Food-101 were employed. We used extended MNIST digits (EMNIST) [116], CIFAR-10 [117], and ImageNet [133] as the source domain for LeNet-5, VGG-16, and ResNet-50, respectively. The architecture of our feature discriminator and feature generator closely follow the deep convolutional GAN (DCGAN) architecture model [107], which is an extended model of the GAN that uses three transpose convolution layers in the generator and three convolution layers in the feature discriminator. A summary of the benchmarked dataset is presented in Table 4.1. The source code to reproduce our experiments is available at `https://github.com/opensuh/DFG`.

To evaluate the proposed method under the class-imbalanced condition, we adopt the step imbalance type described in [5]. The step imbalance assumes that the classes are divided into minority and majority groups. Then, classes in the same group have the same number of data points, making a step in the data distribution plot. In this evaluation, we set the number of majority classes as two and eight to evaluate under imbalanced conditions. All combinations of class imbalance cases were tested in the evaluation. For small training datasets such as STL-10 and Caltech-256, we did not create an imbalanced condition. Under the class balanced condition, each experiment was repeated 10 times.

Table 4.1: Summary of the benchmark datasets.

Dataset	Classifier	# of training dataset
SVHN	LeNet5	73,237
F-MNIST	LeNet5	60,000
STL-10	VGGNet-16	5,000
CINIC-10	VGGNet-16	18,000
Caltech-256	ResNet-50	30,607
Food-101	ResNet-50	75,750

SVHN: To test the transferability in the same domain, we set EMNIST as the source task and SVHN as the target task, where both are single-digit images. To reduce the image dimension gap, the SVHN images are converted to grayscale. For the imbalance 10:1 ratio set-up in training, the number of majority class images is 5000, and one of the minority class images is 500 in SVHN.

F-MNIST: To test the transferability in the different domains, we set EMNIST as the source task and F-MNIST as the target task which has images of fashion and clothing items. For imbalanced 40:1 ratio set-up in training, the number of majority class images is 6000 and that of minority class images is 150 in F-MNIST.

STL-10: STL-10 images are resized to 32 $\times$ 32 to match the image dimension of CIFAR-10. In this case, the class imbalance is not applied, but a small training dataset where 500 images per class are used in STL-10.

CINIC-10: CINIC-10 dataset contains 9000 training images in each class and we sampled 900 images from each minor class for an imbalance 10:1 ratio.

Caltech-256: Caltech 256 contains 257 object categories and 30607 images. In this study, we sampled 60 training samples for each category, referring to [75, 1]. We resized the input images to 256 $\times$ 256, followed by data augmentation operations of the random mirror and random crop to 224 $\times$ 224.

Food-101: Food-101 contains 101 food categories with 101000 images. A total of 750 training images and 250 test images were provided for each class. We sampled 250 training samples from minority classes for an imbalance of 5:1 ratio set-up during training.

All the experiments were implemented using Python scripts in the PyTorch framework and tested on a Linux system. Training procedures were performed on NVIDIA Tesla V100 GPUs.

We implemented the proposed DFG method with LeNet-5 [108], VGGNet-16 [109], and ResNet-50 [132] as the feature extractor and feature classifier. We divided the original architecture of the classifiers into the feature extractor and feature classifier. The architecture of the feature extractor and feature classifier is presented in Tables 4.2, 4.4, and 4.6. The output dimensions of the feature extractor and the feature generator should be the same. The architecture of our feature discriminator and feature generator closely follow the DCGAN architecture model [107], which is an extended model of the GAN that uses 3–4 transpose convolution layers in the

Table 4.2: Architecture of feature extractor and feature classifier based on LeNet-5 used for SVHN and fashion-MNIST datasets.

Network	Layers	Act. Func.	Dimension
Feature Extractor	Input Image	-	$32 \times 32 \times 1$
	Conv 5×5	ReLU	$28 \times 28 \times 6$
	Max Pooling 2×2	-	$14 \times 14 \times 6$
Feature Classifier	Input Feature	-	$14 \times 14 \times 6$
	Conv 5×5	ReLU	$10 \times 10 \times 16$
	Max Pooling 2×2	-	$5 \times 5 \times 16$
	Fully Connected	ReLU	120
	Fully Connected	ReLU	84
	Fully Connected	-	10
	SoftMax	-	10

generator and three convolution layers in the feature discriminator. The architecture of the generator and the discriminator for LeNet-5, VGGNet-16, and ResNet-50 are shown in Tables 4.3, 4.5, 4.7, respectively.

We used extended MNIST digits (EMNIST) [116], CIFAR-10 [117], and ImageNet [133] as the source domain for LeNet-5, VGG-16, and ResNet-50, respectively. For transfer learning, we followed the same procedure as in [1] because of the close relationship between the behavior regularization in this study and that one. After adopting the pre-trained weight only for ResNet-50 and before fine-tuning the network with the target dataset, we replaced the last layer of the base network with random initialization in a suit for the target dataset because the number of classes in the source dataset, ImageNet, is different from those in the target datasets, Caltech-256 [130] and Food-101 [131].

LeNet-5: LeNet-5 model was used for SVHN [128] and Fashion-MNIST (F-MNIST) [111]. To remove the image dimension difference, SVHN images were converted to grayscale and F-MNIST images were resized to 32×32 with zero-padding. The input images were normalized to zero-mean and the pixel values to range $[-1, 1]$. We did not use any data augmentation method for preprocessing. The weights of the feature extractor and feature classifier were pre-trained using EMNIST. The weights of the feature generator and feature discriminator were initialized using a Xavier procedure, and the biases were set to zero [121]. We used a batch size of 64 and the Adam optimizer for all networks with parameters $\beta_1 = 0.5$ and $\beta_2 = 0.9$ [119]. The learning rates of the feature extractor, feature classifier, feature generator, and feature discriminator for SVHN were 2.0×10^{-5}, 2.0×10^{-5}, 2.0×10^{-4}, and 1.0×10^{-4}, respectively (η_E, η_C, η_G, and η_D in Algorithm 2.). The learning rates for F-MNIST were $\eta_E = \eta_C = 1.0 \times 10^{-5}$ and $\eta_G = \eta_D = 1.0 \times 10^{-4}$. The training steps were 20000 iterations. The parameters in Algorithm 2 were set to $n_{C1} = 2$, $n_{C2} = 10$, $n_W = 5000$, $\rho = 0.75$, $\delta = 0.95$, and $\alpha = \beta = \gamma = 0.333$.

VGGNet-16: VGGNet-16 model was used for STL-10 [129] and CINIC-10 [118].

Table 4.3: Architecture of the generator and the discriminator for LeNet-5 classifier on SVHN and fashion-MNIST datasets.

Network	**Layers**	**Act. func.**	**Dimension**
Feature generator	Input	-	110
	FC	-	$4 \times 4 \times 128$
	BatchNorm	-	$4 \times 4 \times 128$
	DeConv 3×3, st = 2	lReLU	$9 \times 9 \times 48$
	BatchNorm	-	$9 \times 9 \times 48$
	DeConv 3×3, st = 1	lReLU	$11 \times 11 \times 12$
	BatchNorm	-	$11 \times 11 \times 12$
	DeConv 4×4, st = 1	Tanh	$14 \times 14 \times 6$
	BatchNorm	-	$14 \times 14 \times 6$
Feature discriminator	Input	-	$14 \times 14 \times 6$
	Conv 5×5, st = 2	lReLU	$7 \times 7 \times 32$
	Conv 5×5, st = 2	lReLU	$4 \times 4 \times 64$
	Conv 5×5, st = 2	lReLU	$2 \times 2 \times 128$
	FC	-	512

Table 4.4: Architecture of feature extractor and feature classifier based on VGGNet-16 used for STL-10 and CINIC-10 datasets.

Network	**Layers**	**Act. Func.**	**Dimension**
Feature Extractor	Input Image	-	$32 \times 32 \times 3$
	$2\times$Conv 3×3	ReLU	$32 \times 32 \times 64$
	Max Pooling 2×2, st = 2	-	$16 \times 16 \times 64$
Feature Classifier	Input Feature	-	$16 \times 16 \times 64$
	$2\times$Conv 3×3	ReLU	$16 \times 16 \times 128$
	Max Pooling 2×2, st = 2	-	$8 \times 8 \times 128$
	$3\times$Conv 3×3	ReLU	$8 \times 8 \times 256$
	Max Pooling 2×2, st = 2	-	$4 \times 4 \times 256$
	$3\times$Conv 3×3	ReLU	$4 \times 4 \times 512$
	Max Pooling 2×2, st = 2	-	$2 \times 2 \times 512$
	$3\times$Conv 3×3	ReLU	$2 \times 2 \times 512$
	Max Pooling 2×2, st = 2	-	$1 \times 1 \times 512$
	Adaptive Avg Pooling	-	$7 \times 7 \times 512$
	Fully Connected	ReLU	4096
	Fully Connected	ReLU	512
	Fully Connected	-	10
	SoftMax	-	10

CIFAR-10 (similar class labels to STL-10 and CINIC-10) was used for pre-training as the source domain. We downscaled the 96×96 image dimension of STL-10

Table 4.5: Architecture of the generator and the discriminator for VGGNet-16 classifier on STL-10 and CINIC-10 datasets.

Network	Layers	Act. func.	Dimension
Feature generator	Input	-	110
	FC	-	$4 \times 4 \times 128$
	BatchNorm	-	$4 \times 4 \times 128$
	DeConv 4×4, st = 2	lReLU	$8 \times 8 \times 128$
	BatchNorm	-	$8 \times 8 \times 128$
	DeConv 4×4, st = 2	lReLU	$16 \times 16 \times 128$
	BatchNorm	-	$16 \times 16 \times 128$
	DeConv 4×4, st = 1	lReLU	$17 \times 17 \times 64$
	BatchNorm	-	$17 \times 17 \times 64$
	DeConv 4×4, st = 1	Tanh	$16 \times 16 \times 64$
	BatchNorm	-	$16 \times 16 \times 64$
Feature discriminator	Input	-	$16 \times 16 \times 64$
	Conv 5×5, st = 2	lReLU	$8 \times 8 \times 32$
	Conv 5×5, st = 2	lReLU	$4 \times 4 \times 64$
	Conv 5×5, st = 2	lReLU	$2 \times 2 \times 128$
	FC	-	512

to match the 32×32 dimension of CIFAR-10 and CINIC-10. The input images were normalized to zero-mean and the pixel values to range $[-1, 1]$. We applied data augmentation operations of random horizontal flip and random crop with 4-pixels padding. The weights of the feature extractor and feature classifier were pre-trained with CIFAR-10. The weights of the feature generator and the feature discriminator were initialized using a He procedure, and the biases were set to zero [134]. We used a batch size of 64 and the Adam optimizer for all networks with parameters $\beta_1 = 0.5$ and $\beta_2 = 0.9$. The learning rates for STL-10 were set to $\eta_E = \eta_C = 5.0 \times 10^{-5}$ and $\eta_G = \eta_D = 1.0 \times 10^{-4}$. The learning rates for CINIC-10 were set to $\eta_E = \eta_C = 2.0 \times 10^{-5}$ and $\eta_G = \eta_D = 1.0 \times 10^{-4}$. The training steps for STL-10 and CINIC-10 were 20000 and 40000 iterations, respectively. The parameters in Algorithm 2 were set to $n_{C1} = 2$, $n_{C2} = 10$, $n_W = 5000$, $\rho = 0.75$, $\delta = 0.95$, and $\alpha = \beta = \gamma = 0.333$.

ResNet-50: ResNet-50 model was used for Caltech-256 [130] and Food-101 [131]. ImageNet was used as the source domain for pre-training. We replaced the last fully connected layer of ResNet-50 because of the different number of classes in ImageNet, Caltech-256, and Food-101 (1000, 257, and 101, respectively). For ResNet-50, the input images were normalized to zero-mean and the pixel values to range $[-1, 1]$ and resized to 256×256. We applied the data augmentation operation of the random crop to 224×224 and random horizontal flip. For the Caltech-256 dataset, we sampled 60 training samples and 20 testing samples for each category, according to [75, 1]. The weights of the feature extractor and the feature classifier were initial-

Table 4.6: Architecture of feature extractor and feature classifier based on ResNet-50 used for Caltech-256 and Food-101 datasets.

Network	**Layers**	**Act. Func.**	**Dimension**
Feature Extractor	Input Image	-	$224 \times 224 \times 3$
	Conv 7×7 st = 2	ReLU	$112 \times 112 \times 64$
	BatchNorm	-	$112 \times 112 \times 64$
	Max Pooling 3×3, st = 2	-	$55 \times 55 \times 64$
	Residual Block #1	ReLU	$55 \times 55 \times 256$
Feature Classifier	Input Feature	-	$55 \times 55 \times 256$
	Residual Block #2	ReLU	$28 \times 28 \times 512$
	Residual Block #3	ReLU	$14 \times 14 \times 1024$
	Residual Block #4	ReLU	$7 \times 7 \times 2048$
	Adaptive Avg Pooling	-	$1 \times 1 \times 2048$
	Fully Connected	-	No.Class
	SoftMax	-	No.Class

Table 4.7: Architecture of the generator and the discriminator for ResNet-50 classifier on Caltech-256 and Food-101 datasets.

Network	**Layers**	**Act. func.**	**Dimension**
Feature generator	Input	-	4900 + No.Class
	FC	-	$4 \times 4 \times 512$
	BatchNorm	-	$4 \times 4 \times 512$
	DeConv 4×4, st = 2	lReLU	$8 \times 8 \times 512$
	BatchNorm	-	$8 \times 8 \times 512$
	DeConv 4×4, st = 2	lReLU	$16 \times 16 \times 256$
	BatchNorm	-	$16 \times 16 \times 256$
	DeConv 2×2, st = 2	lReLU	$30 \times 30 \times 256$
	BatchNorm	-	$30 \times 30 \times 256$
	DeConv 2×2, st = 2	Tanh	$56 \times 56 \times 256$
	BatchNorm	-	$56 \times 56 \times 256$
Feature discriminator	Input	-	$56 \times 56 \times 256$
	Conv 5×5, st = 2	lReLU	$28 \times 28 \times 32$
	Conv 5×5, st = 2	lReLU	$14 \times 14 \times 64$
	Conv 5×5, st = 2	lReLU	$7 \times 7 \times 128$
	FC	-	6272
	FC	-	512

ized by the model pre-trained on ImageNet, provided by Torchvision [135], and the feature generator and the feature discriminator were initialized using a He initialization, and the biases were set to zero. We used a batch size of 64 and the Adam optimizer with parameters $\beta_1 = 0.5$ and $\beta_2 = 0.9$ for the feature generator and the

Table 4.8: Comparison results (in %) of our method with competing algorithms. The numbers are represented as *mean*±*std*. Four datasets are under the data-imbalanced condition, and two datasets are under the data-balanced conditions. The number of the training datasets is small.

Dataset (IR)	Original	Fine-tuning	DELTA	DIFA + cMWGAN	Ours(DFG)
SVHN (10:1)	76.57 ± 1.65	76.66 ± 0.65	78.47 ± 0.57	76.64 ± 1.32	**80.81 ± 0.25**
F-MNIST (40:1)	77.13 ± 3.24	75.39± 3.54	78.91 ± 2.27	78.27 ± 2.32	**82.65 ± 0.28**
STL-10 (1:1)	66.73 ± 0.67	72.89 ± 0.38	79.41 ± 0.27	80.08 ± 0.19	**81.09 ± 0.17**
CINIC-10 (10:1)	58.14 ± 3.42	64.42 ± 0.81	68.89 ± 0.18	71.82 ± 0.35	**72.09 ± 0.20**
Caltech-256 (1:1)	43.30 ± 0.34	81.99 ± 0.12	85.33 ± 0.15	82.23 ± 0.32	**86.29 ± 0.19**
Food-101 (5:1)	30.17 ± 1.72	68.99 ± 0.42	72.03 ± 0.35	71.71 ± 0.08	**76.00 ± 0.36**

feature discriminator, and stochastic gradient descent with a momentum of 0.9 for optimizing the feature extractor and the feature classifier. The learning rate for the feature generator and the feature discriminator was set to $\eta_G = \eta_D = 1.0 \times 10^{-4}$, and the learning rate for the feature extractor and feature classifier started with 0.01 and was divided by 10 after 20000 iterations. The training steps were completed at 40000 iterations. The parameters in Algorithm 2 were set to $n_{C1} = 1$, $n_{C2} = 5$, $n_W = 2000$, $\rho = 0.75$, $\delta = 0.95$, and $\alpha = \beta = \gamma = 0.333$.

Additionally, we used the following publicly available source code to implement our benchmarks.

- DELTA [1]: `https://github.com/lixingjian/DELTA`
- DIFA+cMWGAN [99, 100]:
  `https://github.com/ricvolpi/adversarial-feature-augmentation`

4.2.2 Results and comparisons

To establish a baseline for comparison, we denote “Original” as the performance of a classification model trained on the target dataset. “Original” is trained under the class-imbalanced condition in the case of SVHN, F-MNIST, CINIC-10, and Food-101. In the case of STL-10 and Caltech-256, “Original” is trained under the class-balanced condition that contains a small number of train data.

As mentioned, “Original” in Table 4.8 is trained without transfer learning and shows accuracies tested on the target test dataset achieved with classifiers trained on the target training dataset. The class imbalance problem, in which classes are selected as majority classes, is an important factor in determining the performance of the neural network, and the deviation of result accuracies is large. Thus, accuracies under the class imbalanced condition show the mean and standard deviation.

The proposed method is compared with previous fine-tuning methods.

- Fine-tuning: Traditional transfer learning method by fixing weights before the last convolution block and fine-tuning the rest of them.
- DELTA [1]
- DIFA+cMWGAN: The adversarial feature augmentation method based on [99]. We modified the structures of GAN with cMWGAN [100].

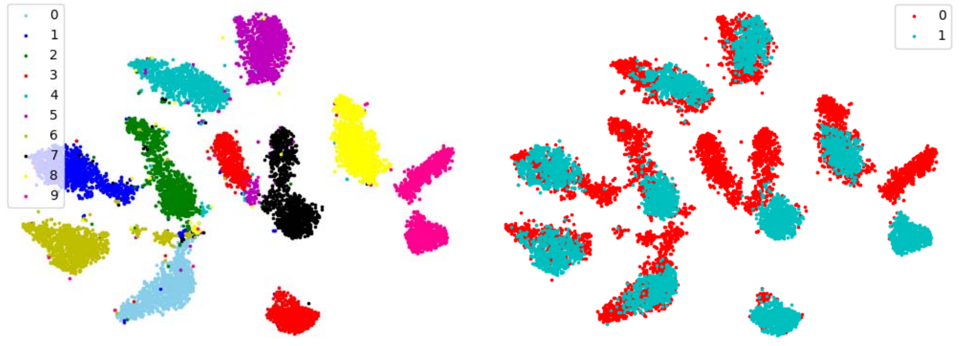
For a fair comparison with the other methods, we conducted experiments on the same neural network architecture for classification, such as LeNet5, VGGNet-16, and ResNet-50. Table 4.8 shows the results of the proposed DFG method compared to the fine-tuning, DELTA, and DIFA+cMWGAN methods. The results are the top-1 classification accuracies. The results in Table 4.8 show that our method based on DFG leads to accuracies higher than other methods for three different classifiers and six different benchmark datasets.

4.2.3 Qualitative results and ablation study

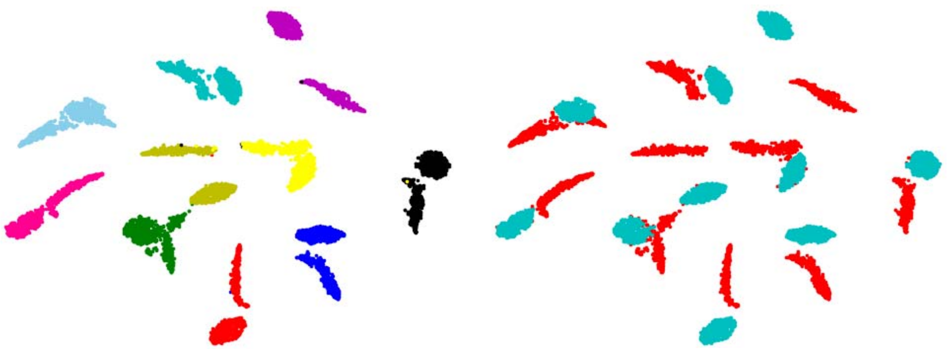
Although the proposed DFG method shows higher accuracy than the other methods, we would like to better understand the benefits brought by DFG and visualize how closely the proposed DFG generated the distribution of features that resembled that of the original in two-dimensional spaces. We apply PCA [112] and t-SNE [113] analyses of real and generated features. Figure 4.2 shows comparisons between the real and generated features. Different colors indicate different classes in the left part, and two different colors indicate real and generated features in the right part. From a qualitative point of view, real and generated features are distributed similarly, and the inter-class structure is preserved.

Additionally, we analyzed the activation maps from the proposed DFG method using class activation maps [136]. From Figures 4.3 and 4.4, we observe that the regions of the target object have higher activation than that of the original classifier and fine-tuning. Compared with the activation maps from the DELTA, the proposed method improved the concentration and activation degrees.

We carried out an ablation study to better understand how the filter weight changes during the training procedure. We perform an experiment by changing the ρ value in (4.7). Figure 4.5 presents the distribution of the weights of two representative filters in the case of the SVHN dataset for two different hyperparameters: $\rho = 0$ and 0.75. The first row in Figure 4.5 represents the distribution of the weights of two representative filters with $\rho = 0.75$, and the second row represents the distribution of the weights of the same filters with $\rho = 0$. The red straight line is the line of equality for all filters. For instance, if a weight value is placed above the line of equality, it means that the importance of the filter for the class is higher than the



(a)



(b)

Figure 4.2: t-SNE visualization of feature distribution on (a) SVHN and (b) STL. In the left part, different colors indicate different classes. In the right part, red and cyan dots indicate real and generated features, respectively.

average of all filters. The case with $\rho = 0$ involves the training of the feature generator with the weights of the filters by updating from the feature extractor trained on the target dataset, and the case with $\rho = 0.75$ involves the training of the feature generator with the weights of the filters by two feature extractors: one is trained on the source dataset and the other is trained on the target dataset in the training procedure. We observed a significant difference between the two distributions of the weights. The distribution of the weights of the filters with $\rho = 0$ converges to the equality baseline, whereas the distribution of the weights of the filters with $\rho = 0.75$ maintains the characteristics of the weight for each class label. Our hypothesis of a possible cause of the difference is that the feature extractor trained on the target dataset tends to be trained to activate all filters and be overfitted to the target dataset. The feature extractor trained on the source dataset inactivates several convolution filters, and a small number of filters are activated. The average accuracy with $\rho = 0.75$ is 80.81% and that with $\rho = 0$ is 79.98%. The weights of the filters with $\rho = 0.75$, using the weight values from the feature extractor trained on the source dataset, prevent overfitting and help to generate discriminative features for

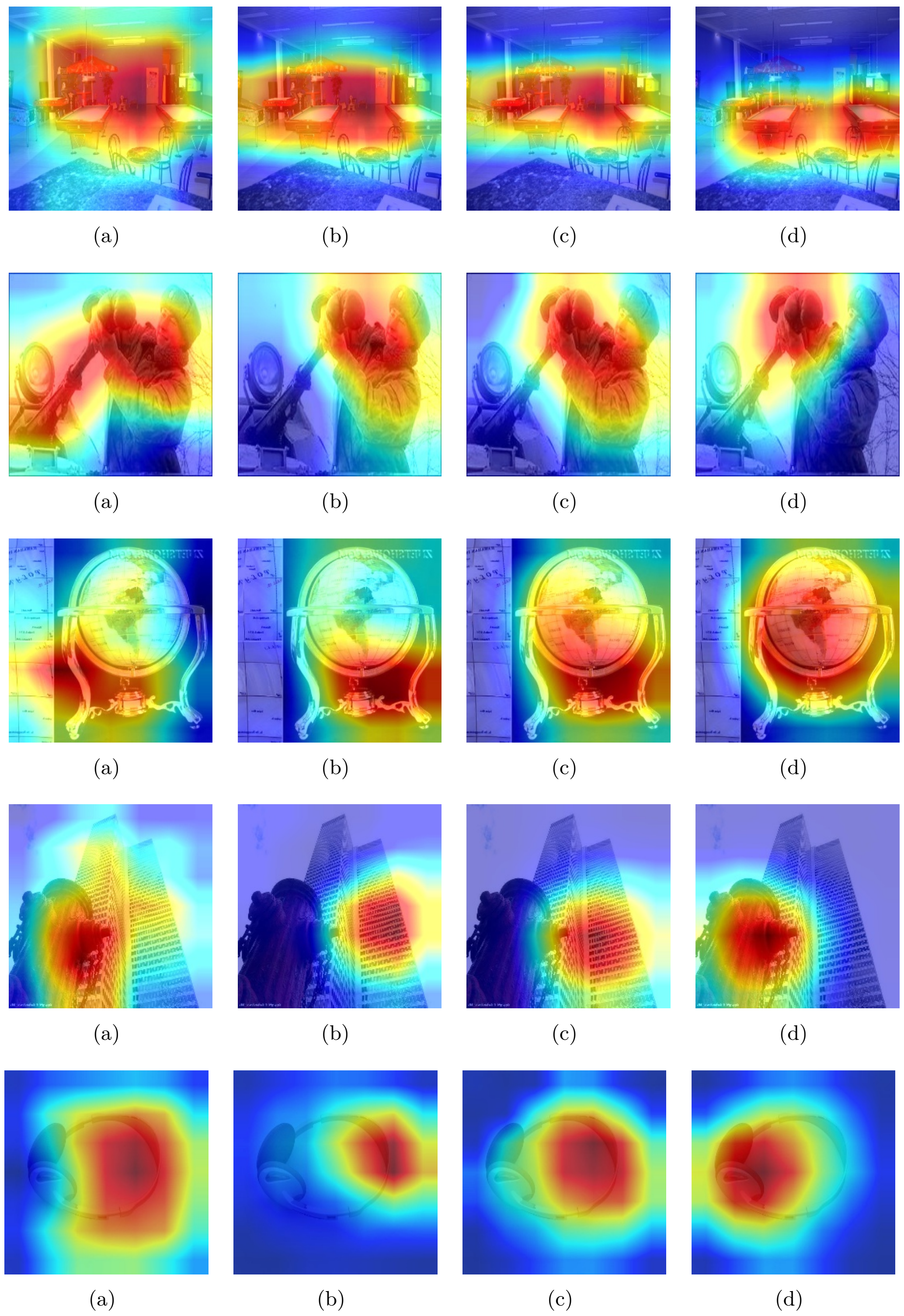


Figure 4.3: Effect of the discriminative feature generation method on the Caltech-256 dataset: (a) original, (b) fine-tuning, (c) DELTA, and (d) DFG.

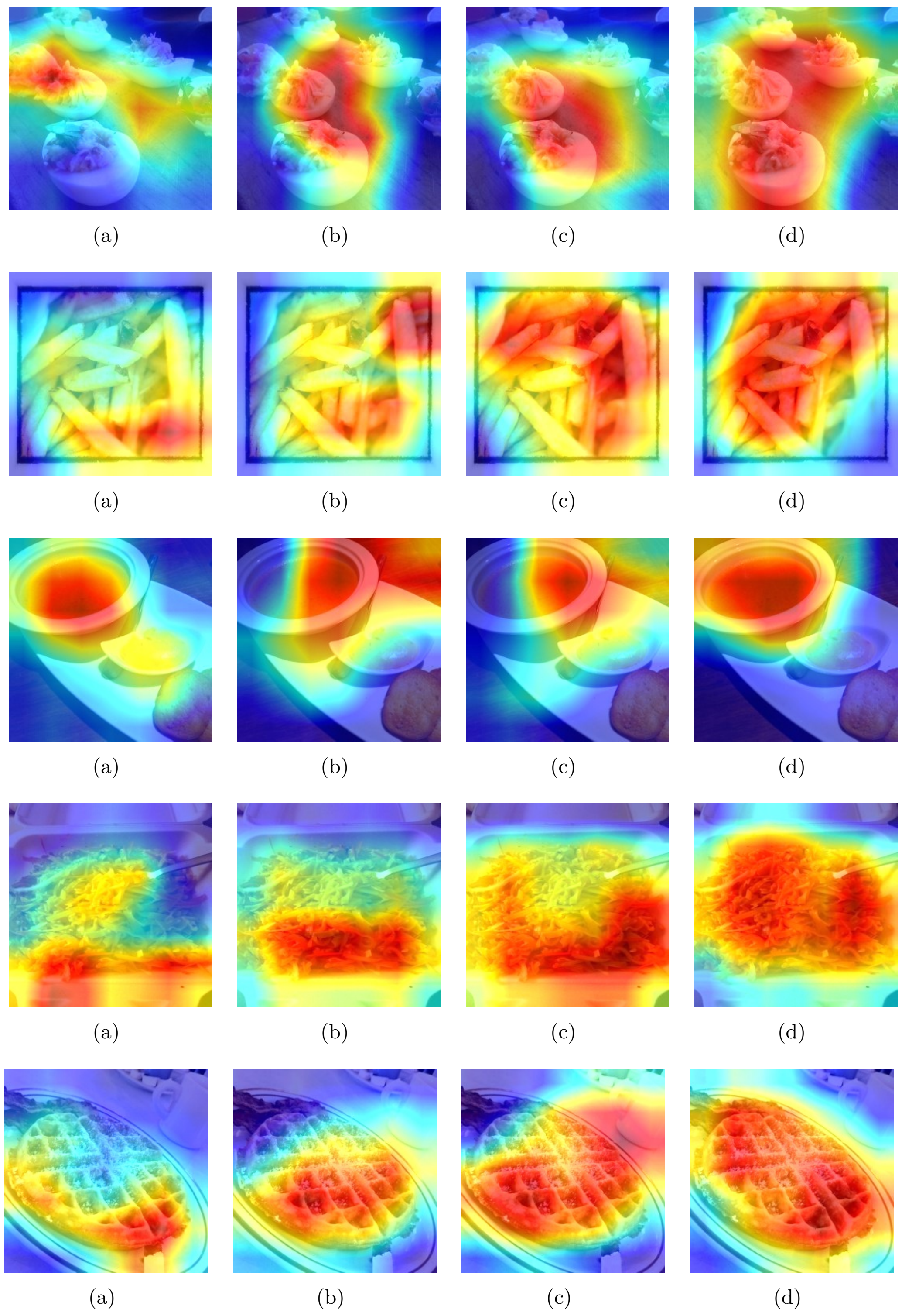


Figure 4.4: Effect of the discriminative feature generation method on the Food-101 dataset: (a) original, (b) fine-tuning, (c) DELTA, and (d) DFG.

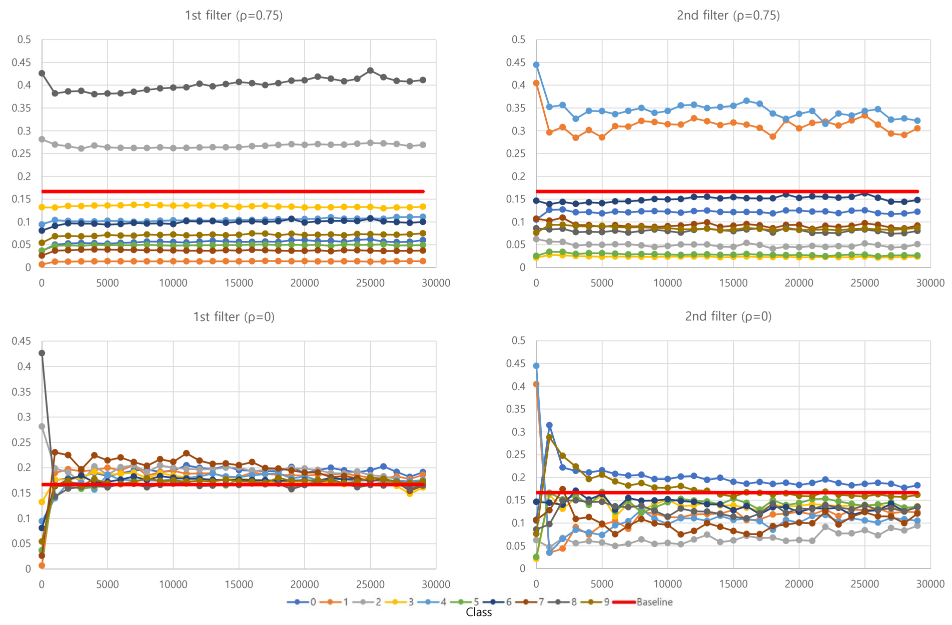


Figure 4.5: Distribution of the weights of two representative filters for the SVHN dataset. The red straight line is a baseline that indicates the equality of all filters.

each class label.

Figure 4.6 shows the effect of hyperparameter ρ on the classification accuracy on the SVHN dataset using DFG. In this case, DFG with hyperparameter $\rho = 0.75$ outperformed that with other hyperparameters including the cases that the weights of the filters by the feature extractor only trained on the source or the target dataset. The test accuracy varies smoothly according to hyperparameter ρ, and the combination of the two different weights of a filter yields better accuracy.

4.3 Conclusions

In this chapter, we proposed a novel DFG method using attention maps in the feature space. The structure of the feature classifier network of the independent network was employed in the proposed GAN structure. To generate discriminative features, we adopted supervised attention mechanisms for each class label and applied the mechanisms to the feature generator network for the filter weights and feature activation. The experimental results showed that the proposed DFG method improved the accuracy of the classification models of LeNet-5, VGG-16, and ResNet-50 by 4.24%, 5.52%, 14.73%, 13.95%, 42.99%, and 45.83% on six different benchmark datasets, respectively. Compared to the transfer learning method, DELTA, and the other adversarial feature augmentation methods, the proposed DFG method provided the

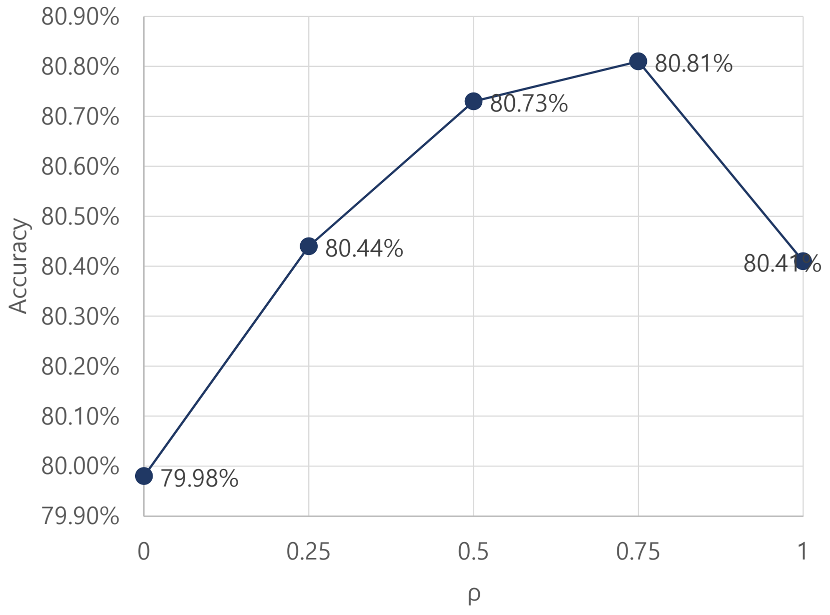


Figure 4.6: Classification accuracy on SVHN using DFG by changing hyperparameter ρ.

largest improvement in classification performance. Furthermore, we visualized the distribution of generated features by applying PCA and t-SNE and analyzed the activation maps from the proposed method using class activation maps. The qualitative results showed that the features generated by the proposed method are distributed similarly to the real features and the proposed method improved the concentration and activation degrees in the class activation maps.

In the future, we will extend the proposed method to real-world problems and contemplate further accuracy improvement in the feature generation method. Although the proposed method improved the performance of the classification under the class-imbalanced conditions, we believe there is room for improvement as a gap still exists between our method and the balanced condition.

5 Application: Generative Oversampling Method on Bearing Fault Detection and Diagnosis

5.1 Problem Definition

This chapter presents the approaches to predict the remaining useful life (RUL) and detect faults of the bearings used in machinery. To predict the RUL or detect faults of the bearings, it is necessary to extract features from the raw vibration signal. For the RUL prediction, the proposed method is composed of three stages: 1) feature extraction by transforming the raw vibration signal into a nested scatter plot (NSP) image, 2) the health stage (HS) division by using CNN, 3) the RUL prediction by using CNN-LSTM. In this study, two different feature extraction methods are introduced: One is an image transformation method with Hilbert–Huang transformation (HHT) and Fast Fourier transform (FFT), and another is a generalized multiscale feature extraction method with the GAN scheme. For the fault detection and diagnosis (FDD), a CNN-based classifier with the imaging method for vibration signal and an oversampling method with a generative model to improve the performance when a dataset is imbalanced between normal and faulty conditions are introduced in this study. The methods presented in this chapter have been published in Sensors [77] and Applied Sciences [53]. The generalized multiscale feature extraction method with the GAN scheme is available online [78].

5.1.1 Health Stage Division and Remaining Useful Life Prediction of Bearing

Bearings are widely employed in various mechatronic systems, such as induction motors, turbines, aircraft machines, and vehicles. More than 50 % of the failures in induction motors are caused due to the degradation of the bearings [7]. Thus, it is necessary to predict the future health conditions of the bearing to prevent catastrophic accidents by the corresponding maintenance.

Prognostics and health management (PHM) technology collects status information from industrial systems, such as manufacturing machines, facilities, and power plants, to detect failures of the system and enables maintenance schedule in advance by predicting the point of failure through analysis and predictive verification [8]. One of the PHM technologies predicts the RUL of the rolling element bearings to prevent unexpected failures and improve reliability [9, 10]. A large number of studies have been conducted for RUL prediction, which can be categorized into model-based [11] and data-driven methods [13]. Model-based methods estimate the RUL through analytical models based on physical laws and mathematical functions. These methods include physics-based methods, empirical-based methods [8], Kalman filter [17], and particle filter [18]. However, they require expert domain knowledge, and to build a precise model in the industrial system has become increasingly complex [19]. Recently, data-driven methods have garnered significant interest along with substantial development in machine learning and minimal requirement of expert knowledge. The data-driven methods capture the direct relationship between collected machine data and degradation status with machine learning techniques.

Recently, deep-learning-based RUL prediction methods have been proposed and have achieved better prediction performance than conventional data-driven methods [137, 138, 139, 140, 141, 142, 143, 144]. Although these deep-learning-based RUL prediction methods have been successfully developed, less importance has been given to three challenging issues:

(1) These methods require domain knowledge to extract features or we have to manually specify the feature types. The various time-domain statistical features, such as signal root mean square (RMS), crest factor, kurtosis, and entropy, and the frequency-domain features, such as FFT, HHT, and wavelet transformation, were extracted and used to predict the RUL [137, 138, 139, 143]. In other words, only one time-domain feature or time-frequency feature by wavelet transformation [141], short-time Fourier transform (STFT) [142], or RMS [144] was used to predict the RUL. The RUL prediction can remove the feature extraction by simply taking raw data as input to the deep neural networks (DNNs) [140], but it results in high computation cost. Even worse, such a method cannot correct the training error, if the extracted feature is used to predict RUL with the long short-term memory (LSTM) neural networks.

(2) The aforementioned data-driven methods assume that training and test data are collected by the same sensors under the same operating conditions or are from the same distribution. However, these assumptions can be impractical in industries, since machinery working conditions usually change with respect to tasks, and the training and test data can be collected from different entities. Though the machinery working conditions, such as motor power and rotating speed, and degradation generation conditions, such as radial force, are identical in an experimental platform to validate bearings fault detection, the bearing faults usually vary. Generally, the degradation trends are not the same for different bearings under the same conditions, resulting in significant data distribution discrepancies between entities. To overcome these problems, adversarial domain adaptation approaches and CNN-based health stage prediction methods have been proposed for RUL prediction [145, 146, 147, 148]. However, those approaches only focus on the single-source single-target adaptation setting. In real-world applications, the training data may come from multiple domains with different distributions so that it is expensive in both time and cost to train the network of domain adaptation for every condition and bearing.

(3) For accurate RUL prediction, it is essential to properly determine the HS

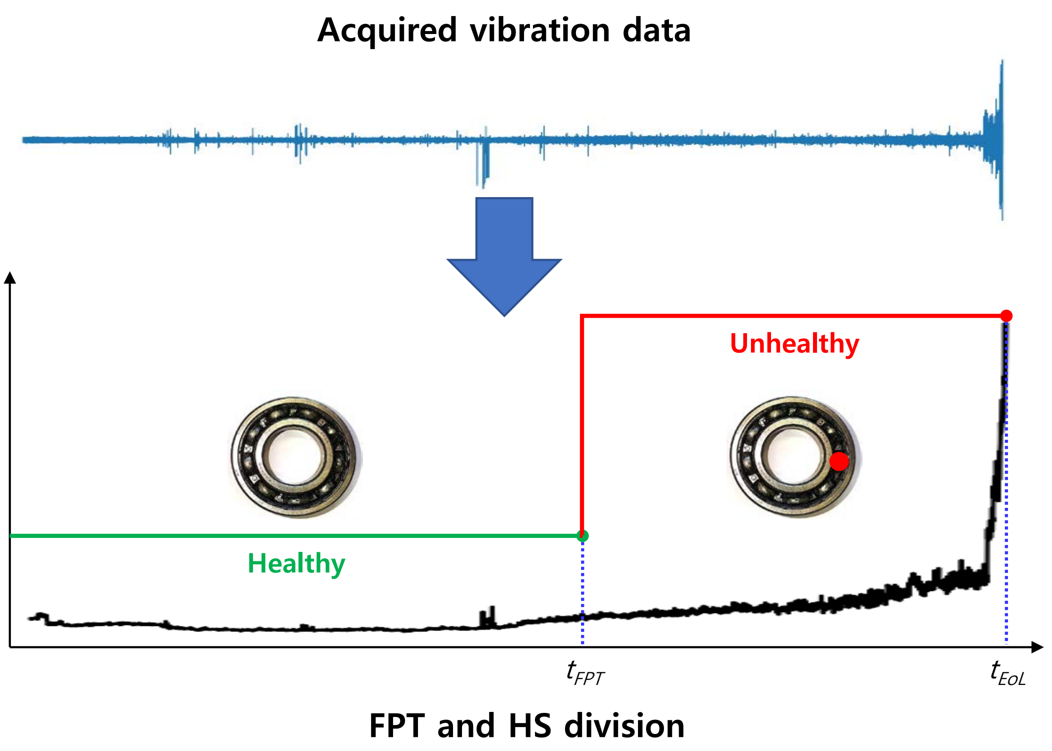


Figure 5.1: Degradation process with first predicting time (FPT) and health stage (HS) division

of the machinery; because the machine in a healthy state would not have any remarkable difference in the run-to-failure training dataset. However, extant conventional methods predict the RUL without determining the first predicting time (FPT) [145, 146, 147], which is the start time of the unhealthy stage. As shown in Figure 5.1, the continuous degradation process is divided into two HSs by determining FPT. The FPT can be determined using conventional features, such as kurtosis and RMS [142, 144]. Although unsupervised methods [149], such as auto-encoder, promise the extraction of efficient health indicators(HI); however, they possess a challenge assigning an appropriate threshold for determining FPT. Recently, the CNN-based HS division methods have been proposed, but they require an appropriate threshold to determine the FPT [148].

In this study, we propose a supervised data-driven method for HS division, that avoids usage of a pre-specified HI as well as the employment of pre-specified thresholds. The principle idea involves supervised training of a binary regression model using the early stage (nominal functioning) samples and last stage (near to complete failure) samples from a run-to-failure bearing degradation database. Unlike conventional HI methods, there is no need to set the threshold and it only needs a small number of labels. The laborious task of labeling, which is typically associated with supervised methods, is also avoided as a simple distinction is made between the early and last stages of component degradation containing relatively smaller sets of data to be labeled. The proposed method has two phases. Phase one consists of transforming the time series data of raw vibration data into an NSP image for feature extraction. In phase two, these images are used to train a binary regression model using CNN.

Additionally, we propose a generalized multiscale feature extraction method for the RUL prediction. GAN is used to learn the distributions of multiple training data from different bearings and extract domain-invariant generalized prognostic features. For the determination of FPT and RUL prediction, the proposed feature extraction method consists of two steps; the first of which trains multiscale adversarial neural networks to reconstruct the vibration input signals to the generalized prognostic features. Here, three different levels of a one-dimensional U-Net [150] architecture are trained to minimize the loss functions of the GAN-based proposed feature extraction method; The second step transforms the generalized features into an NSP image to determine the FPT and to predict the RUL. A CNN-based binary regression model determines the FPT, and a CNN-LSTM model predicts the RUL.

5.1.2 Bearing Fault Detection and Diagnosis under Imbalanced Data Condition

Traditional FDD methods were developed using physical models based on mathematics and mechatronics [14, 15, 16]. Those methods require complex analysis steps with domain knowledge. In addition, because physical models are highly dependent on individual specifications, user configuration is needed when the model is used in a specific facility. To overcome the problems of physical models, assorted data-driven FDD methods that use machine learning and statistics, such as support vector machine [20, 21] and fuzzy logic [22], have been proposed. However, these data-driven FDDs still require a complicated pre-processing step before training the models.

Recently, DNN with more powerful fitting abilities have been developed and widely applied to prognostics and health management. In [23, 24, 25, 26, 27], time-domain and frequency-domain features are extracted in data processing, and then an FDD model is applied for motor status classification. In [28, 29], vibration image generation with signal analysis was utilized for feature extractions. The feature extraction method itself can be automated using these approaches, but establishing such models requires complicated signal processing steps. Automatic feature extraction using an auto-encoder has been proposed [30], but the computation cost of the DNN model itself is quite high. To realize the wide adoption of data-driven FDD to industry, simpler and more efficient methods are required in both data-processing and DNN models.

Furthermore, these data-driven methods have shown great performance with low domain knowledge requirements, but problems related to the quantity and quality of data still remain. Data imbalance is common in FDD because normal condition data are more common than faulty condition data in real manufacturing environments [31]. Such imbalanced conditions degrade data-driven FDD, especially for CNN-based classifiers. Among oversampling [24, 32], down-sampling [33], and ensemble learning [34], all of which have been proposed to solve the data imbalance issue, oversampling is most suitable for industrial FDD because of severe data imbalance ratios. Additionally, oversampling is always the most effective way to deal with class imbalance for the CNN model on image classification [5].

In this study, we investigated the data-driven FDD of bearing faults in an induction motor under data imbalance conditions. Firstly, a CNN-based classifier with an imaging method for vibration signals was proposed. In NSP, the correlated time-series data are represented by a square matrix, similar to a scatter plot, where the elements of density dots are calculated like a heat-map. Experimental evaluation using measured vibration signals from a test bench confirmed that the features of bearing faults are easily extracted using NSP and the CNN-classifier.

Secondly, an oversampling method with a generative model was proposed to improve performance when a dataset is imbalanced between normal and faulty conditions. The generative model was developed based on a WGAN-GP [71] and deep convolutional generative adversarial networks (DCGAN) [107]. We evaluated the performance improvement of the CNN classifier for various conditions after applying our oversampling method to faulty condition data.

5.2 Deep Representation Feature Extraction and Image Transformation

5.2.1 Image Transformation of Vibration Signals

NSP is a data wrangling method that transforms correlated time series data into an image for multi-variate correlation analysis [53, 77]. It is a matrix representation similar to the heat map of the quantized value of time series data. Each vibration signal is quantized and mapped into nested clusters. The cumulative number of signal values in the nested cluster is normalized in order to represent the density of the nested cluster. Although NSP removes the non-stationarity of the time sequence, it is an efficient imaging method for multi-variable correlation analysis [53].

By using NSP, we transform multi-channel vibration signals into a single fixed-size image. Continuous multi-channel vibration signals are split into given intervals. A three-step approach is used: feature extraction using bandpass filters, decomposition of the nested clusters in each bandwidth, and aggregation of the decomposed sections into a single RGB image. As represented in Figure 5.2, at least two different data channels as data sources are required and the first step is the incorporation extraction of signals in three different bandwidths. HHT and FFT are used to determine the bandwidth of bandpass filters [53]. In the second step, the extracted two-channel signals are compressed into nested clusters. Each channel signal is mapped on the x- and y-axis. Three different extracted signals are colored in red, green, and blue,

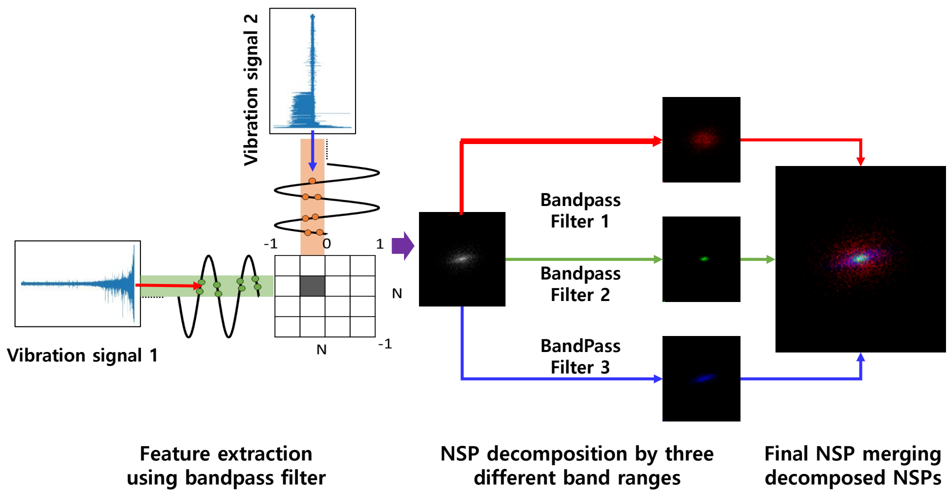

Vibration signal 2
Vibration signal 1
-1
0
1
N
N
-1
Bandpass Filter 1
Bandpass Filter 2
BandPass Filter 3
Feature extraction using bandpass filter
NSP decomposition by three different band ranges
Final NSP merging decomposed NSPs

Figure 5.2: Nested-scatter plot image (NSP) methods and application to vibration signals.

respectively. In the final stage, three scatter plots are aggregated together to form a single RGB image that is used as an input in our proposed method. An example of decomposition and merged NSP is shown in Figure 5.2.

5.2.2 HS Division Prediction Using CNN

By transforming continuous raw vibration time series data into images, we can now use our data as inputs to the CNN, which has proved its outstanding performance in computer vision [151]. The structure of the proposed convolutional neural networks for the HS division model (CNN-HS) is as follows. There are three convolution (Conv) layers with kernel sizes of 10 by 10, five by five, and three by three. The Conv layers are followed by two fully connected (FC) layers. A dropout rate of 50% was applied at the end of each FC layer. The final output layer is activated by a softmax function so that the obscurity during the interim period between the two stages can be quantified by a value between zero and one. The structure is shown in Table 5.1.

In the CNN architecture, the selectable hyperparameters are the number of filters in each Conv layer and the size of each FC layer. In the fault detection and diagnosis using NSP [53], the optimal numbers were found by varying the number of filters in each Conv layer. The size of the FC layer affects the expressiveness of the networks and the training time. Following [53], the size of each FC layer was determined as 500 and 50. In the experiments, the number of epochs during training was 30 and the validation data ratio was 1:9. An Adam optimizer was implemented with a learning rate of 0.001.

In a supervised manner, labeling of the target data is required for supervised learning to distinguish the features of the HS of a machine under different bearing

Table 5.1: The structure of the convolutional neural network (CNN)-health stage (HS).

Layers	Activation function	Dimension
Input	-	128 × 128 × 3
Conv 10 × 10	ReLU	60 × 60 × 20
Conv 5 × 5	ReLU	28 × 28 × 40
Conv 3 × 3	ReLU	24 × 24 × 20
Fully Connected	ReLU	500
Fully Connected	ReLU	50
Output	Softmax	2

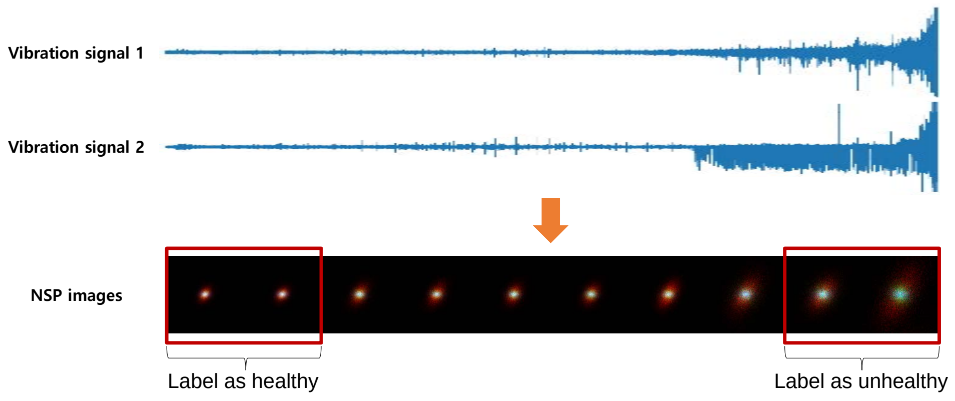


Figure 5.3: Binary labeling on part of training dataset on bearing wear experiment

degradation conditions. The training dataset is divided into two HS stages based on the time of acquisition. Figure 5.3 shows the division of our training dataset. The initial part, corresponding to the nominal functioning in the entire vibration dataset, is labeled as healthy, and the last part of the sample duration, when the bearing is damaged, is labeled as unhealthy. This is based on the assumption that degrading data of rotating machinery are obtained until the point of failure. By the supervised method, it is unnecessary to label all train datasets and analyze the dataset for labeling. To minimize labeling efforts on the entire dataset, the optimal sizes (small set) of the initial and last part, labeling 'healthy' and 'unhealthy', respectively, are evaluated.

Once training is over, the performance of the trained CNN-HS is evaluated in the testing phase. The trained CNN-HS was able to classify the NSP of an external test dataset, which was not used in training but operated under the same conditions as the rest of the training datasets. The binary regression results of the whole run-to-failure are computed, in which there is no need to specify a threshold value. As the trained CNN-HS learns the differences in the features of the NSP for healthy and unhealthy data labeled in part of the training datasets, it can recognize the degra-

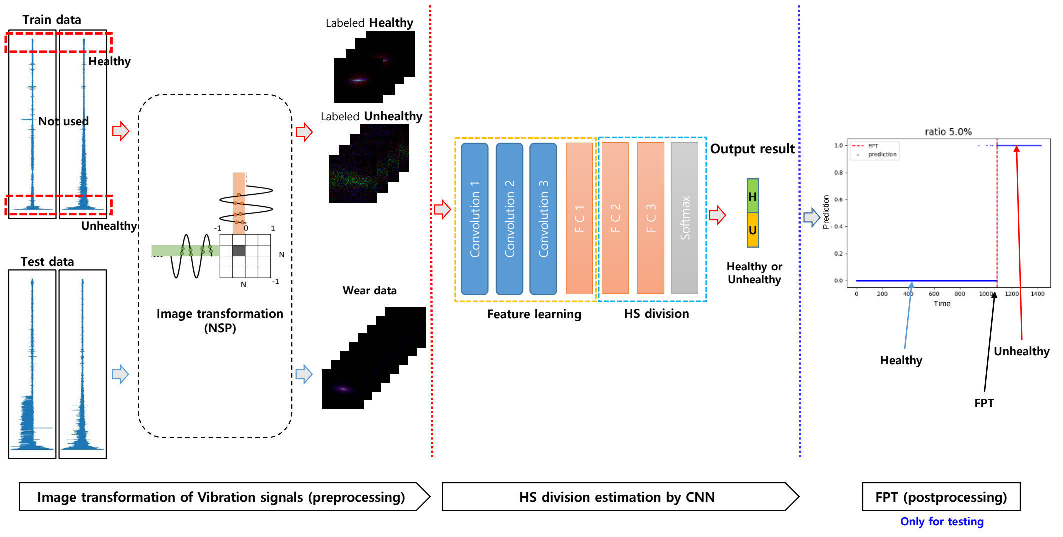


Figure 5.4: CNN-HS combined with NSP.

dation pattern of all of the data. For HS division, it is important to determine FPT, which is a starting point of the deterioration of bearings, in terms of maintenance. However, it is difficult to determine FPT since the features of the deterioration at FPT are weaker than the features in the unhealthy stage.

To determine FPT, we implemented a simple continuous trigger mechanism in which the machinery was considered to be in an unhealthy stage when a certain number of consecutive values of one (unhealthy) were given out by our model. This trigger mechanism prevents unnecessary oscillation and uncertainty to determine earlier FPT. The trigger mechanism has generally adopted an HS division strategy [9, 152, 153]. In the experiments, the threshold number of consecutive estimations of unhealthy is set to three based on the analysis of signals in order to prevent false alarms during the normal stages. The overview and flowchart of CNN-HS are shown in Figure 5.4.

5.2.3 Generalized Multiscale Feature Extraction

In this section, we introduce a generalized multiscale feature extraction method based on the GAN scheme and U-Net architecture for the HS division and the RUL prediction. Figure 5.5 presents the overall framework with the detailed steps of the learning procedure of the proposed feature extraction model. The GAN structure is a kind of adaptation of classification enhancement GAN (CEGAN) [70]. We define three types of independent networks in the proposed GAN scheme: 1) the multiscale feature extractor as a generator network, 2) a discriminator to separate the features generated by the generator from the real input data and to distinguish the data from different domains, 3) the CNN-based HS division and the LSTM-based RUL

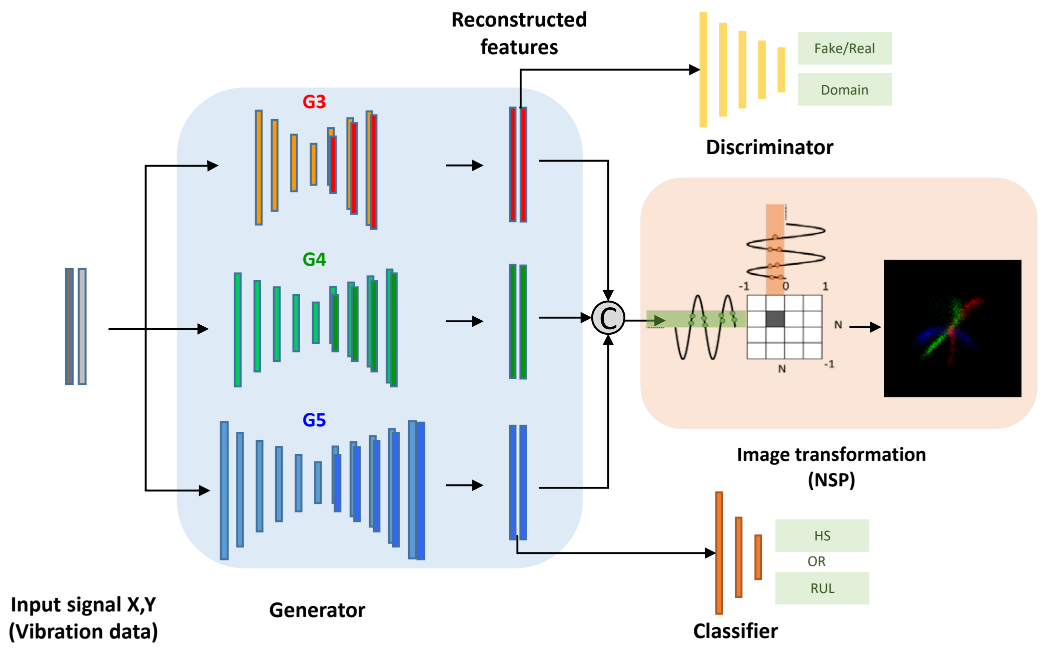


Figure 5.5: The structure of the proposed framework for the generalized multiscale feature extraction.

predictor as the classifier network.

Generator

The generator is composed of three multiscale generators as shown in Figure 5.5. The proposed generator adopts basic concepts from U-Net [150] for one-dimensional(1D) time-series feature reconstruction by substituting the original 2D convolutions with 1D operations. U-Net [150] was invented for image segmentation, concatenating feature maps from different levels to improve segmentation performance and it combines low-level detail information with high-level semantic information. It has achieved promising performance on various image segmentation problems. Based on CNN, the U-Net architecture composes of an encoder, a decoder, and skipped connections between the encoder and the decoder. The encoder is composed of several levels of convolutional operations, which extract increasingly abstract representations of the input data, and a downsampling block to reduce input complexity. The convolutional operations followed by max-pooling downsampling are applied to encode the input image into feature representations at multiple different levels. The decoder block is also composed of several levels of convolutional operations, upsampling blocks, and concatenation blocks. The decoder block semantically projects the discriminative features learned by the encoder onto higher resolution image space to get a dense

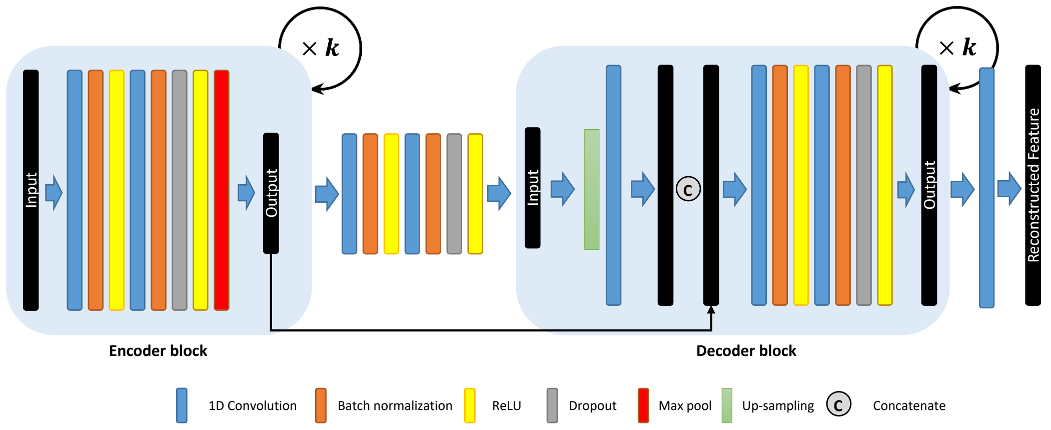


Figure 5.6: Structure of the generator based on U-Net.

classification. Unlike the general usage of U-Net for image segmentation tasks, we found that the structure of the U-Net enables the network to consider the deep and the shallow features at the same time and improve the effectiveness of the feature extraction. Thus, it helps to reconstruct the features to satisfy the conditions we designed. Whereas Gaussian noise is input to the generator in the general GAN, we provide noise only in the form of dropout applied on several layers of the generator. The architecture of the generators is shown in Figure 5.6. The generator consists of the encoder and the decoder which process the input data. The three multiscale generators have a different number of encoder and decoder blocks: 1) three encoder-decoder blocks, 2) four encoder-decoder blocks, 3) five encoder-decoder blocks.

Discriminator

The proposed GAN was developed for purpose of domain adaptation from multiple sources. Generally, the discriminator is trained to distinguish generated and real data in GAN. However, our discriminator also has a functionality of domain discriminator which distinguishes the data domain. It is because the proposed framework for the generalized multiscale feature extraction aims to extract the features of bearing wear from multiple domains. The adversarial training enables the generator to extract high-level features containing domain-unbiased information, and makes it hard for the discriminator to classify only real or fake (denoted as a degree) as well as source domain. To do so, the convolutional layer of the discriminator extract the features of input (reconstructed features of real and fake data from the generator) with a Leaky ReLU, and then the two separated linear layers output the true degree and the domain classification, respectively (shown in Figure 5.7).

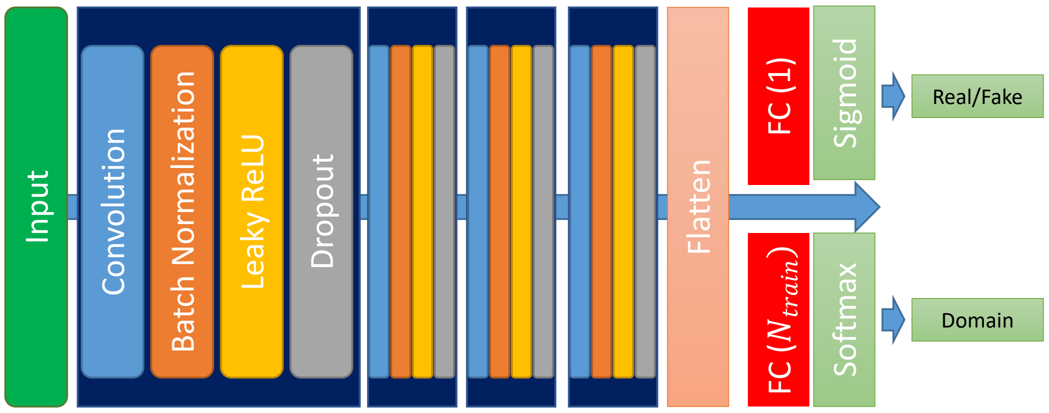


Figure 5.7: Structure of the discriminator.

Classifier

To improve the performance of the HS division and the RUL prediction, we employ the structure of the classifier network to the independent network in the proposed GAN structure. In Figure 5.5, the first stage involves a generalized multiscale feature extraction G_{HS} for the HS division and the HS division model C_{RUL} with the CNN and the transformed NSP images from the extracted features. In the second stage, a generalized multiscale feature extraction model G_{RUL} for the RUL prediction is trained and a CNN-LSTM model C_{RUL} predicts the RUL with the transformed NSP images from the extracted features.

Loss functions and training procedure

In this study, it is assumed that the vibration signals are collected by two high-frequency vibration sensors placed horizontally and vertically on each bearing, and a number of N_{train} run-to-failure vibration data in the whole life cycle from different bearings can be used to train the proposed DNNs. Suppose that $X_j = \{x_j^i\}_{i=1}^{n_j} \in \mathcal{R}^{N_{samples}}, j = 1, 2, ..., N_{train}$ denotes n_j sequential training samples from the jth bearing, where $N_{samples}$ is the dimension of the sample. In each bearing data, two samples, each from horizontal and vertical vibration sensors, exist.

We aim to extract generalized prognostic features to improve the performance of the HS division and the RUL prediction. In the HS division, labeling of the target data is necessary for supervised learning to distinguish the features of the HS of a machine under different bearing degradation conditions. The training dataset can be divided into two health stages based on the time of acquisition. The initial part, corresponding to the nominal functioning in the entire vibration dataset, is labeled as healthy, and the last part of the sample duration, when the bearing is damaged, is labeled as unhealthy. This is on the assumption that degrading data of rotating machinery are obtained until the point of failure. This manner not only reduces the effort of labeling but also improves the prediction of HS division when no ground truth of HS is available (most of the open dataset has no information about HS). The corresponding HS labels are expressed as follows.

$$y_{HS}{}_j^i = \begin{cases} 0, & \text{if } i < n_j \times p \\ 1, & \text{if } i > n_j \times (1-p) \end{cases} \tag{5.1}$$

where $y_{HS}{}_j^i$ represents the HS label of x_j^i and p is the percentage of the total degradation process. By the CNN-based binary regression model, the HS and the FPT of the jth bearing, $T_{FPT}{}_j$, can be determined.

Next, the RUL labels are expressed as follows.

$$y_{RUL}{}_j^i = \frac{n_j - i}{n_j - T_{FPT}{}_j} \tag{5.2}$$

where $y_{RUL}{}_j^i$ denotes the RUL label of x_j^i.

The generator network is decomposed into two sub-generators: G_{HS} and G_{RUL}. The two sub-generators correspond to discriminators D_{HS} and D_{RUL} with the same structure. For the stability of the training procedure and the quality of the generated data, we apply the WGAN-GP to the objective function to guide the training process. The objective functions are defined as follows.

$$\begin{aligned} \mathbb{L}_D(x, d; \theta_D) = &- \mathbb{E}_x[D_R(x)] + \mathbb{E}_x[D_R(G(x))] \\ &+ \alpha\, \mathbb{E}_{\hat{x}\sim P_{\hat{x}}}[(\|\nabla_{\hat{x}} D_R(\hat{x})\|_2 - 1)^2] + \lambda_D\, \mathbb{L}_{CE}(D(x), d), \end{aligned} \tag{5.3}$$

$$\begin{aligned} \mathbb{L}_G(x, d, y; \theta_G) = \mathbb{E}_x[D_R(G(x))] + \lambda_G\, \mathbb{L}_C(G(x), y) \\ - \lambda_D\, \mathbb{L}_{CE}(D_D(x), d), \end{aligned} \tag{5.4}$$

$$\mathbb{L}_C(x, y; \theta_C) = \begin{cases} \mathbb{L}_{C_{HS}}(G(x), y), & \text{for the HS division} \\ \mathbb{L}_{C_{RUL}}(G(x), y), & \text{for the RUL prediction} \end{cases} \tag{5.5}$$

where x is a sequential training sample, y is the HS label in the case of the HS division and the RUL label in the case of the RUL prediction, d is the associated label of the domain, λ_D is a hyperparameter that controls the effect of domain generalization, λ_G controls the relative importance of different loss terms, $\mathbb{L}_{CE}$ denotes the standard cross-entropy loss function, and θ_D, θ_G, and θ_C are parameters of the discriminator, the generator, and the classifier, respectively. Both the real data and generated data are entered into the discriminator, while the data reality and the domain classification are the output of the discriminator D_R and D_D, respectively. Whereas the discriminator D is trained to minimize $\mathbb{L}_D$ for distinguishing the data from different domains and between the real and the generated data simultaneously, the generator G is trained to minimize $\mathbb{L}_G$. Furthermore, the classifier C, which is the CNN-based HS division C_{HS} or the LSTM-based RUL predictor C_{RUL}, is trained to classify the HS or predict the RUL. In other words, the generator reconstructs the generalized features to fool the discriminator and to improve the performance of the HS division and the RUL prediction. Additionally, by reconstructing features that are similar to the real data, feature values can be generated within a certain range, and the time-series properties can be maintained. Finally, the classifier losses for the HS division and the RUL prediction are defined as follows.

$$\begin{aligned}\mathbb{L}_{C_{HS}}(G(x), y) =& \mathbb{L}_{BCE}(G(x), y),\\ \mathbb{L}_{C_{RUL}}(G(x), y) =& \lambda_{MAE}\,\mathbb{L}_{MAE}(G(x), y)+\\ & \lambda_{RMSE}\,\mathbb{L}_{RMSE}(G(x), y)+\\ & \lambda_{MAPE}\,\mathbb{L}_{MAPE}(G(x), y)\end{aligned} \tag{5.6}$$

$$MAE = \frac{1}{N_S}\sum_{i=1}^{N_S}|ActRUL_i - PreRUL_i| \tag{5.7}$$

$$MAE = \sqrt{\frac{1}{N_S}\sum_{i=1}^{N_S}(ActRUL_i - PreRUL_i)^2} \tag{5.8}$$

$$MAPE = \frac{1}{N_S}\sum_{i=1}^{N_S}\frac{|ActRUL_i - PreRUL_i|}{ActRUL_i} \tag{5.9}$$

where $\mathbb{L}_{BCE}$ denotes the standard binary cross-entropy loss, $\mathbb{L}_{MAE}$ is mean absolute error (MAE) in (5.7), $\mathbb{L}_{RMSE}$ is root mean square error (RMSE) in (5.8), and $\mathbb{L}_{MAPE}$ is mean absolute percentage error (MAPE) in (5.9), N_S is the number of testing samples, and $ActRUL_i$ and $PreRUL_i$ are the actual RUL and the predicted RUL values corresponding to the ith testing sample, respectively.

Three different reconstructed features are transformed into three channels of an NSP image by the following two steps. The first step is to compress the reconstructed two-channel features into nested clusters. Each channel feature is mapped onto the x- and y-axis and three different reconstructed features from the three multiscale generators are colored in red, green, and blue, respectively. To represent the intensity of each cluster from the reconstructed features, the mapped values count was translated into pixel intensity. In the second step, three scatter plots are aggregated together to form a single RGB image as shown in Figure 5.5.

The training details for the proposed multiscale feature extraction method are summarized in Algorithm 3. $X_j = \{x_j^i\}_{i=1}^{n_j} \in \mathcal{R}^{N_{samples}\times 2}, j = 1, 2, ..., N_{train}$ denotes n_j sequential training samples from the jth bearing, where $N_{samples}$ is the dimension of the sample.

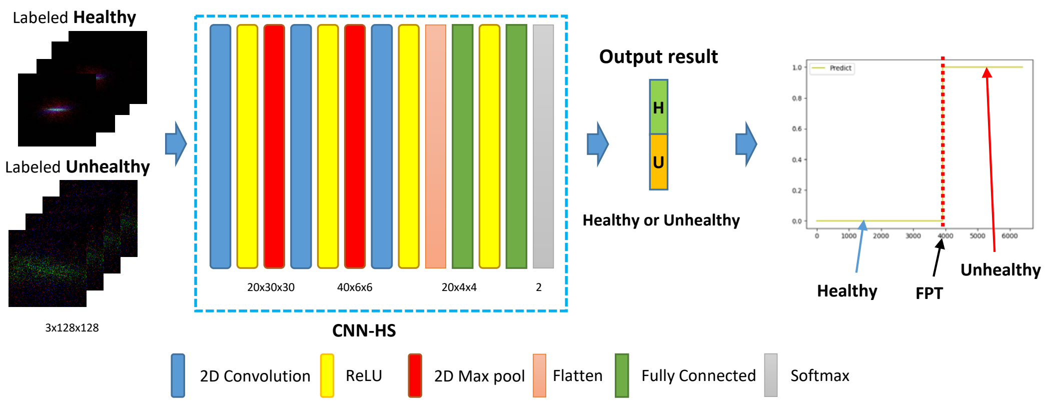
Algorithm 3 Training procedure for generalized multiscale feature extraction using adversarial networks. We use default values of $\alpha = 10$, $\lambda_D = 1$, $\lambda_G = 50$

```
Require: Batch size $m$, Adam hyperparameters $\eta$, hyperparameter for $\lambda_D$, $\lambda_G$.
 1: Input: $\{x_j^i\}_{i=1}^{n_j} \in \mathcal{R}^{N_{samples} \times 2}, j = 1, 2, ..., N_{train}$, $y$, $d$.
 2: for number of training iterations do
 3:     Sample $\{x_j^i\}_{i=1}^{m}$ a batch from the training dataset, corresponding HS or RUL
        label $\{y_j^i\}_{i=1}^{m}$, and corresponding domain label $\{d^i\}_{i=1}^{m}$.
 4:     for $k$ =3 to 5 do
 5:         Update discriminator $D$ by descending the gradient of (5.3) with $G_k$:
 6:         $\theta_D \leftarrow \theta_D - \eta_D \nabla_{\theta_D} \mathbb{L}_D(x, d; \theta_D)$
 7:         Update generator $G_k$ by descending the gradient of (5.4):
 8:         $\theta_{G_k} \leftarrow \theta_{G_k} - \eta_G \nabla_{\theta_{G_k}} \mathbb{L}_G(x, d, y; \theta_{G_k})$
 9:         Update classifier $C$ by descending the gradient of (5.5) and (5.6) with $G_k$:
10:         $\theta_C \leftarrow \theta_C - \eta_C \nabla_{\theta_C} \mathbb{L}_C(x, y; \theta_C)$
11:     end for
12: end for
13: for $k$ = 3 to 5 do
14:     Sample $\{x_j^i\}_{i=1}^{m}$ a batch from the dataset
15:     Transform $G_k(x)$ into nested clusters
16: end for
17: Merge three scatter plots to a single RGB image
```

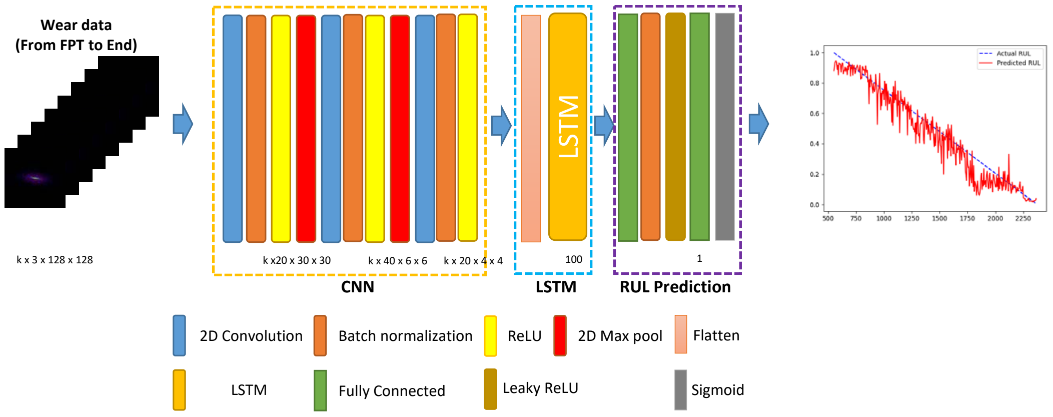
5.2.4 Health Stage Division and Remaining Useful Life Prediction with NSP

By reconstructing continuous raw vibration time-series data to multiscale features and transforming the features into NSP images, we can change the signal processing problem to an image classification problem and classify the images by using the CNN for the HS division model (CNN-HS) [77] and the CNN-LSTM for the RUL prediction model (CNN-LSTM-RUL). The structures of the proposed CNN-HS and CNN-LSTM-RUL are presented in Fig 5.8.

The trained CNN-HS and the trained CNN-LSTM can classify the NSP of an external test dataset, which was not used in training but operated under the same conditions as the rest of the training datasets. For the HS division, the binary regression results of the whole run-to-failure are computed by the CNN-HS, wherein specifying a threshold value is not necessary. As the trained CNN-HS learns the differences in the features of the NSP for healthy and unhealthy data in the training datasets, it can recognize the degradation pattern of all of the data. After determining the FPT using the CNN-HS, extracting the multiscale features, and transforming the features into the NSP images, the CNN-LSTM can recognize the degradation patterns of all of the data from the FPT to the end time. The CNN in the CNN-LSTM performs feature extraction from the transformed NSP images, and the LSTM and the RUL prediction part calculate the percentage of the RUL.



(a) CNN-HS



(b) CNN-LSTM

Figure 5.8: Proposed deep neural network architecture, (a) the proposed CNN-HS, (b) the proposed CNN-LSTM-RUL

5.3 Generative Model for Oversampling Fault Condition Data

5.3.1 Fault Classification Using CNN

Using NSP, solving FDD problems by time-series data analysis becomes an image recognition problem. We employ a CNN classifier because it provides outstanding performance in image recognition and classification [24]. The structure of the proposed CNN classifier, the CNN-based bearing fault detector (CBFD), is shown in Figure 5.9. There are three convolution (Conv) layers with kernel size 10×10, 5×5, and 3×3, respectively. Batch normalization was applied to the first Conv layer only. Two fully connected (FC) layers followed the Conv layers. Output nodes reflected fault types; there were four output nodes for the normal condition and three bearing

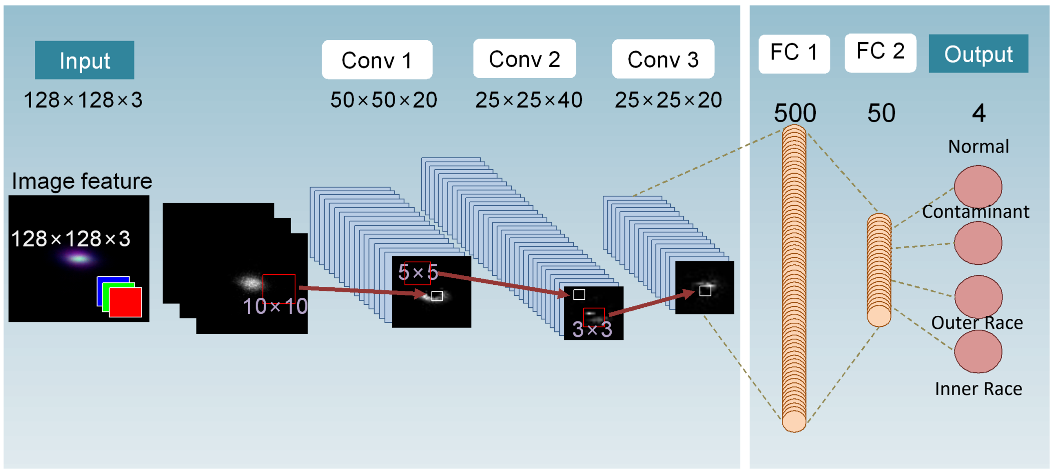


Figure 5.9: Convolutional neural network (CNN)-based bearing fault detector (CBFD) architecture.

fault types (inner race, outer race, and contaminant fault). The activation functions for all layers were ReLU.

In the CNN architecture, the selectable hyperparameters were the number of filters in each Conv layer, the size of each FC layer, and the dropout rate. To determine the hyper parameters, the criteria for the fault classifier under data imbalanced condition was defined as accuracy over 95% for test sets in mild imbalance ratio less than 20:1 and high accuracy even in cases of severe imbalance ratio over 50:1.

We tested varying the number of filters in each Conv layer and found the optimal numbers that met the criteria of training and test accuracy. The results of the number of filters should range from 10 to 50 because underfitting occurred when the number of filters was less than 10 and overfitting occurred when it was more than 50. The size of the FC layer affected the expressiveness of the networks and the training time. A large FC layer size increased the risk of overfitting and training time but could increase the accuracy. Through our experimentation, two FC layers of sizes less than 200 and 20 could not meet our performance criteria. Considering the overfitting and the training time, the optimal size of each FC layer was determined as 500 and 50. Finally, a dropout layer with 75% keep probability was applied to the first FC layer to mitigate overfitting in the training phase. The CBFD model is described in Figure 5.9.

5.3.2 Generative Model for Oversampling Fault Condition Data

The performance of FDD heavily depended on the designated frequency bands in NSP and the architecture of the CBFD model. However, here, we emphasize the training method used for the data-imbalance conditions.

CNN-based classification performs well when the distribution of classes is roughly balanced. However, faulty condition data are generally lower in volume than normal condition data [31]. Such imbalances cause lower recognition accuracy for the minor class, in this case, the faulty condition data. This phenomenon is important because the recognition rate of faulty conditions is the most important practical measure of effective FDD in engineering applications.

There are three solution types for data imbalance: oversampling [24, 32], down-sampling [33], and ensemble learning [34]. For extreme cases such as rare occurrence fault condition data, oversampling was the most suitable approach. In contrast, reducing the size of the entire dataset by down-sampling and modeling the minor data can cause a lack of data and make training itself impossible.

To solve the data-imbalance problem in FDD, we considered a generative oversampling method. Oversampling with generative adversarial networks (GANs) improved FDD accuracy in a previous study [24].

GANs [66] represent a class of generative models based on a game theory scenario in which a generator network G competes against an adversary D, a discriminator. DCGAN [107] is an extended model of GAN that uses de-convolution layers in the generator and convolution layers in the discriminator to extract features of images and construct a model to generate realistic fake images. By using DCGAN, NSP can be generated and used for oversampling. Figure 5.10 shows the DCGAN architecture employed in this study, in which we considered the convergence problem of DCGAN.

GANs aim to approximate the probability distribution function that the input data are assumed to be drawn from. In the original formulation of GANs [66], it was achieved by treating the discriminator as a binary classifier for real and fake data distributions. In this way, the discriminator provides meaningful gradients for the generator so that it minimizes the Jensen–Shannon (JS) divergence between the real and fake data distributions. However, this process is shown to be extremely unstable and difficult to train in practice. It has been shown that even in considerably simple scenarios the JS divergence does not supply useful gradients for the generator [93]. For this reason, numerous recent studies have focused on improving the stability and performance of GANs by enhancing the quality of the gradients derived from the discriminator.

To stabilize our generative model, we propose WGAN-GP [71] on the DCGAN architecture model (DCWGAN-GP). The original WGAN [93] exploited earth mover's distance (EMD) as a better means to measure the similarity between the two distributions. In this way, the losses of the discriminator and the generator correlate well with the output image quality. WGAN-GP utilizes the same distance measurement but ensures higher stability by penalizing the norm of the discriminator's output with respect to its input data. Our DCWGAN-GP is therefore resilient to the vanishing gradients problem and generates realistic fake images in a stable manner.

When input data are imbalanced, minor data are oversampled by the generative model until the desired ratio is met; at this point, we trained the CBFD model.

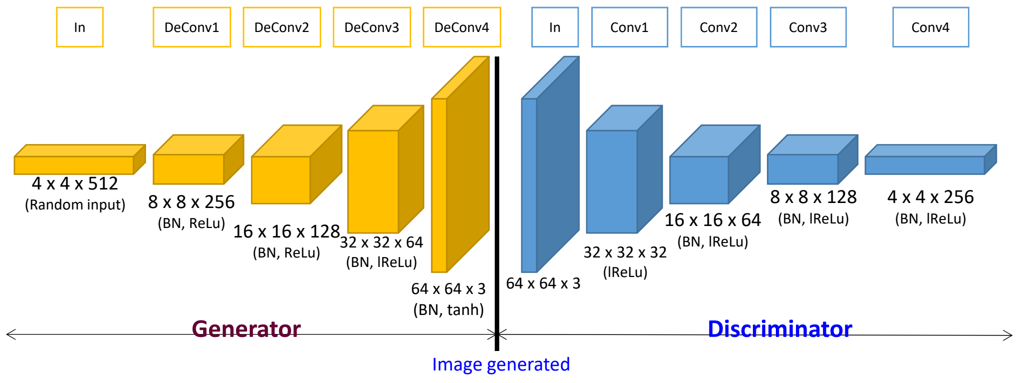


Figure 5.10: Deep convolutional generative adversarial networks (DCGAN) for generative model of NSP images.

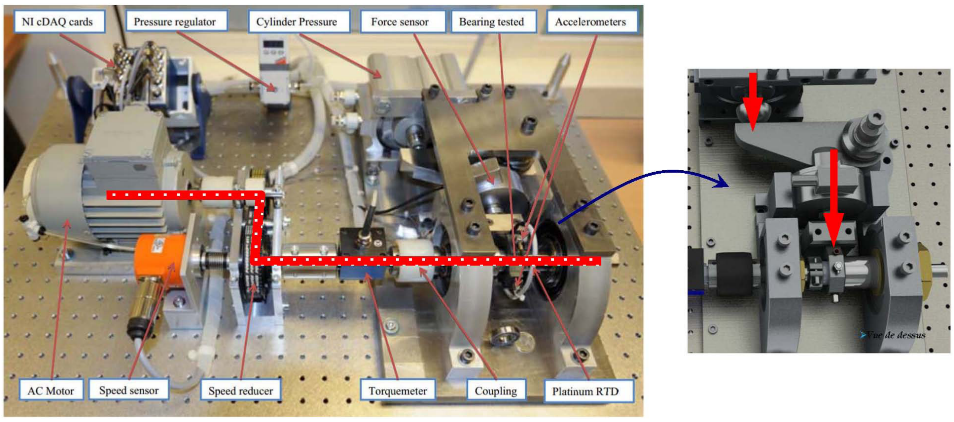
5.4 Experimental Results

5.4.1 Data Description and Preparation

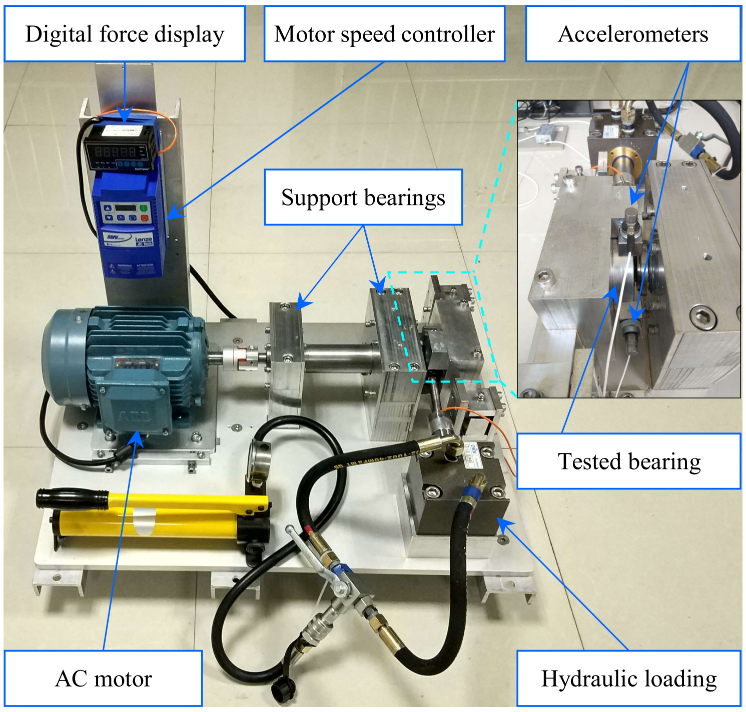
Run-to-failure datasets

The nature of this study needed continuous time series data on machinery that degraded over-time to the point of rolling bearing failure, including any failures in the balls, rings, and cage. The purpose of early fault detection is to detect the degradation as early as possible so that maintenance or prognostic-based decisions can be implemented. FPT is the indicator that detects the start of the degradation.

To evaluate the proposed method (denoted as GMFE) on the run-to-failure vibration dataset, two types of popular datasets were used: FEMTO [154] and XJTU-SY dataset [155]. The FEMTO dataset was collected by the PRONOSTIA test rig and has been available to the public since the IEEE PHM 2012 Prognostic Challenge (PHM 2012). The test rig mainly contains an asynchronous motor, a shaft, a speed controller, an assembly of two pulleys, and tested rolling ball bearings, which is shown in Figure 5.11(a). The vibration data in the horizontal direction were investigated. The dataset was composed of 17 run-to-failure data in which a single bearing was tested (two columns of vibration data: horizontal and vertical). The sampling frequency of data acquisition is 25.6 kHz and the sampling time length was 0.1 s, which was recorded per 10 s. The 17 datasets are grouped into three sections by operating conditions that differ in rotation speed and radial load. Each dataset has a different run-to-failure time, requiring fault detection methods that are adaptable to time-varying operational conditions and environments. When the amplitude of the vibration data exceeds 20 g, the run-to-failure experiments were stopped and the bearing is considered defective. More details about the test rig and the datasets can be referred to from [154]. Figure 5.12 shows the spectral density difference in bandwidths through FFT on the FEMTO dataset and the comparison between



(a) FEMTO dataset [154]



(b) XJTU-SY dataset [155]

Figure 5.11: Bearing test rigs (a) of the PRONOSTIA in the FEMTO dataset, (b) in the XJTU-SY dataset.

healthy and unhealthy data. The spectral density from 500 to 1200 Hz of unhealthy data was different from that of healthy data. Taking the sampling frequency into consideration, three different bandpass filters for NSP in 5.2.1 were used: 500–800 Hz, 800–900 Hz, 900–1200 Hz.

The XJTU-SY dataset contains complete run-to-failure data of 15 rolling element

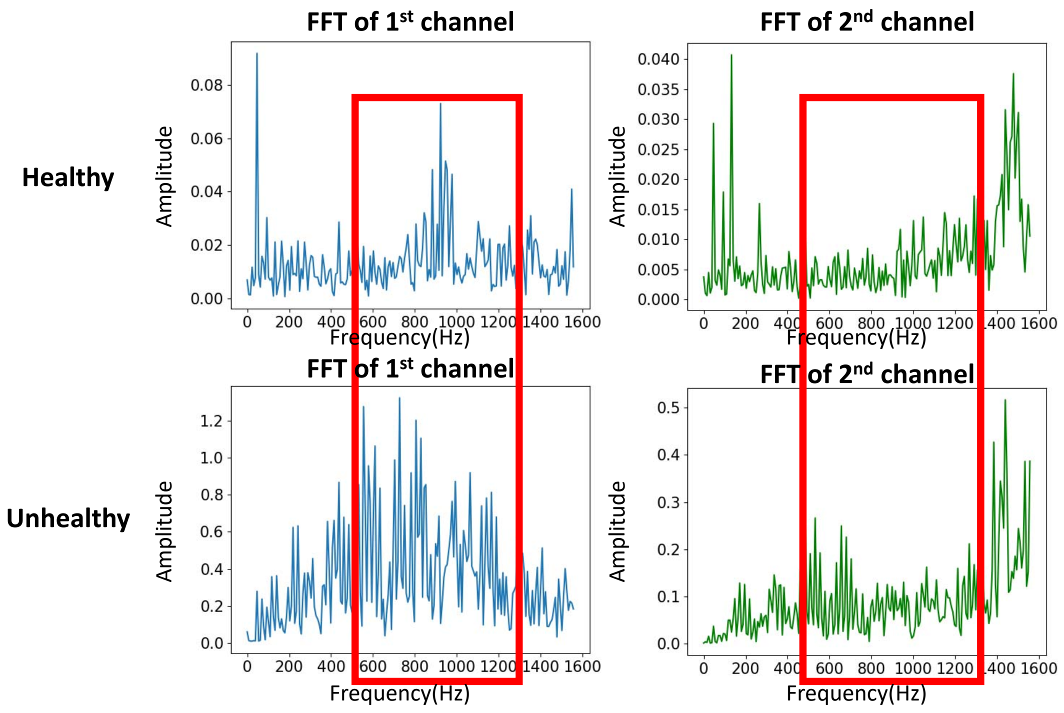


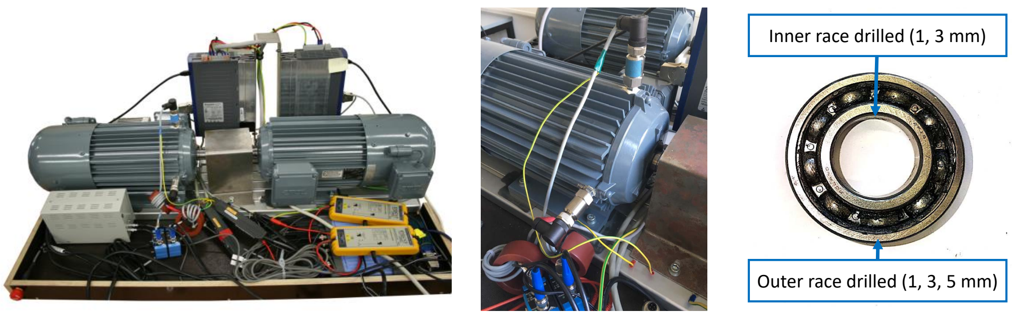
Figure 5.12: Spectral density difference in bandwidths through fast Fourier transform on Fanche-Comte Electronics Mechanics Thermal Science and Optics—Sciences and Technologies Institute (FEMTO) dataset.

bearings that were collected by the Xi'an Jiaotong University and the Changxing Sumyoung Technology [155]. The vibration signals were collected by two accelerometers placed at 90 degrees on the housing of the tested bearings, as shown in Figure 5.11(b). The sampling frequency is 25.6 kHz and the sampling time length was 1.28 s, which was recorded per minute. Like the FEMTO dataset, the accelerated degradation bearings tests are stopped when the amplitude of the vibration data exceeds 20 g.

In the experiment, 14 run-to-failure datasets under two different operating conditions and 10 run-to-failure datasets under two different operating conditions are used from the FEMTO dataset and the XJTU-SY dataset, respectively.

Data Collection for Fault Detection and Diagnosis

The purpose of our fault detection method was to detect bearing faults and diagnose the fault types. Among the various fault types, inner race, outer race, and contaminant faults were considered. As an initial step, we collected two channels (the horizontal and vertical axes) of vibration signals for normal and faulty conditions using the apparatus shown in Figure 5.13 (a).



(a) (b) (c)

Figure 5.13: Test bench and bearings. (a) test bench (model: EMOD FK-FIE2100LA, Pmech: 3 kW, nnom: 1440 /min, Vnom: 400V, Inom: 6.4A, cosϕ: 0.78), (b) sensors mounted on the motor, (c) bearing with different faulty conditions.

The motors of the test bench were 3 kW three-phase induction motors with rated voltage 400 V and current 6.4 A. The configuration followed the power drive setup in [156]. We connected the motors to the controller via inverters to control the speed and torque. The running environment had a 25 Hz operation frequency and 10 Nm torque. The motors and inverters were mounted on the steel rail to fix the electronic machines.

For data acquisition, two vibration sensors (model: MMF KS80D, range: ±60 g, bandwidth 22 kHz, sensitivity: 0.1 V/g) were used to record vibration in the x- and z-axis (as shown in Figure 5.13 (b)). An oscilloscope was connected to vibration sensors and the recorded vibration signals were stored in a server. In our experiments, the sampling rate of the vibration signal was 1 MHz. Vibration data from the induction motor were collected from the test bench under varying environmental conditions over the course of a year.

To measure faulty condition data, bearings were artificially damaged by the following methods. For inner and outer race faults, we drilled into the middle of the inner and outer raceway in the bearings after removing the metal shield and the grease in the bearing. The drilling diameter was 1, 3, and 5 mm for low, medium, and high severity, respectively. For the contaminant, we inserted metal chips in the cage. The bearing with different faulty conditions is shown in Figure 5.13 (c).

We transformed the sensor data into the image domain using NSP. Faulty condition data showed high spectral density in a high-frequency band (30 to 40 kHz), known as the resonance band. The spectral density for 25 to 40 kHz of faulty condition data was greater than that of normal condition data, as shown in Figure 5.14. In this experiment, three bandpass filters for NSP were used: 10 to 30 kHz, 30 to 50 kHz, and 0 to 250 kHz.

The sensor data were collected under various circumstances. Figure 5.15 shows normal data under various circumstances, with the range of images being very broad.

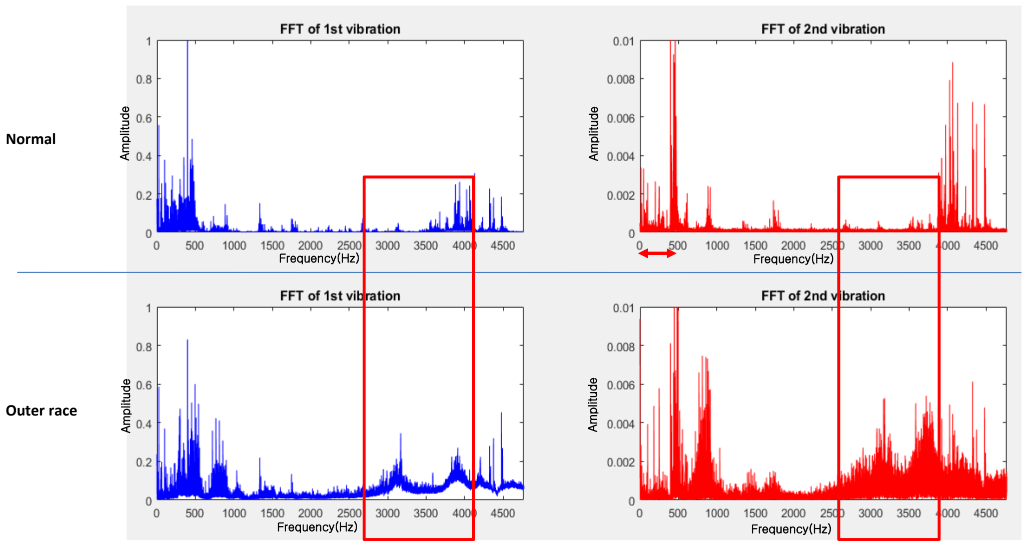


Figure 5.14: Spectral density difference in resonance bandwidths.

Table 5.2: Image data sets for motor condition.

Type		No. of Images
Normal		21,000
Contaminant		800
Inner Race	High	1,000
	Mid	1,000
Outer Race	High	1,000
	Mid	1,000
	Low	1,000

In Figure 5.16, each image shows a fault type that can be distinguished by comparison with the normal image. Table 5.2 describes the detailed dataset for each operating condition. To verify the capability of FDD in data imbalanced conditions, the number of images for normal conditions was much greater than the number of images of bearing faults.

5.4.2 Results and Comparisons

Evaluation of CNN-HS for the FPT determination

Datasets of bearings in condition 1 (1800rpm and 4000N, C1) and condition 2 (1650rpm and 4200N, C2) from the FEMTO dataset were used in our experiment. Each condition has seven run-to-failure datasets (we denote them as B1 to B7). Two datasets from each operating condition were used as the training dataset, and the

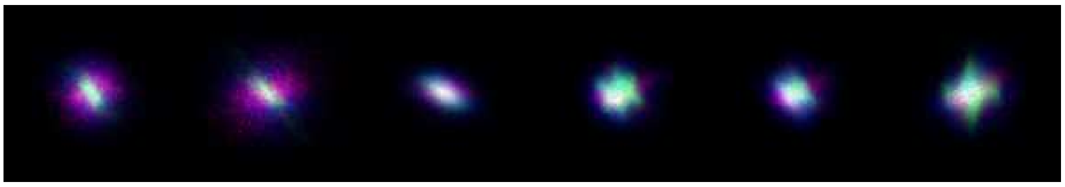


Figure 5.15: Images of normal data under various conditions.

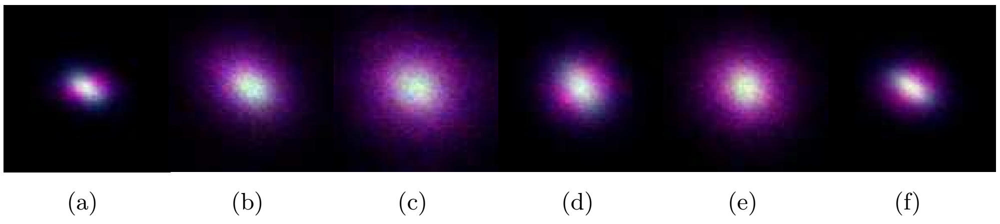


Figure 5.16: Images of bearing fault data. (a) Contaminant, (b) outer high, (c) outer medium, (d) outer low, (e) inner high, and (f) inner medium.

Table 5.3: Predicted FPT in FEMTO dataset (condition C1 and C2, unit: seconds).

Bearing (End time)	**RMS**	**AE**	**CNN-HS**
C1B3 (23,750)	16,760	14,940	740
C1B4 (14,280)	10,870	10,850	1,880
C1B5 (24,630)	24,100	24,130	16,670
C1B6 (24,480)	24,070	24,130	21,590
C1B7 (22,590)	22,010	21,990	8,800
C2B3 (19,950)	19,390	19,390	19,950
C2B4 (7,510)	7,400	7,350	7,430
C2B5 (23,110)	22,920	23,110	930
C2B6 (7,010)	6,840	6,850	6,870
C2B7 (2,300)	2,210	2,230	2,250

trained model was tested on five other datasets in the same condition. The results of the prediction in C1 are compared with previous methods, RMS and AE, in Figure 5.17 and 5.18. Again, RMS and AE require a pre-specified threshold value for HS division, and the threshold value is represented by the red horizontal line. FPT, determined by a threshold value in RMS and AE, and by a trigger mechanism in CNN-HS, is represented as a vertical red dotted line.

The FPTs measured from all the experiments in both C1 and C2 are recorded in Table 5.3. The test results show that our CNN-HS can detect faulty features much faster. On average, the CNN-HS method predicted faults 47.65% and 44.80% faster than the RMS and AE methods, respectively in terms of time.

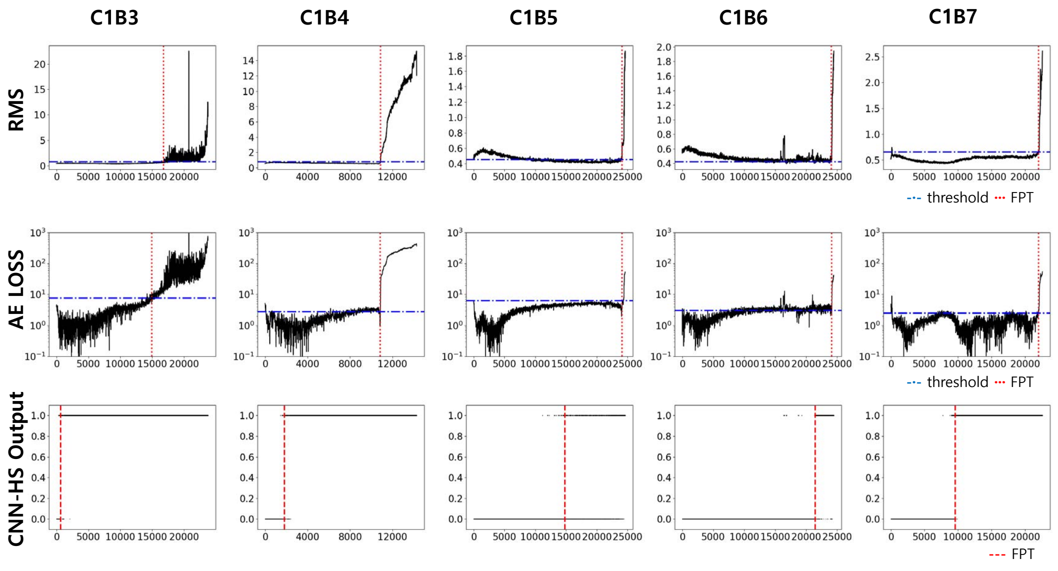


Figure 5.17: Predicted FPT using RMS, AE, and CNN-HS on FEMTO dataset (condition C1).

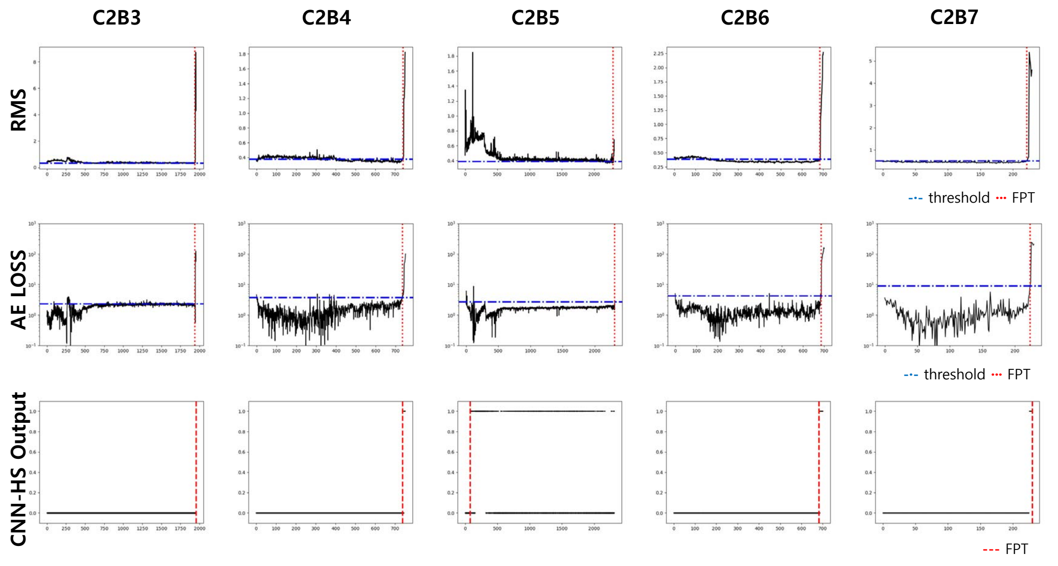


Figure 5.18: Predicted FPT using RMS, AE, and CNN-HS on FEMTO dataset (condition C2).

An analysis of the features extracted by NSP and CNN-HS was conducted. We visualize features using t-Distributed Stochastic Neighbor Embedding (t-SNE) [113],

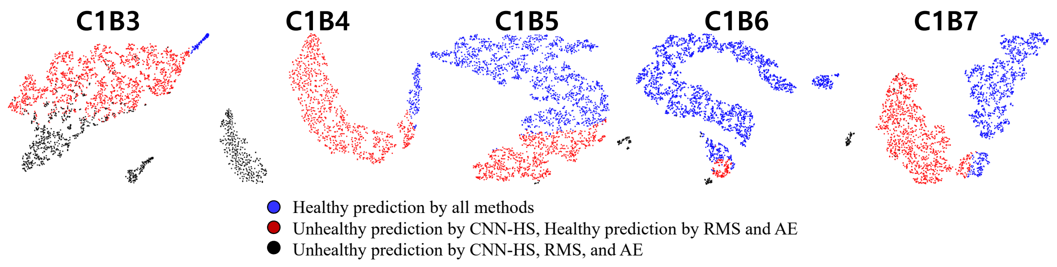


Figure 5.19: Feature maps of CNN-HS using t-Distributed Stochastic Neighbor Embedding (t-SNE) with principal component analysis (PCA) (condition C1).

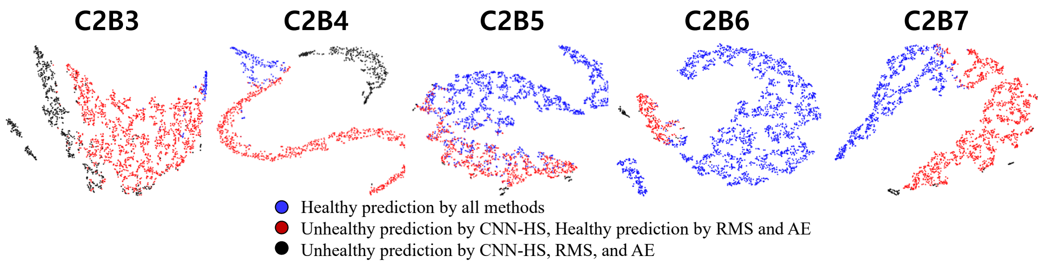


Figure 5.20: Feature maps of CNN-HS using t-SNE with PCA (condition C2).

which is commonly used for visualizing high-dimensional features in scatter plots and projects high-dimensional objects into two-dimensional points, such that similar objects are closer and dissimilar objects are further away from each other. The features were extracted from the output of the first fully connected (FC) layer, which consists of 500 nodes. The 500-dimensional features were then reduced to 100 dimensions by using principal component analysis (PCA) [112] and further reduced to two dimensions by using t-SNE. By this procedure, the features of NSP were mapped into a two-dimensional plane. Figures 5.19 and 5.20 show the reduced features of tested bearings in C1 and C2 mapped into a two-dimensional plane.

This analysis shows that our CNN-HS model is able to detect the HS division features of NSP better than RMS and AE. Blue and black points are vibration conditions, where RMS, AE, and CNN-HS predict 'healthy' and 'unhealthy', respectively. The difference in FPT comes from the red points, where the CNN-HS model was able to identify the faulty features not detected by RMS- and AE-based methods. This feature map proves that the NSP is an appropriate tool for extracting useful features from raw vibration data to represent the health of machinery. The NSP was able to transform vast amounts of raw vibration signal data into a single image from which features could be extracted using the CNN-HS model. Thus, there

Table 5.4: Evaluation of RUL prediction on the FEMTO and XJTU-SY datasets.

Methods	Metric	FEMTO Condition1	FEMTO Condition2	XJTU-SY Condition 1	XJTU-SY Condition 2
	MAE	0.1856 ± 0.0546	0.1738 ± 0.0521	0.2274 ± 0.0496	0.2063 ± 0.0427
MCNN [142]	RMSE	0.2344 ± 0.0701	0.2105 ± 0.0557	0.2484 ± 0.0452	0.2302 ± 0.0319
	MAPE	0.8943 ± 0.4802	1.3567 ± 0.7913	1.1659 ± 0.3749	1.1802 ± 0.3945
	MAE	0.1739 ± 0.0542	0.1771 ± 0.0515	0.2588 ± 0.0499	0.2112 ± 0.0891
DANN [148]	RMSE	0.2063 ± 0.0603	0.2060 ± 0.0578	0.2966 ± 0.0587	0.2398 ± 0.0973
	MAPE	**0.4633 ± 0.0811**	**0.4970 ± 0.1200**	**0.4340 ± 0.0816**	**0.3872 ± 0.1535**
	MAE	0.1697 ± 0.0267	0.2694 ± 0.0913	0.1919 ± 0.0469	0.1238 ± 0.0434
CNN-HS [77]	RMSE	0.1959 ± 0.0297	0.2932 ± 0.0890	0.2172 ± 0.0505	0.1429 ± 0.0419
	MAPE	0.7983 ± 0.4179	1.2953 ± 0.9574	0.6574 ± 0.3538	0.6563 ± 0.4435
	MAE	**0.0906 ± 0.0237**	**0.1644 ± 0.0456**	**0.1620 ± 0.0476**	**0.1132 ± 0.0476**
GMFE (ours)	RMSE	**0.1053 ± 0.0222**	**0.1870 ± 0.0408**	**0.1795 ± 0.0488**	**0.1259 ± 0.0488**
	MAPE	0.4794 ± 0.0859	0.8515 ± 0.5413	0.5306 ± 0.1445	0.5721 ± 0.4197
	MAE	0.0678 ± 0.0212	0.2142 ± 0.0591	0.2541 ± 0.0802	0.2145 ± 0.0790
No FPT	RMSE	0.0842 ± 0.0274	0.2317 ± 0.0540	0.2647 ± 0.0770	0.2217 ± 0.0747
	MAPE	0.5860 ± 0.3584	1.3122 ± 1.0572	2.7105 ± 3.3264	0.9537 ± 0.6268
	MAE	0.1983 ± 0.0944	0.2879 ± 0.1059	0.2393 ± 0.1239	0.1295 ± 0.0664
1D-GAN	RMSE	0.2099 ± 0.0936	0.3143 ± 0.1098	0.2529 ± 0.1200	0.1427 ± 0.0650
	MAPE	0.8234 ± 0.4407	1.5198 ± 1.3135	0.6782 ± 0.2530	1.2928 ± 1.6532

is a clear distinction between the features that were predicted to be unhealthy and healthy by the CNN-HS method as well as compared with the predictions of other methods. Moreover, we observed that our CNN-HS method distinguishes the faulty features of the images produced by the NSP. The region of the black point is small and far away from the blue and red regions. CNN-HS combined with NSP is able to detect minor symptoms (undetected by the other two methods), which leads to more lead time for the investigation of the failure process.

Evaluation of GMFE for the RUL prediction

To evaluate the proposed method, while one bearing dataset is used for testing, the other bearings under the same operating conditions are adopted to train the DNNs. The experimental results are averaged by five trials to reduce the effect of model randomness. To address the advantage of the proposed method, we compare the proposed method with the multiscale CNN method (MCNN) by Li et al. [142], the deep adversarial neural networks (DANN) by Li et al. [148], and the proposed CNN-LSTM method with the supervised HS division method by Suh et al. [77]. Table 5.4 shows the quantitative evaluation results compared to other methods by using three evaluation metrics. It can be observed that the proposed method generally achieves lower prediction errors than other methods.

To evaluate the effect of FPT and NSP, the ablation study is conducted. ‘No FPT’ is adopted and the degradation is considered to start at the beginning of machine operation to indicate the benefits of the HS division and FPT determination. The

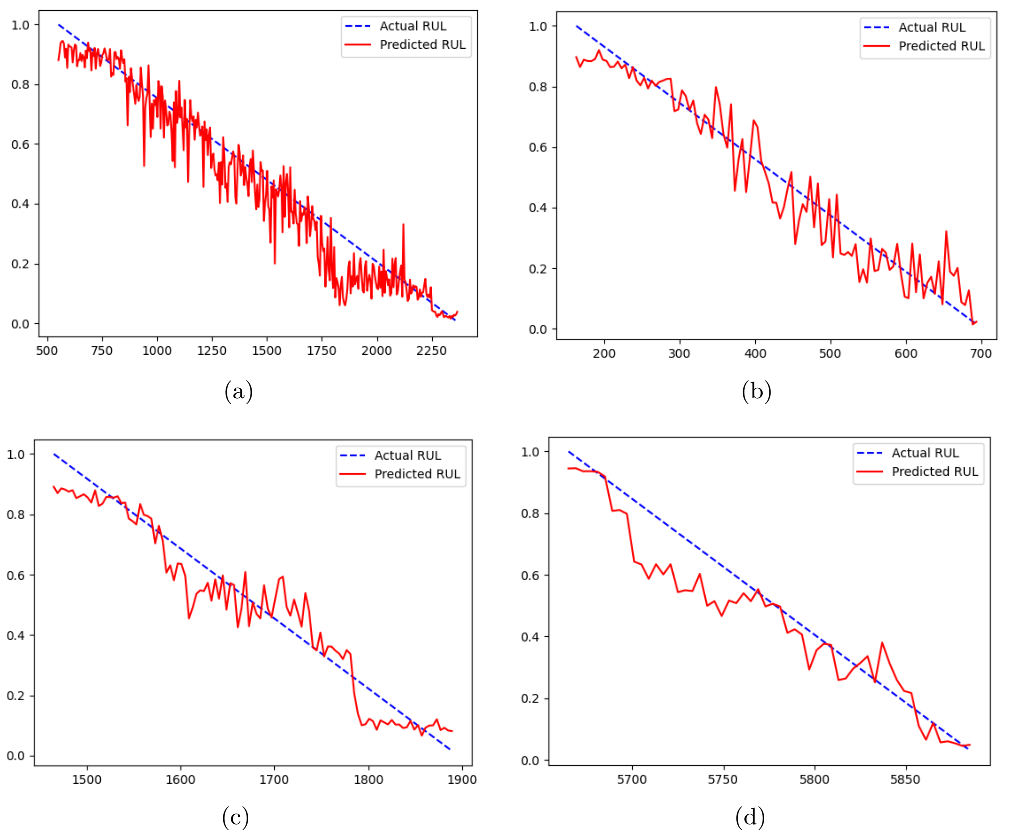


Figure 5.21: RUL prediction results on four testing bearings. (a) Bearing 1-3 in the FEMTO dataset, (b) Bearing 2-6 in the FEMTO dataset, (c) Bearing 1-3 in the XJTU-SY dataset, (d) Bearing 2-1 in the XJTU-SY dataset.

‘1D-GAN’ method is implemented without merging the multiscale features by transforming them into the NSP image and the proposed CNN-LSTM-RUL to show the advantage of the transformed NSP images and the CNN-LSTM-RUL. The proposed method compared to ‘No FPT’ and ‘1D-GAN’ showed better results, indicating that the proposed HS division method is effective in capturing machine degradation at the beginning point and the CNN-LSTM-RUL with the transformed NSP images enhances the generalization ability on various datasets.

Figure 5.21 presents the RUL prediction results on four testing bearings in the two datasets. The degradation evaluations start when the FPTs of the testing bearings are detected. It can be observed that the degrading patterns are effectively captured by the proposed method and the errors between the actual RUL and the predicted RUL are quite small.

Testing Fault Classification under Data Imbalanced Conditions

Before evaluating CBFD under data imbalanced conditions, we considered two issues: (1) that the data imbalance ratio affects the performance of the classifier more than the number of minority classes; and (2) that the learning rate and epoch number should be considered to prevent over-fitting.

In [24], the degradation of FDD classification accuracy under an unbalanced normal and fault condition ratio was demonstrated; for example,

- False alarms, identified when FDD determines a fault despite normal condition;
- Misfiring, where ground truth observations show a fault condition, but FDD indicates normal condition;
- Confusion, where ground truth observations show one fault condition, but FDD determines another.

According to the previous study [24], the accuracy of testing declines overall when the imbalance ratio of the dataset increased in the case of binary classification (normal or rotor fault/bearing fault). This means that the classification performance of these diagnosis methods was easily affected by the imbalance setting. For motor fault detection, which usually presents severely imbalanced data, misfiring and confusion from the classifier were more important than false alarms.

In addition to increased classifier errors, the over-fitting phenomenon also resulted in poor accuracy for the test dataset, but good accuracy at the training stage. In the learning process, the adaptive moment estimation (Adam) optimizer was employed and tested using static and decaying learning rates. The static learning rate was 0.0001 and the decaying learning rate was set as exponential decay from 0.0005 with 50% decay every 100 epochs (with a total of five decay steps in 500 epochs). The decaying learning rate performed better than the static learning rate, and the accuracy evolution of CBFD became weak at approximately 200 epochs. Therefore, for all further experiments, we determined hyperparameters, with the learning rate and terminal epochs being 50% decay from 0.0005 and 250 epochs, respectively.

To verify the relationship between classification accuracy and data imbalance conditions, we performed two types of experiments: 1) training a model by fixing the number of normal data images to 20,000, which was close to the maximum size of the normal dataset shown in Table 5.2, and changing the ratio of the data imbalance from 30:1 to 1000:1; 2) fixing the number of normal data images to 10,000 and changing the ratio of data imbalance from 20:1 to 400:1. As the volume of normal data was larger than the volume of fault data, the number of normal data images was fixed and the ratio of data imbalance was changed.

The test set for all conditions contained 300 randomly selected images for each category. To ensure robust results, we took the mean value of 10 trials for each data imbalance rate case as the final result.

In the first experiment, the model was trained by fixing the number of normal data images to 20,000 and changing the ratio of the data imbalance. Figures 5.22(a)–(c)

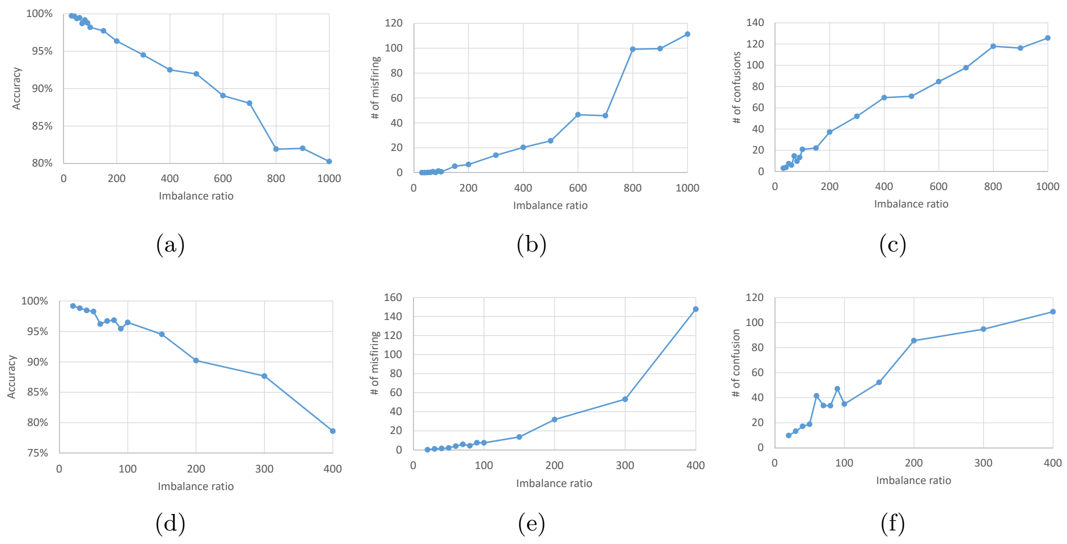


Figure 5.22: Test results for fixed normal data at 20,000 and changing ratio of data imbalance: (a) accuracy, (b) misfiring, and (c) confusion. Test results with fixed normal data to 10,000: (d) accuracy, (e) misfiring, and (f) confusion.

show the overall FDD accuracy and the numbers of misfirings and confusion for various imbalance ratios. No false alarms were observed, however, the accuracy declined and the numbers of misfirings and confusion increased with the imbalance ratio of the dataset. However, the accuracy was still higher than 80%, even when the imbalance ratio was 1000:1.

Figures 5.22(d)–(f) show the result of the second experiment, in which we used 10,000 normal data images. Compared to the first experimental result, it indicates that the numbers of misfirings and confusion were higher for the same imbalance ratio in Figures 5.22(a)–(c). The accuracy also decreased, in general, owing to the smaller dataset but still achieved around 80% even for the most severe imbalance ratio of 400:1.

As shown in Figure 5.23, the proposed method (CBFD with BF-NSP) gave higher classification accuracy than CNN with the continuous wavelet transform scalogram (CWTS) [29, 157] and DNN with HHT [24], in which features were extracted using the HHT and the DNN was used as a classifier. To ensure a fair comparison, both models were trained using 5000 random sample data points and tested using 1200 random samples. The comparison results represent the mean values of 10 trials for every data imbalance rate. We found that CBFD with BF-NSP had 95% accuracy, even when the imbalance ratio was 20:1. Its accuracy fell below 90% when the imbalance ratio reached 50:1. On the other hand, the DNN with HHT [24] was more sensitive to the imbalance ratio. Only for an imbalance ratio of 7:1 was the accuracy over 95%. When the imbalance ratio was 9:1, the accuracy was below 80%.

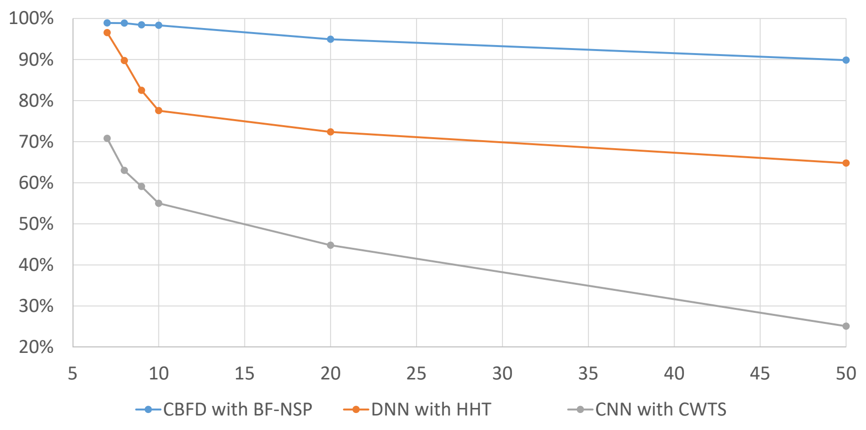


Figure 5.23: Comparison of CBFD with NSP with other methods.

Table 5.5: Computational time comparison of the convolutional neural network (CNN)-based bearing fault detector (CBFD) with the bearing fault-nested scatter plot (BF-NSP) with other methods.

	Training Time (s)	**Testing Time (ms)**
CBFD with NSP	805.48	2.59
DNN with HHT	207.50	1.30
CNN with CWTS	66.15	0.38

In the case of the CNN with CWTS [29, 157], the accuracies in the imbalanced conditions were at least 28.1% points lower than CBFD with BF-NSP. When the image transformation using CWTS was applied to the data, the training accuracy of the CNN with CWTS was lower than 90%, so that the results of the CNN with CWTS declined in the data imbalanced conditions. Even though we tried to classify the CWTS images using our CBFD, the differences between the results of the two methods were minor. As shown in Table 5.5, the CNN with CWTS provided the fastest training and testing among the three methods because the size of the CWTS image (80 × 80) was smaller than the size of the NSP image (128 × 128). In summary, our CBFD with BF-NSP outperformed the DNN with HHT and the CNN with CWTS, and was capable of detecting bearing faults even when the data imbalance was high.

Testing Fault Classification with Oversampling

As discussed, to improve the classification accuracy under data imbalance conditions, oversampling using DCWGAN-GP was employed. To mitigate problems caused by severe data imbalance, we trained the DCGAN and the proposed DCWGAN-GP for each fault type, allowing the model to generate synthetic fault images. Each

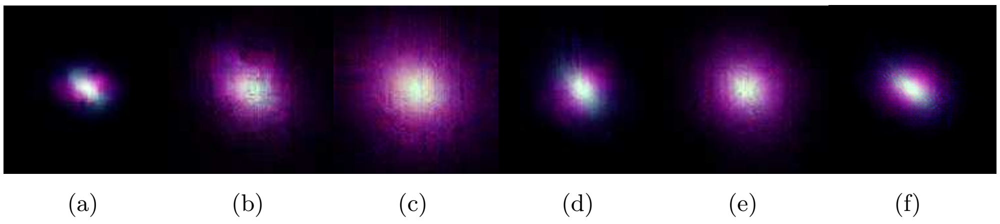


Figure 5.24: Generated images of bearing fault data using the Wasserstein generative adversarial networks with gradient penalty on the DCGAN architecture model (DCWGAN-GP). (a) Contaminant, (b) outer high, (c) outer medium, (d) outer low, (e) inner high, and (f) inner medium.

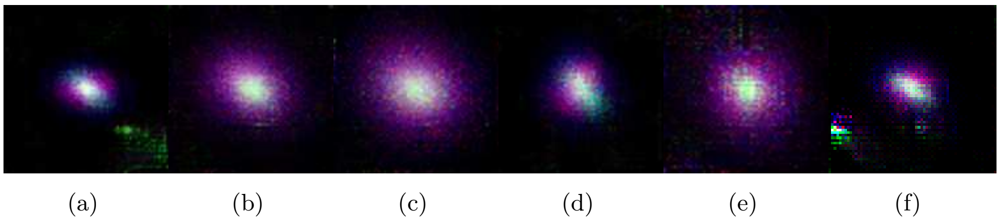


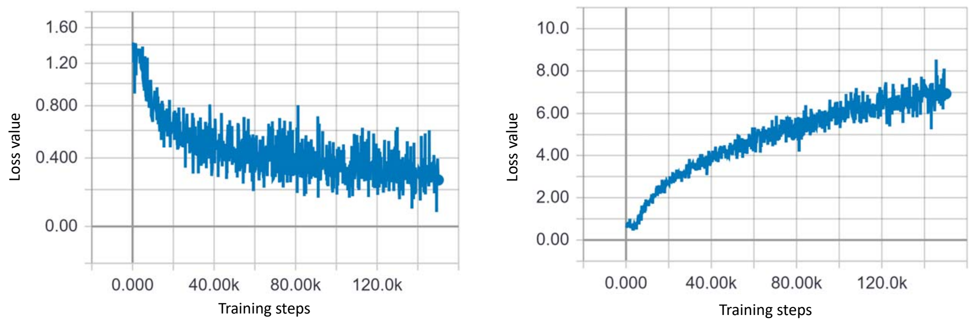
Figure 5.25: Generated images of bearing fault data using the DCGAN. (a) Contaminant, (b) outer high, (c) outer medium, (d) outer low, (e) inner high, and (f) inner medium.

model was trained using the available fault dataset defined above. Figure 5.24 shows images computed by DCWGAN-GP for each fault type. The generated images can be distinguished by comparison with normal images and were similar to the images of real bearing fault data. For comparison, images generated using DCGAN [107] are shown in Figure 5.25; these images can also be distinguished by comparing with normal images and are similar to images of real bearing fault data.

The images generated using DCGAN contained background noise. To identify the reason for the noise, we monitored the trend of the loss value in the training step (Figure 5.26). We found that the loss of the DCGAN generator increased as the step proceeded and that the loss showed significant variation. Figure 5.27 shows the losses of the proposed DCWGAN-GP; the losses for both the discriminator and the generator converged toward zero as the training step proceeded. Both models were successfully trained for the fault types, however, the objective function of the WGAN-GP provided superior stability and quality of gradients.

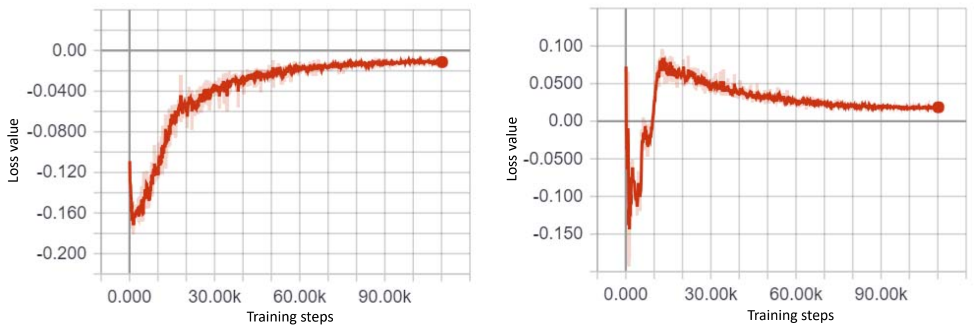
We oversampled the fault data of the proposed generative model under various data-imbalanced conditions. Here, the ratio between normal and fault data images is denoted as the normal-to-fault ratio (NFR), and the ratio that enhances NFR using oversampling as adjusted NFR (A-NFR).

- NFR = (# of normal data images) : (# of fault data images),



(a) (b)

Figure 5.26: Losses of the DCGAN. (a) Losses of the discriminator, and (b) losses of the generator.



(a) (b)

Figure 5.27: Losses of DCWGAN-GP. (a) Losses of the discriminator, and (b) losses of the generator.

- A-NFR = (# of normal data images) : (# of fault data images) + (# of oversampled fault data images)

We considered the serious data-imbalanced conditions where NFR > 100:1. In the experiment, not only the imbalance ratio but also the number of data sets were considered, because the number of the majority and minority datasets had an effect on model training. Firstly, we fixed the number of normal data images to 10,000. Under a normal number of data images of 10,000, we considered two cases of the data-imbalanced condition: fault data images of 50 (which yields NFR = 200:1), and fault data images of 100 (which yield NFR = 100:1). Based on these conditions, we oversampled fault data images using DCGAN and DCWGAN-GP until the number of total fault data images (original and oversampled) reached 500, 1000, 2500, 5000, and 10,000 (which yielded A-NFR = 20:1, 10:1, 5:1, 2:1, and 1:1). The experimental results of the 10,000 normal data images are shown in Figures 5.28 and 5.29. The lower bound of FDD accuracy is 90.77% and 94.63% when NFR is 200:1 and 100:1 without oversampling, respectively. The upper bound of FDD accuracy was 98.99% when NFR was 20:1 without oversampling.

Secondly, we fixed the number of normal data to 20,000. Similar to normal data images of 10,000, we considered two data-imbalanced conditions: NFR = 400:1 and 200:1, and oversampling using DCGAN and DCWGAN-GP was conducted in the cases of A-NFR = 20:1, 10:1, 5:1, 2:1, and 1:1. The experimental results of 10,000 normal data images are shown in Figures 5.30 and 5.31. The lower bound of FDD accuracy is 91.30% and 95.63% when NFR is 400:1 and 200:1 without oversampling. The upper bound of FDD accuracy is 99.42% when NFR is 40:1 without oversampling. (The detailed experimental values are shown in Table 5.6.)

Oversampling using DCWGAN-GP and DCGAN improved the FDD overall accuracy. In the case of 10,000 normal data images (shown in Figures 5.28 and 5.29), DCWGAN-GP improved the accuracy by 7.28% and 4.67% at A-NFR = 1:1 compared to the accuracy at NFR = 200:1 and 100:1. DCGAN also improved the accuracy by 3.12% and 2.42% at A-NFR = 1:1 compared to the accuracy at NFR = 200:1 and 100:1. Similarly, oversampling using DCWGAN-GP and DCGAN showed better accuracy compared to the baselines that were no sampling cases (shown in Figures 5.30 and 5.31).

DCWGAN-GP outperformed DCGAN in generative oversampling. In the case of NFR = 100:1 (shown in Figure 5.29), DCWGAN-GP improved accuracy from 4.1% to 4.7% compared to no oversampling when DCGAN did only from 0.5% to 2.4%. In addition, in the case of NFR = 400:1 (shown in Figure 5.30), oversampling using DCWGAN-GP improved the accuracy by 6.28 to 7.11% when the result of DCGAN was only 2.24 to 4.33%. Furthermore, the variance in accuracy with oversampling of DCWGAN-GP was around 1%, with overall accuracies 98% and 99% higher, respectively. While the accuracy for oversampling was lower than just 2% or less than the accuracy for upper bound results (NFR =20:1 and 40:1, in the case of normal data images 10,000 and 20,000, respectively.)

As seen in Figures 5.28–5.31, as oversampling enhanced A-NFR, the accuracy increased. We emphasize here that oversampling using DCWGAN-GP showed 97% or higher accuracy when A-NFR was 20:1. As mentioned before, BF-NSP and CBFD showed good performance in the tolerable imbalance condition such as NFR = 20:1 to 50:1. This means that oversampling using DCWGAN until A-NFR met the tolerable imbalance conditions was enough for FDD accuracy. DCGAN showed less than 96.5% accuracy (a gap between DCWGAN-GP and DCGAN ranges from 2.7% to 5.97%) when A-NFR is 20:1. Therefore, DCGAN required more oversampling data than DCWGAN-GP, and this fact resulted in the increased cost of computation.

NFR was one of the reasons for FDD performance degradation, but the number of datasets was also important. Figures 5.28 and 5.31 have the same NFR= 200:1, but different number of datasets. Comparing these two results, the case of 20,000 normal data images showed better accuracy than the case of 10,000 normal data images.

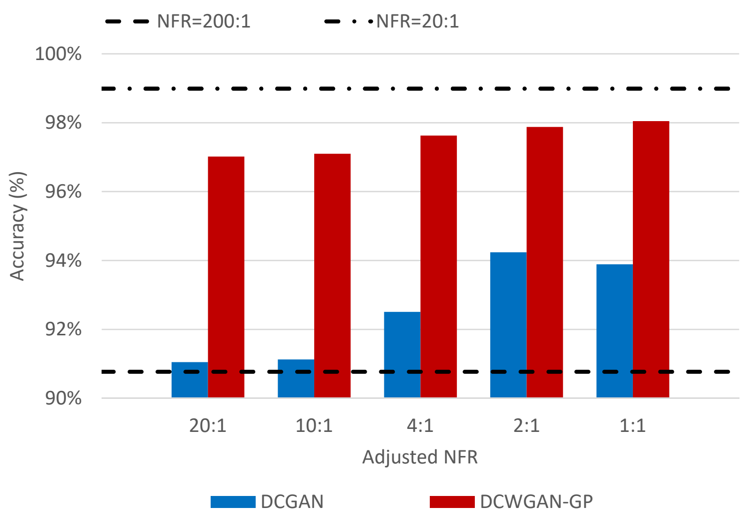


Figure 5.28: Fault detection and diagnosis (FDD) accuracy testing:normal-to-fault-ratio (NFR)=200:1 with 10,000 normal data.

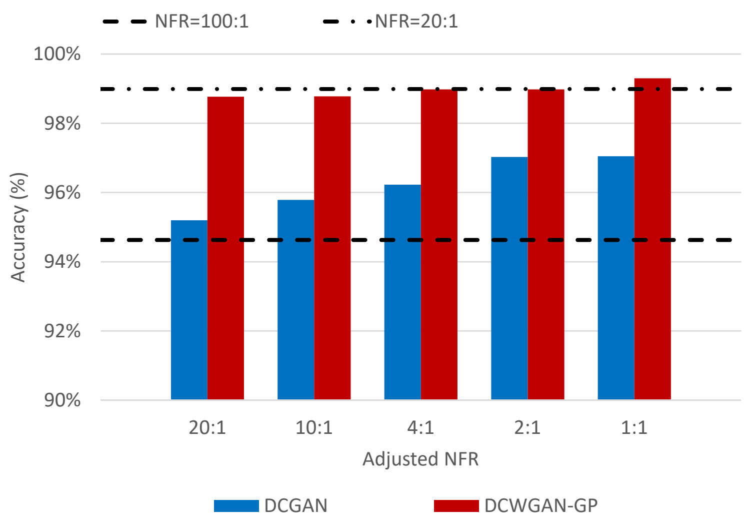


Figure 5.29: FDD accuracy testing:NFR=100:1 with 10,000 normal data.

5.4.3 Discussion

By experimental evaluation, the feasibility of our model for HS division is demonstrated. Our model is able to capture faulty features ahead of previous methods, as demonstrated in the feature maps, which generate a significant amount of lead time on the maintenance time horizon. To verify this capability, we conducted another test for the feasibility of our method for HI prediction. Thanks to the capability of the initial wearing feature extraction, our proposed method can make earlier

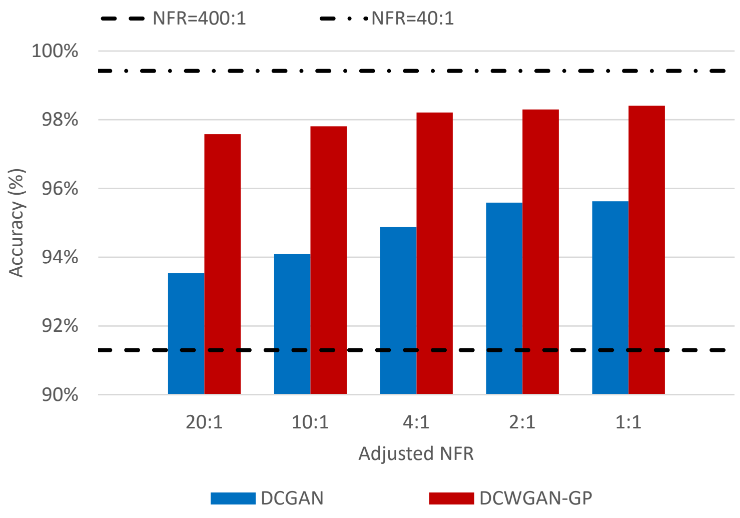


Figure 5.30: FDD accuracy testing:NFR=400:1 with 20,000 normal data.

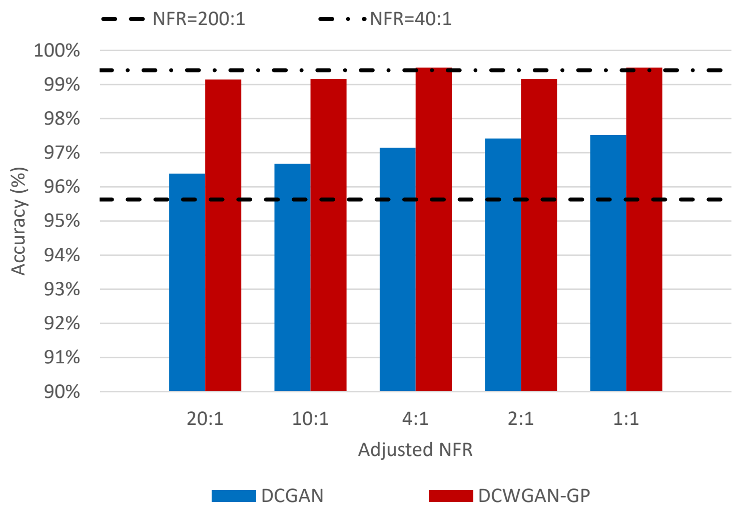


Figure 5.31: FDD accuracy testing:NFR=200:1 with 20,000 normal data.

FPT predictions than the signal-based and unsupervised data-driven method that requires specified thresholds.

In order to verify the performance of the proposed method, more experimental evaluations on the reliability of the dataset are needed. There are other benchmark datasets, but we could not work with them due to the following reasons. NSP is a scalable and signal-independent data-wrangling method, but it requires multiple channel signals. Other open benchmark datasets containing multiple channels of vibration data were not found. As of now, there are no datasets available with

Table 5.6: Accuracy of original data and oversampled data using the deep convolution generative adversarial network (DCGAN) and the proposed Wasserstein generative adversarial networks with gradient penalty (WGAN-GP) on DCGAN architecture model (DCWGAN-GP) method. (a) Normal-to-fault ratio (NFR) = 200:1 with 10,000 normal data; lower bound = 90.77% at NFR=200:1, upper bound = 98.99% at NFR = 20:1. (b) NFR = 100:1 with 10,000 normal data; lower bound = 94.63% at NFR=100:1, upper bound = 98.99% at NFR = 20:1. (c) NFR = 400:1 with 20,000 normal data; lower bound = 91.30% at NFR=400:1, upper bound = 99.42% at NFR = 40:1. (d) NFR = 200:1 with 20,000 normal data; lower bound = 95.63% at NFR=200:1, upper bound = 99.42% at NFR = 40:1.

(a)

A-NFR	20:1	10:1	4:1	2:1	1:1
DCGAN	91.05%	91.13%	92.51%	94.24%	93.89%
DCWGAN-GP	97.02%	97.10%	97.63%	97.88%	98.05%

(b)

A-NFR	20:1	10:1	4:1	2:1	1:1
DCGAN	95.20%	95.79%	96.23%	97.03%	97.05%
DCWGAN-GP	98.77%	98.78%	98.98%	98.98%	99.30%

(c)

A-NFR	20:1	10:1	4:1	2:1	1:1
DCGAN	93.54%	94.10%	94.88%	95.59%	95.63%
DCWGAN-GP	97.58%	97.81%	98.21%	98.30%	98.41%

(d)

A-NFR	20:1	10:1	4:1	2:1	1:1
DCGAN	96.39%	96.68%	97.15%	97.42%	97.52%
DCWGAN-GP	99.15%	99.16%	99.50%	99.16%	99.50%

labeled ground truth FPT. As such, this work has been developed in the face of this existing difficulty and thus remains among the first works of this kind.

For efficient HS division using supervised learning, there is a need for an extensive amount of degradation datasets under variable operational conditions. Wearing patterns are varying in laboratory test benches as well as in real-world applications. Some wearings degrade gradually, while others degrade rapidly. A general supervised HS division method can be developed if a large amount of data covering such varying wear is made available. Presently, there is a dearth of rich data in this context. In light of this aspect, the present work becomes significantly important as it provides the early detection of FPT, exceeding the performance of common existing approaches.

Table 5.7: FPT to total duration ratio in different training data variation in CNN-HS under conditions 1 and 2.

Bearing(test)	**Bearing(train)**		
	C1-B1B2	**C1-B1B4**	**C1-B3B4**
C1B5	67.6%	79.2%	87.0%
C1B6	88.1%	86.9%	97.0%
C1B7	38.9%	51.2%	60.0%
	C2-B3B4	**C2-B1B3**	**C2-B1B2**
C2B5	2.9%	2.4%	4.0%
C2B6	5.3%	11.8%	98.0%
C2B7	2.0%	5.6%	97.8%

Impact of Combination of Training Data

In our proposed method, training data from the same operating condition is needed, and the selection of the training dataset of our model plays a critical part in the performance. During our experimental phase, we tested different combinations of training data to observe how the variation in bearing degradation in the training set affected the model in both C1 and C2. The impact of the different combinations of training data in terms of the average FPT to the total duration ratio is shown in Table 5.7. These results show that the predicted FPT varies by up to 34.45%. The bearing datasets under accelerated degradation show two phenomena: bearing that shows gradual degradation and bearing that shows rapid degradation. From our intuition, a suspicion arises as to whether the model is predicting healthy conditions to be unhealthy prematurely since, in the training phase, relatively healthy conditions might have been labeled as unhealthy since rapid degradation was shown near the end. If this is the case, then the model might cause erroneous false alarms during healthy conditions. In the future, we plan to test our proposed method on a test bench with both degrading bearing data and normal bearing data.

Impact of Labeling Ratio of the Wearing Process

In the evaluation, NSP from the first 5% of the total degradation process was labeled as healthy, and the last 5% as unhealthy. This labeling makes the ratio of used training data to unused training data in the training dataset (denoted as labeling ratio) equal to 1:9. The labeling ratio of 1:9 is determined by experimental comparisons: from 1:9 to 5:5. In each condition of the FEMTO dataset, two bearing datasets are selected as training data, and the rest of the datasets are tested. First, FPT is predicted using the trained model in each test dataset. Based on the predicted FPT, the test dataset is divided into a healthy condition period (HCP) and an unhealthy condition period (UCP). The ground truth of HS division is set to ‘healthy’ for HCP data, and to ‘unhealthy’ for UCP data. The F1 score is used to evaluate the reliability of FPT made by the given labeling ratio. The F1 score is

Table 5.8: The reliability (F1 score) of the predicted FPT corresponding to the labeling ratio.

Condition	**Labeling Ratio**				
	1:9	**2:8**	**3:7**	**4:6**	**5:5**
C1	0.91	0.89	0.87	0.86	0.87
C2	0.97	0.95	0.82	0.77	0.83
C1 + C2	0.94	0.92	0.85	0.82	0.85

defined as follows:

$$F_1 = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{5.10}$$

where $Recall = \frac{TP}{TP+FN}$ and $Precision = \frac{TP}{TP+FP}$. TP, FP and FN denote the true positive, false positive, and false negative values, respectively. The CNN-HS model trained with a ratio of 1:9 showed the highest average F1-score, as shown in Table 5.8.

Effectiveness of the CNN-HS Model and NSP in Representing Degradation Patterns

To verify the effectiveness of the CNN-HS model and NSP in representing degradation patterns, we conducted an experiment on the FEMTO dataset. The experiment we proposed was to predict RMS values using NSP images and CNN-HS. RMS, which is a signal-based method, uses the original time series vibration data. The RMS prediction model using NSP and CNN-HS is shown in Figure 5.32. In the first step, the time series vibration data is transformed into NSP images and the NSP image is used as an input image of CNN-HS. In the training procedure, the RMS value of the sliding frame corresponding to the NSP image is set as an input label of CNN-HS. Unlike the proposed method, NSP images from the total degradation process are used in the training procedure. In the testing phase, the trained CNN-HS is used to compute out RMS values of NSP images acquired from the test data.

The RMS values for each bearing under two operating conditions in the FEMTO dataset are shown in Figure 5.33. Because each bearing has a different degradation pattern and defect level, the pattern of RMS values and the maximum RMS value on each bearing and condition are different. To reduce the scale difference in RMS values, we saturate the maximum RMS value to a threshold value. In this experiment, we evaluated the CNN-HS with a threshold value of two because the minimum value among the maximum RMS values on bearings is two. In order to verify the accuracy of the RMS prediction using NSP and CNN-HS, the root mean square error (RMSE) is used to calculate the accuracy of the prediction values. The RMSE is defined as follows.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(RMS_t - P_t)^2} \tag{5.11}$$

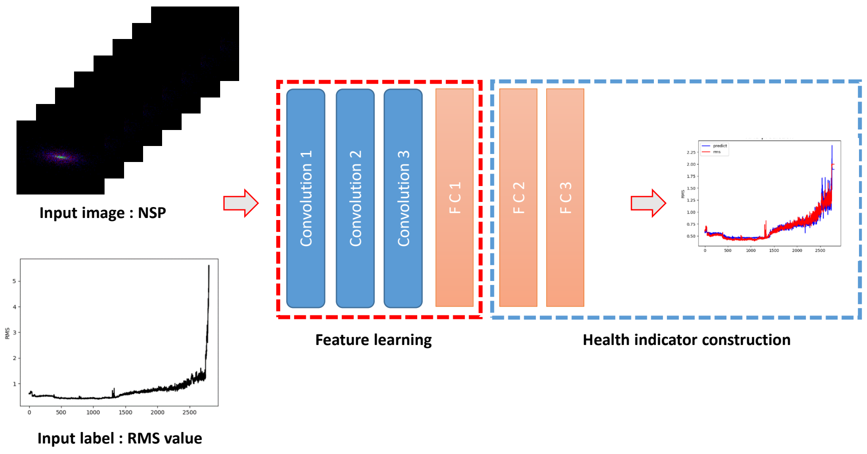
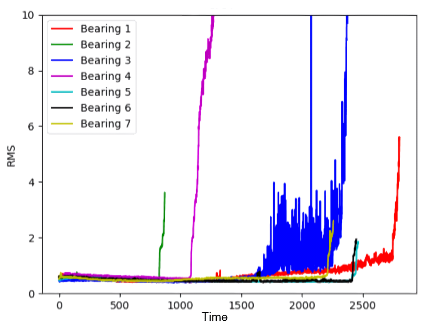
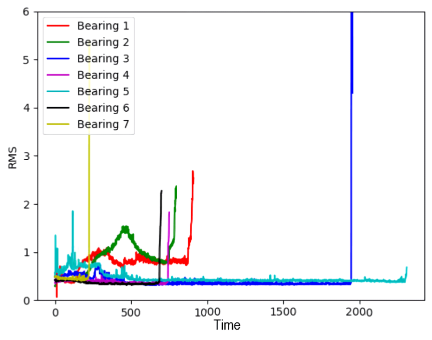


Figure 5.32: The RMS prediction model using NSP and CNN-HS.



(a)



(b)

Figure 5.33: RMS values for each bearing in the FEMTO dataset, (**a**) Condition 1, (**b**) Condition 2.

where RMS_t is the RMS value at time t and P_t is the predicted RMS value by the CNN-HS model. The RMSEs are shown in Table 5.9. The RMSE of C1B3 is higher than the RMSE of other bearings since the RMS value of C1B3 oscillates greatly. Despite the high RMSE of C1B3, most of the RMSE values are quite low. Figure 5.34 shows that the predicted RMS value by the CNN-HS model has very similar patterns to the RMS values. Therefore, NSP can represent the features of time series vibration data and CNN-HS can accurately predict the health stage of rolling bearings.

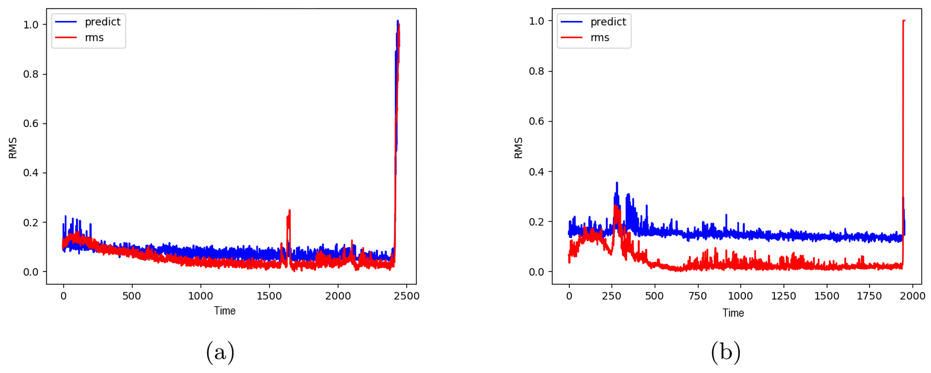


Figure 5.34: RMS value and predicted RMS value by NSP and CNN-HS in (**a**) C1B6 and (**b**) C2B3

Table 5.9: The prediction errors using CNN-HS combined with NSP.

Bearing	**RMSE**	**Bearing**	**RMSE**
C1B3	0.4920	C2B3	0.1522
C1B4	0.2010	C2B4	0.0962
C1B5	0.1028	C2B5	0.1097
C1B6	0.0774	C2B6	0.1503
C1B7	0.2156	C2B7	0.1645
Average	0.2529	Average	0.1271

5.5 Conclusions

In this chapter, novel methods to predict the RUL and detect faults of the bearings used in machinery were proposed. The proposed methods were composed of three different approaches: 1) a supervised HS division using CNN for bearing wear, 2) generalized multiscale feature extraction and RUL prediction method using multiscale GAN, 3) a generative oversampling method to address the data imbalance issue for bearing FDD. For the HS division, the combination of NSP, which allowed us to integrate a frequency analysis that helped to extract faulty wear features, and a CNN-based binary regression model, rid the necessity of designating a threshold, one of the most crucial problems in early fault detection. The CNN-HS model effectively distinguished healthy and unhealthy states unambiguously by changing the anomaly detection problem into a binary regression problem, utilizing a simple trigger mechanism. To minimize the labeling process in supervised learning, a small portion of the dataset was utilized for training the model. Secondly, to extract generalized prognostic features, we designed a multiscale 1D U-Net architecture for the generator and adversarial learning procedure between the generator and the discriminator which contains the reality and domain discriminator. The experimental results indicated that the proposed method, where CNN and CNN-LSTM are combined with NSP based on the reconstructed features of bearing wear, could capture the degrading patterns effectively and achieve better RUL prediction results compared to other methods. Lastly, a novel generative oversampling method to address the data imbalance issue for bearing FDD was proposed. In short, because the volume of faulty condition data was much lower than that of normal condition data, the lower recognition accuracy of the fault condition results. Before introducing the oversampling method, the proposed method was used to transform time-series data into the image domain via the NSP method; bearing faults in the induction motor were classified using designed CNNs. To overcome the data imbalance problem, we generated fault images using DCWGAN-GP. Experiments demonstrated that the proposed method improved accuracy by 7.2% and 4.27% on average and gave maximum values with 5.97% and 3.57% higher accuracy than the previously developed DCGAN approach. Additionally, the accuracy of the proposed method was close to that under data imbalance ratio conditions of 20:1 and 40:1 without oversampling.

Despite the fast and accurate health stage prediction results achieved by the proposed method, it requires a sufficient amount of training data for model development. In real industrial scenarios, supervised data with degradation is difficult to collect. In future research, we plan to divide the health state into numerous stages based on degradation time instead of just healthy and unhealthy in a supervised manner. This model could act as a health indicator and be used to recognize patterns that could be used for RUL prediction. Moreover, we plan to develop the fault prediction model with an attention mechanism and unsupervised transfer learning.

6 Application: Robust Shipping Label Recognition and Validation for Logistics by Fusion of Global-Local Features and using Two-Stage Generative Adversarial Networks

6.1 Problem Definition

This chapter presents the methods proposed to recognize and validate shipping labels by using deep neural networks. To recognize and validate the shipping label, the proposed method comprises three steps: 1) input image quality verification, 2) address and barcode area detection with image calibration, 3) address text recognition with image enhancement. The methods presented in this chapter have appeared in ICIP 2019 [43] and ICPR 2020 [44]. The address text recognition with image enhancement is available online [79].

6.1.1 Input Image Quality Verification

As a first step, the input image quality verification is proposed. The performance of OCR engines and text detection engines is sensitive to image quality and defects on target objects. The quality of the captured image can be affected by many degradations caused during image acquisition. Further, during the packaging and delivery processes, the shipping label can be damaged. Thus, proper assessment and classification of the captured image and object are required to improve the text detection and recognition processes. Four poor conditions of captured images are defined, and example images of generated and collected datasets are shown in Figure 6.1 and Figure 6.2.

1. Unreadable: Blurring is the most common issue in real-world applications, occurring because of defocusing or camera motion [158], and an image acquired at the wrong positioning leads to missing the ROI. By detecting the blur degradation and incorrect positioning, the user reacquires the image and the OCR accuracy can be improved.

2. Contaminated: Shipping labels are often contaminated in the delivery process, or some part of the address area in the shipping label can be occluded by a

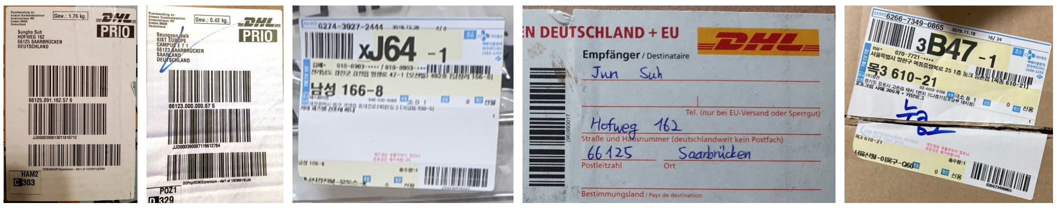

(a) Normal (b) Contaminated (c) Unreadable (d) Handwritten (e) Damaged

Figure 6.1: Examples of the annotated real dataset of the shipping label.

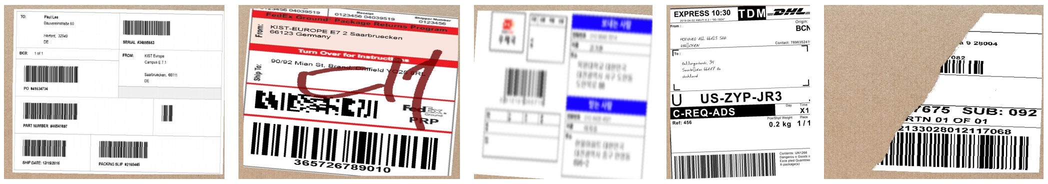

(a) Normal (b) Contaminated (c) Unreadable (d) Handwritten (e) Damaged

Figure 6.2: Examples of the annotated generated dataset of the shipping label.

pen mark. By processing with methods such as document image enhancement and binarization [159, 160], the address in the contaminated or occluded label can be recognized better.

3. Handwritten: The shipping label sometimes contains handwritten addresses. Because recognition engines for machine-printed characters and handwritten characters are different, the two types of characters must be classified differently.

4. Damaged: The shipping label can be torn or occluded by another shipping label in the processes of packaging and delivery. By detecting damaged shipping labels, misdelivery can be reduced and the performance of the text recognition process can be improved.

In this chapter, we propose an input image quality verification method using convolutional neural networks (CNNs) for shipping label inspection. The proposed method classifies the mentioned previously five image types: normal, contaminated address, unreadable image, handwritten address, and damaged shipping label.

However, classification methods based on CNNs are not suitable for direct use for input image quality verification because of two problems. One problem is the varying aspect ratios and sizes of shipping label images. The varying aspect ratios make it difficult to design a CNN with a fixed image size as input. Another problem is that it is difficult to distinguish between contaminants in the address area and contaminants in other areas. In general, there are two methods for handling input images of different sizes: (1) resizing the input image while maintaining the aspect ratio in the padding space and (2) using patches as inputs to the CNN to extract discriminative features. However, resizing the input images leads to increasing the ambiguity between normal and blurred images because of the loss of detailed features, making it difficult to detect defects in ROIs. Conversely, dividing an input image into patches can lead to the loss of global features, degrading the classification performance.

We integrate local and global features from different CNNs. To localize the local features, we detect the barcode and address areas using the deep-learning object detection algorithm You Only Look Once (YOLO) [83, 161]. Then, we localize and crop regions with strong features using one of the famous feature extraction methods, features from accelerated segment test (FAST) [162]. The global features are extracted from the global CNNs with a resized input image, and the local features are extracted from independent local CNNs with the detected address and barcode area and the localized regions using FAST. We concatenate the global and local features together into fully connected fusion layers to classify the shipping label image conditions. The contribution of this study can be summarized into two points. (1) We detect and localize the ROIs using YOLO and FAST and extract the local features using independent local CNNs. (2) By constructing the variant global and local images to combine the global and local features, we propose a novel CNN model-based stacked generalization ensemble method [163] for image quality verification.

6.1.2 Address and Barcode detection with Image Calibration

To verify the addresses in the shipping label, optical character verification (OCV) and recognition (OCR) systems can be deployed. Ribeiro et al. [41] proposed an automatic recognition of expiration dates in food packages by using EAST [39]. As stated in [41], several traditional image processing methodologies have been applied for detection and recognition of text regions, but deep learning-based approaches have shown to be especially effective in these tasks [39, 36, 37, 38]. The utilization of a deep neural network architecture showed high accuracy for the verification and recognition of expiration dates in food package photos.

However, unlike the food package application, the quality of the barcode also has to be inspected for the shipping label. The printed address not only has to be compared to the user-typed address, but its validity also has to be verified. Like OCV, there exist several barcode quality and printed label inspection systems, but they are mostly off-the-shelf and black-box commercial ones [41] and the systems usually cost high and support only a limited number of shipping label formats. Though the address verification is available in [42], their service is to verify the delivery address requested by customers, not the address in the printed shipping label. Therefore, it could not be inspected in the process after packaging.

The main problems of existing methods are at least one of the following: (1) they cannot inspect the barcode and the address in the shipping label simultaneously, (2) they support only a limited number of shipping label formats, (3) they do not verify whether the address actually exists.

In this chapter, we propose a verification and recognition method for various types of shipping labels by using deep neural networks. To effectively detect barcode and address areas, a deep learning object detection algorithm, You Only Look Once (YOLO) [83], is fine-tuned for the shipping label dataset and the angle of the image is calibrated by the angle estimated from the barcode. Moreover, we use Google Maps API [164] to check the address validity. To train deep neural networks, a number of shipping label data are required. For that purpose, we generated and collected 25 different types of shipping labels to train and test the proposed method.

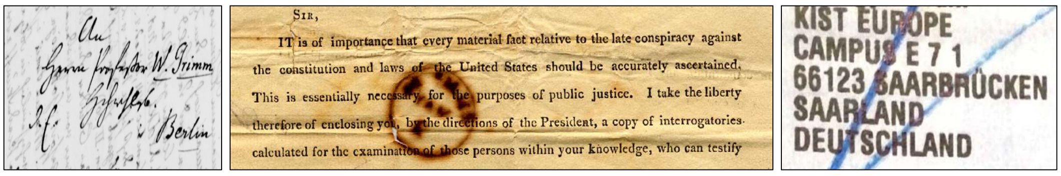
6.1.3 Text recognition with Image Enhancement

Document image enhancement is one of the most important preprocessing steps for the tasks involved in analyzing an imaged document. The main aim of document image enhancement is to extract the foreground text from the degraded background in the document image, a process called image binarization. The better binarization methods will improve the performance of subsequent document analysis tasks [165] such as layout analysis [166], text line and word segmentation [167], and optical character recognition (OCR) [168]. The challenge is that document images normally suffer from various types of degradation and interference, such as page stains, uneven illumination, contrast variation, ink bleed, paper yellowing, background color, bleed-through, and environmental deterioration [169, 170].

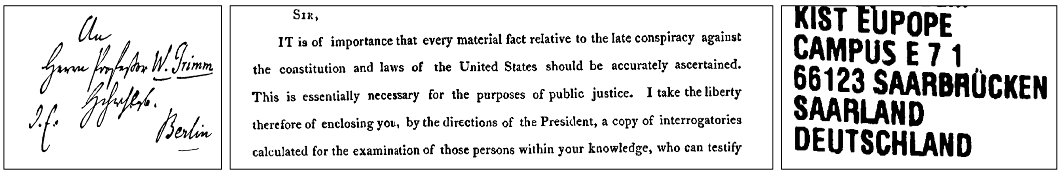
Over the past few years, many document image enhancement methods have been proposed in the literature, such as those that improve image quality by removing degradation effects and artifacts present in an image to restore its original appearance [171, 172]. However, traditional unsupervised document enhancement methods rarely handle multiple degradations. Most focus on a single specific problem, such as bleed-through [173, 174].

Recently, deep-learning-based image binarization methods have provided significant improvements over traditional image processing methods, not simply solving a specific problem but eliminating multiple degradations [175, 176, 177]. A selectional auto-encoder model for document image binarization was proposed by Calvo-Zaragoza and Gallego [178]. To generate binarized output the same size as the input image, a fully convolutional neural network (FCN) was used, generating binary results as a special case of semantic segmentation [179, 180, 181]. To improve the performance of the binarization for a degraded document, several approaches have been proposed, including a hierarchical deep supervised network [160], an iterative supervised network [159], and a conditional generative-adversarial-network-based binarization method [182], which outperformed traditional methods and other deep-learning-based methods.

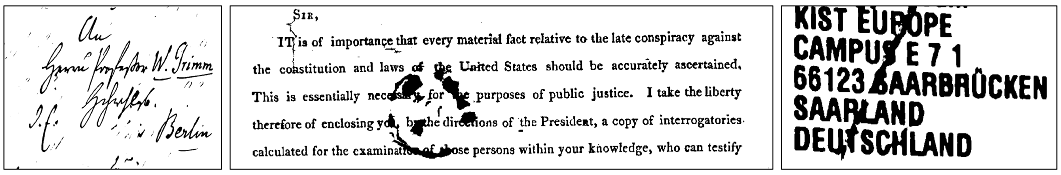
However, all of these methods focus only on grayscale document images because the scanned historical documents that most of them are designed to handle are contaminated document images in black and white. Although these methods remove



(a)



(b)



(c)

Figure 6.3: Examples of the degraded document and their challenges in image binarization. (a) some of the example images from DIBCO dataset and shipping label image dataset, (b) the corresponding ground truth images, (c) the corresponding binarization results by Vo's method.

colorless degradation factors well, they are weaker in extracting the target from colorful backgrounds and in removing colored contaminants. Furthermore, most of the methods are based solely on context information in a particular neighborhood region, using small local image patches instead of the whole image as the input to the deep learning network. For regions of densely placed text, local features in local image patches are advantageous for extracting text from backgrounds, whereas the spatial contextual information can sometimes be missed for regions with a largely blank background. These two major problems are exemplified in Figure 6.3.

Taking these problems into consideration, we propose a framework of generative adversarial networks for color document image enhancement and binarization. The proposed method consists of two stages, the first of which trains multiple color channel adversarial neural networks to extract color foreground information from small local image patches by removing background information for document image enhancement, and the second of which learns the local and global binary results with multi-scale adversarial neural networks for document image binarization. In the first stage, four color-independent networks are trained with red, green, blue, and gray channel images, respectively. In the second stage, the local binary transformation network is trained with the merged enhanced images from the first stage, and the global binary transformation network is trained with the full original input image.

The contributions of this study can be summarized as follows. (1) Unlike previous methods, the proposed method focuses on the multi-color degradation problem with its design of four color-independent networks. (2) By combining local and global binary transformation networks, the method can balance the fine extraction of strokes and the suppression of background misclassification. (3) The proposed method is evaluated on multiple datasets and is found to achieve better performance than state-of-the-art models.

6.2 Shipping Label Recognition and Validation

6.2.1 Shipping Label Recognition and Validation System

In the packing and delivery process, it requires fast and accurate detection and inspection algorithm for various types of labels. For this reason, we employed deep neural networks as there exist several outstanding methods to detect a specific object in an image. The overview of the proposed system is shown in Figure 6.4. The proposed system consists of five steps. The first three steps are to detect the barcode and address in the shipping label image, the fourth step is to recognize the text of the receiver address, and the last step is to validate the recognized address.

As the first step, barcode detection is processed using YOLO. YOLO is known as a fast general-purpose object detector using deep neural networks and achieved real-time detection [161], which is much faster than R-CNN [183] and Faster R-CNN [184] based methods. Unlike conventional region-based object detection algorithms, the YOLO detector considers object detection as an end-to-end regression problem and uses a single convolutional network to predict the bounding boxes and the corresponding class probabilities. We fine-tune the YOLO detector to locate the barcode and address areas in the shipping label images. We exploit the YOLO network implemented on the Darknet19 [185] framework with the resized input image of size 416×416.

Though the regions of barcode and address are correctly identified, the accuracy of the text recognition result for the address area can be low when the cropped address area contains angular transformation. The machines in the package manufacturing process or the product transfer process in logistics could cause this undesired rotation. In order to improve the text recognition, a calibration procedure of the input image with respect to the angular error should be preceded as the second step. Since the address and the barcode are generally oriented in the same direction in the shipping label image, the input image can be calibrated from the angle measurement of the cropped barcode area. Because the barcode consists of several perpendicular lines, the angle of the barcode can be obtained through a line detection algorithm using the Hough transform.

After conducting rectification for angular error, the YOLO detector is applied once again to identify the sections of the barcode and address in the corrected image. In

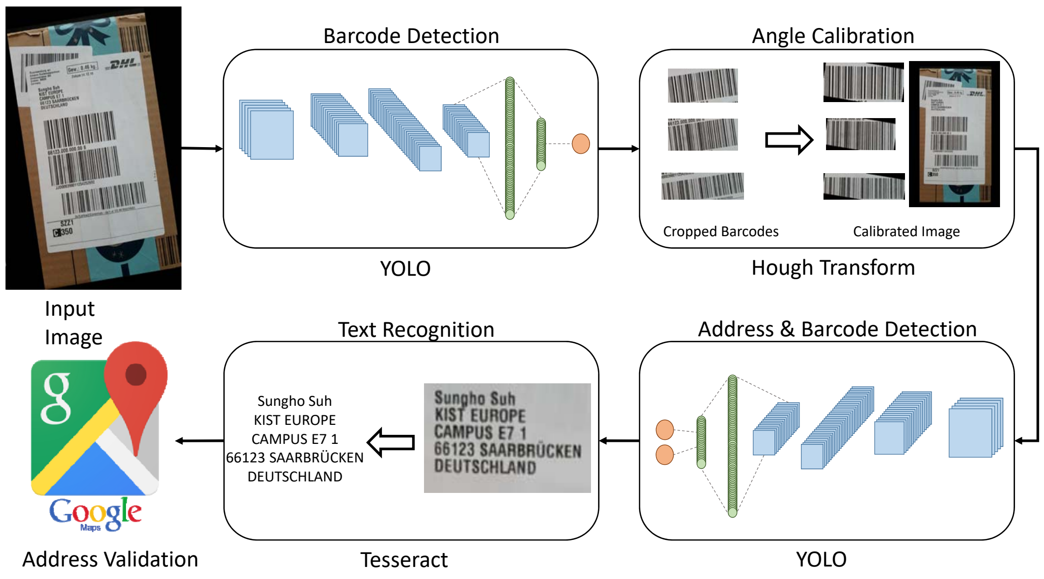


Figure 6.4: System overview

this way, some address areas not detected in the first YOLO step can be detected. Additionally, since the detected areas are now upright, the accuracy of the barcode decoding and the address text recognition can be improved. We use Code Reader of Matrox Imaging Library (MIL) [186] for the barcode decoding.

To implement text recognition for the receiver address, we utilize Tesseract [168]. Tesseract recognizes characters based on Long Short Term Memory (LSTM), which is a type of Recurrent Neural Network (RNN). This engine provides a huge database set consisting of 116 languages. Since the accuracy of the text recognition by Tesseract decreases as the angular transformation becomes severe, the calibration of the cropped address image has significant importance. The comparison result with different angular error severities is shown in Section 6.4.

To verify whether the recognized address actually exists, we check the validity of the address using Google Maps API [164] in the final step. Google Maps API judges the address as valid if the names of street, city, and country of the address are recognized. If the recognized address is invalid, it cannot be searched and Google Maps API returns an invalid status value. However, the recognition result from the previous step contains the receiver's name or the building in numerous instances. In order to prevent such false-negative cases, we check the address to Google Maps API repeatedly, by removing a word from the beginning of each trial, until we find the valid address or we encounter the end of the text.

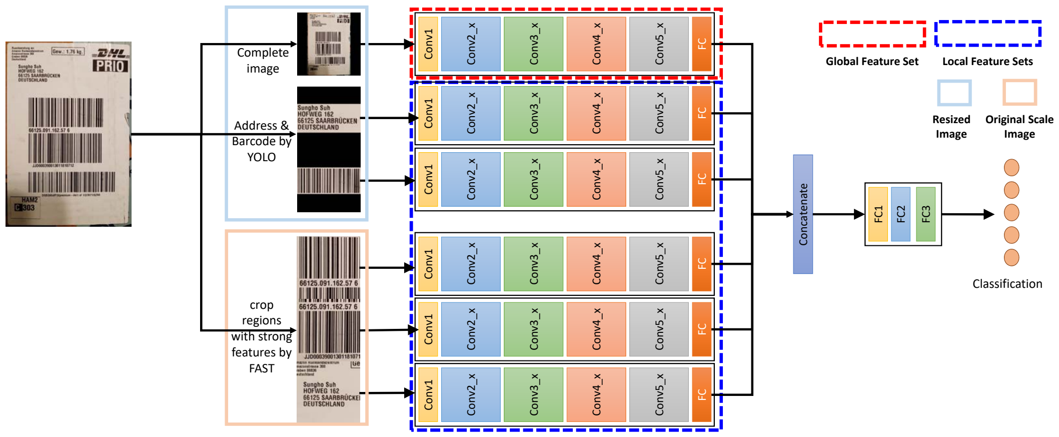


Figure 6.5: Overall network architecture of the proposed method

6.2.2 Fusion of Global-Local Features for Image Quality Inspection of Shipping Label

The shipping label inspection system [43] involves six steps: input image quality verification, calibration, salient region detection, image enhancement, text recognition, and address validation.

In the first step, the proposed input image quality verification process classifies the five image types: normal, contaminated address, unreadable image, handwritten address, and damaged shipping label. The unreadable image and the damaged label are reported to the user to acquire the image properly or check the condition of the shipping label. The contaminated and handwritten addresses provide information on the address area for image enhancement and/or text recognition processes.

The goal of the input image quality verification method is to classify the five image types. We propose an input image quality verification method using convolutional networks with global and local features. The overall network architecture of the proposed method is presented in Figure 6.5.

Feature Localization

Classification with a resized entire input image leads to ambiguity between normal and blurred images because of the loss of detailed features, making it difficult to detect contaminated defects and damaged defects in ROIs. For these reasons, we detect barcodes and address areas and crop regions with strong features to extract local features. In this study, we train global and local CNNs to extract the global and local features and combine the global and local features, as shown in Figure 6.5.

To detect and localize the ROIs, we use YOLO to detect barcode and address areas and FAST to localize regions with strong features. YOLO is a fast general-purpose object detector using deep neural networks that can achieve real-time detection [161],

Algorithm 4 Feature localization using the FAST algorithm. We use the default threshold value of the FAST algorithm, $t = 50$, patch size $D_p = (256, 256)$, and number of patches $n_p = 3$

```
Input: input image of shipping label
Output: localized regions using FAST
 1: Convert input image to grayscale image
 2: Detect feature points $p$ using FAST with $t$
 3: $D_i$ =size of input image
 4: if $(D_i > D_p)$ then
 5:     $patchNum_i = D_i/D_p$
 6:     $L_p$ = number of $p$
 7:     for each $j \in [1, patchNum_i]$ do
 8:         $iCountPoint = 0$
 9:         $R_j$ = region of $j$-th patch
10:         for each $k \in [1, L_p]$ do
11:             if $(p(k) \in R_j)$ then
12:                 $iCountPoint = iCountPoint + 1$
13:             end if
14:         end for
15:         $iCountFeature(j) = iCountPoint$
16:     end for
17:     Extract the $n_p$ largest patches and crop the region $I_P$
18: else
19:     Copy the image to $I_P$ in the padding space of size $D_p$
20: end ifreturn $I_P$
```

which is much faster than R-CNN- [183] and Faster R-CNN [184]-based methods. Unlike conventional region-based object detection algorithms, the YOLO detector considers object detection as an end-to-end regression problem and uses a single convolutional network to predict the bounding boxes and the corresponding class probabilities. We fine-tune the YOLO detector to locate the barcode and address areas in the shipping label images. We exploit the YOLO network implemented on the Darknet19 [185] framework with a resized input image of size 416×416. After detecting the barcode and address regions, we crop and resize the detected regions to 256×256 while maintaining the aspect ratio in the padding space to set the input in the local CNNs.

The global features are extracted from the global CNNs with a resized input image, and the local features are extracted from independent local CNNs with the detected address and barcode area, and the localized regions using FAST. FAST is a famous corner-detection algorithm that can be used to extract feature points; it demonstrates rapid operation and a low number of computations compared with other feature-detection methods. We calculate the number of feature points from the FAST algorithm in a patch of size 256×256, the same as the input image size in the local CNNs. By choosing patches with a large number of feature points, regions with strong features, such as barcodes or characters, can be selected as input images of local CNNs. Additionally, because the patches are not resized, but rather maintain the original scale of the input image, they can provide helpful information to classify unreadable input images using the local CNNs.

Global and Local Feature Fusion

Global features describe the overall outline of the input image, whereas local features focus on the local shape, the degree of contamination in the localized regions, lighting conditions, and other imaging factors. To extract the global and local features, a resized complete input image is processed in the global CNNs, and the localized images are processed in the independent local CNNs, as shown in Figure 6.5. In this study, we deploy ResNet-50 [132] pre-trained on ImageNet [133] for global and local feature extractions. In the proposed architecture, four independent CNNs are trained for feature extraction: a global CNN, two local CNNs with address and barcode areas by YOLO, and a local CNN with regions cropped by FAST. The local CNNs with regions cropped by FAST share the network parameters in the training procedure.

To combine the global and local features, we adopt a stacked generalization ensemble. Stacked generalization is an ensemble method where a new model learns how to combine the predictions from multiple models [163]. To control the importance of the feature combination, we add a fully connected layer with different output feature dimensions to the feature extraction architecture. In the proposed model, we set the dimension of the global feature to 2048 and the dimension of each local feature to 512. To verify the proposed model, we compare the proposed ensemble models with majority voting and weighted majority voting algorithms [187].

6.3 Two-Stage Generative Adversarial Networks for Document Image Binarization

Loss functions of the proposed GAN The original purpose of GAN [66] was to train generative models to capture real data distributions and to generate an output image from random noise. A generator network competes against a discriminator network that distinguishes between generated and real images. Unlike the original GAN, CGAN [67] trains the generator not only to fool the discriminator but also to condition on additional inputs, such as class labels, partial data, or input images. By conditioning the generative model on input images, CGAN can be used for pixel-level image-to-image transfer [92]. Previous approaches have provided better results by mixing the objective loss function of the generator with some traditional losses:

$$\min_G \max_D \mathbb{E}_{x,y}[\log D(y,x)] + \mathbb{E}_x[\log(1 - D(G(x),x))] + \lambda\, \mathbb{E}_{x,y}[\|y - G(x)\|_1], \quad (6.1)$$

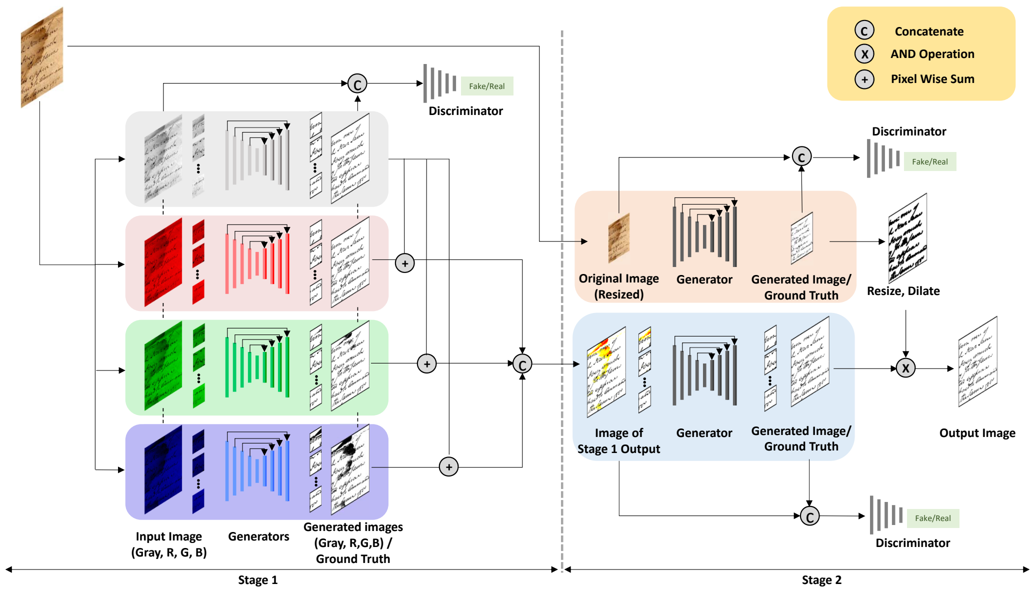


Figure 6.6: The structure of the proposed model for document image enhancement (Stage 1) and document image binarization (Stage 2).

where x is an input image sampled from the input data distribution $\mathbb{P}_x$, y is the ground truth image corresponding to the input image, λ is a hyperparameter that increases the effect of regularization on a model, and the generator G generates an image $G(x)$ from an input image. Equation (6.1) shows that minimization of L1 distance, rather than L2 distance, between the generated image and the ground truth image, encourages less blurring and ambiguity in the generation process [92].

Although GAN and CGAN are capable of generating fake images close to the original input images or target images, the GAN training procedure has instability of loss function convergence [188]. To solve this stability problem, we apply the Wasserstein GAN with gradient penalty (WGAN-GP) [71], which uses the Wasserstein-K distance as the loss function, to the objective function to guide the training process. Furthermore, unlike the general image-to-image transfer task [92], our task, document image enhancement and binarization, is to primarily classify every individual pixel into two classes, text and background. Thus, we decided to use the binary cross-entropy (BCE) loss rather than L1 loss as used in previous approaches [92, 182]. Bartusiak et al. [189] showed experimentally that BCE loss is indeed a better choice than L1 loss for binary classification. The objective loss functions of conditional WGAN-GP with the BCE loss are defined as follows:

$$\mathbb{L}_D(x, y; \theta_D) = -\mathbb{E}_{x,y}[D(y, x)] + \mathbb{E}_x[D(G(x), x)] + \alpha\, \mathbb{E}_{x,\hat{y}\sim P_{\hat{y}}}[(\|\nabla_{\hat{y}} D(\hat{y}, x)\|_2 - 1)^2], \tag{6.2}$$

$$\begin{aligned} \mathbb{L}_G(x, y; \theta_G) &= \mathbb{E}_x[D(G(x), x)] + \lambda \, \mathbb{L}_{BCE}(G(x), y), \\ \text{where } \mathbb{L}_{BCE}(p, q) &= \mathbb{E}_{p,q}[q \log p + (1-q) \log(1-p)], \end{aligned} \tag{6.3}$$

where α is the penalty coefficient; $P_{\hat{y}}$ is the uniform sampling along straight lines between pairs of points from the ground truth distribution P_y and the generated data distribution; λ controls the relative importance of different loss terms; and θ_D and θ_G are parameters of the discriminator and the generator, respectively. Whereas the discriminator D is trained to minimize $\mathbb{L}_D$ for distinguishing between the ground truth and the generated image, the generator G is trained to minimize $\mathbb{L}_G$.

Network architecture of the proposed GAN The GAN architecture has two neural networks, and we selected a generator and a discriminator for image binarization performance. As the encoder in the generator, U-Net [150] is employed. U-Net is a network structure that introduces skip concatenation between the encoder and the decoder layers; it provides good performance in image segmentation. The encoder includes downsampling to extract the context information, and the decoder is an upsampling process that combines the upsampled features and the low-dimensionality features from the downsampling layer to improve the performance of the network. In binarization studies, U-Net is widely adopted [92, 159, 182, 190]. To extract important features effectively in the proposed adversarial neural networks, we adopted EfficientNet [191] as the encoder in the generator; it has achieved much better accuracy and efficiency for image classification than other networks. As the discriminator, the discriminator network in PatchGAN [192, 92, 193, 194, 182, 190] is employed with the modification. PatchGAN has been used frequently in recent work due to its good generalization properties. We employ a network with an architecture similar to that of the discriminator network from the discriminator, based on a Markov random field model, of Pix2Pix GAN [92].

The overall network architecture of the proposed method is shown in Figure 6.6. The first stage of the proposed method, shown on the left side of the figure, consists of four color-independent generators and a discriminator to distinguish between the ground truth and the generated image. Four color-independent networks are trained with red, green, blue, and gray channel images, respectively, and produce an enhanced document image by removing background color information. Each channel image and the corresponding ground truth or generated image are concatenated and fed into the discriminator. In the second stage, shown on the right side of the figure, the local binary transformation network is trained with the merged enhanced images resulting from the first stage, and the global binary transformation network is trained with the full original input image for document image binarization.

6.3.1 Document Image Enhancement using Color-Independent Adversarial Networks

The goal is to enhance the degraded color document and extract the text regions from the image. Owing to the diversity and complexity of the degraded color document images, it is difficult to extract the text regions in the variety of conditions of degradation and printing by training only one generator or by training with a grayscale image. In other words, it cannot sensitively separate the foreground from the color background which has a different color, but similar intensity values to the foreground. Instead of generating a binary image directly from a three-channel color input image or a grayscale input image, we first train four color-independent generators, which focus on extracting color information and removing the color background. Four color-independent networks are trained with split three-channel images and a converted grayscale image.

Before training the proposed GAN framework in the first stage, a pre-processing step for ground truth images of three color input images is required. If a given ground truth image were to be used directly to train four different generators, the corresponding networks would not be able to extract the proper color information and could not be trained to classify the text regions and the background regions from the corresponding color channel owing to the lack of information. To generate ground truth images for three different color images, we combine a split input channel image and a given ground truth image via the logical AND operator. After combining the ground truth image and the split channel image, we generate a binary image via the application of the global threshold. Unlike the three split color images, the given ground truth image is used directly for training the grayscale generator with the converted grayscale image.

After the pre-processing step for ground truth images, the four generators and one discriminator are trained to extract colored text regions from the background regions with the input images and their corresponding ground truth images. During training, we divide each document image into small patches of size 256×256 without resizing. The four generators are trained to fool the discriminator and minimize the distance between the generated image and the ground truth image in each color channel by minimizing the objective loss function $\mathbb{L}_G$ in (6.3), and the discriminator is trained to distinguish between the generated image and the ground truth image by minimizing the objective loss function $\mathbb{L}_D$ in (6.2).

The four generated images $G_r(x_r)$, $G_g(x_g)$, $G_b(x_b)$, and $G_{gray}(x_{gray})$ are predicted for each image patch via the four proposed generators. To integrate the information in the four images predicted by the four-color channel generators, we apply pixel-wise addition between $G_{gray}(x_{gray})$ and the three generated color images $G_r(x_r)$, $G_g(x_g)$, and $G_b(x_b)$, respectively, and merge the three-channel images into a color image. The reason why the grayscale image is combined with each color channel is to prevent the generator to be biased to each color channel. By applying pixel-wise addition between $G_{gray}(x_{gray})$ and the three generated color images, it enables to obtain a result that reflects the characteristics from each color channel based on the grayscale image result. The training details for the proposed method are summarized in Algorithm 5.

Algorithm 5 Training procedure for document image enhancement using adversarial networks. We use default values of $\omega = 0.5$, $\alpha = 10$, $\lambda = 50$

```
Require: Batch size $m$, Adam hyperparameters $\eta$, hyperparameter for $\lambda$.
 1: Initialize: $\theta_{G_r}$, $\theta_{G_g}$, $\theta_{G_b}$, $\theta_{G_{gray}}$ from pre-trained source networks.
 2: for number of training iterations do
 3:     Sample $\{x^{(i)}\}_{i=1}^{m}$ a batch from the input image patches and corresponding
        ground truth $\{y^{(i)}\}_{i=1}^{m}$.
 4:     for $k = \{r, g, b, gray\}$ do
 5:         if $k$ is not gray then
 6:             $y_k \leftarrow x_k \bigcap y$
 7:             Binarize $y_k$ with threshold value $t$.
 8:         else
 9:             $y_k \leftarrow y$
10:         end if
11:         Update discriminator $D$ by descending the gradient of (6.2):
12:         $\theta_D \leftarrow \theta_D - \eta_D \nabla_{\theta_D} \mathbb{L}_D(x_k, y_k; \theta_D)$
13:         Update generator $G_k$ by descending the gradient of (6.3):
14:         $\theta_{G_k} \leftarrow \theta_{G_k} - \eta_G \nabla_{\theta_{G_k}} \mathbb{L}_G(x_k, y_k; \theta_{G_k})$
15:     end for
16: end for
17: for $k = \{r, g, b\}$ do
18:     $\hat{y}_k \leftarrow \omega G_k(x_k) + (1 - \omega) G_{gray}(x_{gray})$
19: end for
20: $\hat{y} \leftarrow [\hat{y}_r, \hat{y}_g, \hat{y}_b]$
```

6.3.2 Document Image Binarization using Multi-scale Feature Fusion

As shown on the right side of Figure 6.6, the second stage combines the global and local results of the document binarization. The binarization in the first stage is performed mainly using local prediction with the small patches. However, for a document image having a large background region portion, local prediction can sometimes misclassify parts of the background region as foreground. To address this problem, we perform the global binarization with the resized original input image and the local binarization with the image result from the first stage. The reason for using the resized original input image is that the image result from the first stage can have a loss of spatial contextual information of the document image since the first stage is only performed using local prediction. The input image for the generator and the binary image, which is the corresponding ground truth image or the generated image, are concatenated and fed into the discriminator. Whereas the input image for the generators is an 8-bit image in the first stage, in the second stage it is a 24-bit three-channel image. Furthermore, the input image for the local prediction is the small patches of the output image from the first stage, whereas the input image for the global prediction is the resized original input image. Thus, the second stage for multi-scale feature fusion requires two discriminators, whereas the first stage trains only one shared discriminator with four independent generators.

For the local and global binarizations, the network architecture of the generator is the same as that in the first stage except for the input image channel. The degradation document image datasets, such as DIBCO and the shipping label dataset, contain images of various sizes and height-to-width ratios. In the local binarization process, regions of different colors can be removed from text components, and text-only regions can be extracted by using the image result from the document image enhancement stage. For the global binarization, we add padding to the whole image or directly resize the original image to the desired size (r, r). The strategies of global binarization are as follows according to the ratio of the input image height h and width w:

- **If** $\frac{\max(w,h)}{\min(w,h)} < 4.0$, resize the whole image to (r, r).
- **If** $4.0 \leq \frac{\max(w,h)}{\min(w,h)} < 6.0$, put the input image at the center of $(\max(w, h), \max(w, h))$, add padding with the median pixel value of the input image set to $(\max(w, h), \max(w, h))$, and resize the image of $(\max(w, h), \max(w, h))$ to (r, r).
- **If** $\frac{\max(w,h)}{\min(w,h)} \geq 6.0$ or $(h < r$ and $w < r)$, do not perform global binarization.

The threshold values were determined through various testing. However, the threshold values do not significantly affect the binarization results if the threshold values for the range are not changed much. Morphological dilation is applied to the global prediction map so that the global map can cover all of the text regions because the resizing process can remove textual components. Lastly, the global binarization image is incorporated into the local binarization map via the logical AND operator to remove any misclassified noise in the background.

6.4 Experimental Results

6.4.1 Datasets and Evaluation Metrics

To the best of our knowledge, there is no open database for shipping labels because of the privacy problem of customers. Because of the scarcity of shipping label images, we generated and collected 5306 and 1092 images of different types and from various countries using smartphones. Note that we collected real data with the agreement of the owner, and we plan to release the dataset onto our public website. The dataset includes images of the shipping labels, the position information of the barcode and address, and five annotated labels: normal, contaminated, unreadable, handwritten, and damaged. Five people manually annotated the dataset with specific sorting criteria. In the case of the “unreadable” class, the receiver’s address is blurred or distorted, or the resolution is too low to recognize the address. For the “contaminated” and handwritten class, the receiver’s address is contaminated, occluded by

Table 6.1: Manually annotated dataset of the shipping labels

Defect Type	# of Images (Generated)	# of Images (Collected)
Normal	1283	660
Contaminated	1054	139
Unreadable	904	107
Handwritten	988	52
Damaged	1077	134
Total	5306	1092

pen mark, or handwritten. Finally, a damaged shipping label indicates that the shipping label is torn or occluded by another shipping label. Detailed information about the dataset is presented in Table 6.1. To evaluate the text recognition, the dataset can be separated into two groups: one written in the Latin alphabet and the other in Korean. In detail, the dataset includes the images of the shipping labels, the location information of the barcode and the receiver address, and the ground truth address.

To further evaluate the performance of the proposed input image quality verification and image calibration methods, we obtained the shipping label formats from 15 different carrier companies, generated barcodes randomly, and entered random addresses from seven different countries. The label images were generated in various sizes using NiceLabel Designer software [195]. To evaluate the proposed method, we generated contaminated, unreadable, and damaged images using image-processing methods, and then we generated handwritten labels using Google handwriting fonts [196]. Finally, there were 5306 generated images for input image quality verification. We selected 100 images per class as the test set and conducted a 10-fold cross-validation procedure for the dataset.

The collected dataset contains only 1092 images and is under a data-imbalanced condition, while the generated dataset has 5306 images and the number of data for each class is similar. To improve the performance of the classification on the collected real dataset, a data augmentation method, such as rotation (90, 180, 270 degrees) and flip, is adopted to extend the data of minority classes. To evaluate the proposed method using deep neural networks, we conducted a 10-fold cross-validation procedure for the shipping label dataset.

For quantitative evaluation and comparison with the state-of-the-art algorithms, we adopted four evaluation metrics, which are used in the image binarization competition, Document Image Binarization Contest (DIBCO): F-measure (FM), pseudo-F-measure (p-FM), peak signal-to-noise ratio (PSNR), and the distance reciprocal distortion (DRD) metric. Additionally, to evaluate the degree to which the proposed method improves OCR performance on degraded document images, we adopted the Levenshtein distance [197] expressed in percent. We measured the Levenshtein distance only for images from the shipping label image dataset, for which the ground truth is given.

- **F-measure (FM)**:

$$FM = \frac{2 \times Recall \times Precision}{Recall + Precision}, \tag{6.4}$$

where $Recall = \frac{TP}{TP+FN}$, $Precision = \frac{TP}{TP+FP}$, and TP, FP, and FN denote the true positive, false positive, and false negative values, respectively.

- **Pseudo-F-measure (p-FM)**:

$$p - FM = \frac{2 \times pRecall \times Precision}{pRecall + Precision}, \tag{6.5}$$

where $pRecall$ is the percentage of the skeletonized ground truth image [198].

- **Peak signal-to-noise ratio (PSNR)**:

$$PSNR = 10 \log \left(\frac{C^2}{MSE} \right), \tag{6.6}$$

where $MSE = \frac{\sum_{x=1} M \sum_{y=1}^{N} (L(x,y) - L'(x,y))^2}{MN}$, and C denotes the difference between text and background. $PSNR$ is a measure of the similarity between two images.

- **Distance reciprocal distortion (DRD) metric**:

$$DRD = \frac{\sum_k DRD_k}{NUBN}, \tag{6.7}$$

where DRD_k is the distortion of the kth flipped pixel, and $NUBN$ is the number of non-uniform 8×8 blocks in the ground truth image [199].

- **Levenshtein distance expressed in percent (Lev)**:

$$Lev(s_1, s_2) = \left(1 - \frac{d(s_1, s_2)}{\max(|s_1|, |s_2|)} \right) \times 100, \tag{6.8}$$

where $d(s_1, s_2)$ denotes the Levenshtein distance between two strings s_1, s_2 (of length $|s_1|$ and $|s_2|$, respectively).

6.4.2 Results

Experimental Results by Input Image Quality Verification

We evaluate the proposed method on the generated dataset in this subsection. Table 6.2 shows the performance of the feature localization by YOLO. The results are the mean average precision [200] and show mean and standard deviation. The feature localization on the generated dataset provided a high performance, above 0.98, and the example images of the detection results are shown in Fig 6.7 (a).

Table 6.2: Detection accuracy by YOLO

Region	mAP (Generated)	mAP (Collected)
Address	0.9653 ± 0.1637	0.8907 ± 0.0215
Barcode	0.9976 ± 0.0018	0.9778 ± 0.0065
Total	0.9814 ± 0.0085	0.9343 ± 0.0105

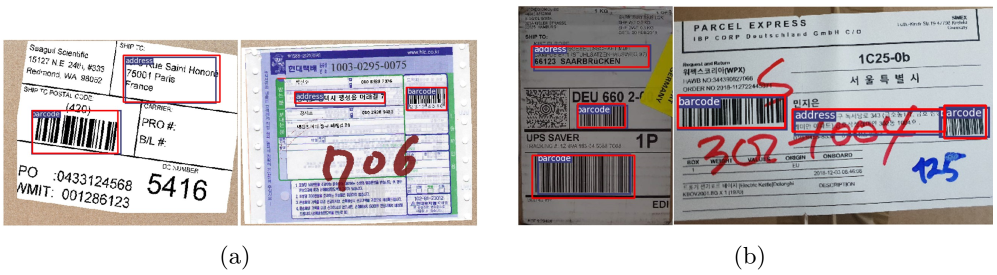


Figure 6.7: Examples of detected barcode and address areas (a) on generated images and (b) on the real dataset: red boxes are ground truth, and blue boxes are detected regions

Table 6.3: Comparison of classification results on the generated real dataset

Methods	VGG-19 Accuracy	Resnet-50 Accuracy
Only global features	95.98 ± 0.74 %	95.80 ± 0.38 %
Global-local fusion (majority voting)	96.40 ± 0.65%	96.62 ± 0.43%
Global-Local fusion (weighted majority voting)	97.16 ± 0.43 %	97.02 ± 0.77 %
Global-Local fusion (ours)	98.32 ± 0.49%	**99.06 ± 0.66%**

For the evaluation, VGG-19 and ResNet-50 were used for classification, and we compared the proposed method by combining the global and local features of the classification methods with only global features and other ensemble learning methods, such as majority voting and weighted majority voting. Table 6.3 shows the classification accuracy compared with respect to the generated dataset. The proposed method with ResNet-50 provided the highest classification accuracy of 98.46 %, which exceeded ResNet-50 with only global features and other ensemble learning methods by 3.46% and 2.04%, respectively.

We evaluate the proposed method on collected real datasets compared with other classification models with only global features and other ensemble learning methods. Compared with the generated dataset, the collected real dataset has more image types, and it is more difficult to extract local features and classify the input image quality. Table 6.2 shows the performance of the feature localization by YOLO. The performance of the feature localization on the collected real images is 0.0471 lesser

Table 6.4: Comparison of classification results on the collected real dataset

Methods	VGG-19 Accuracy	Resnet-50 Accuracy
Only global features	85.40 ± 2.43 %	86.00 ± 3.40 %
Global-local fusion (majority voting)	84.67 ± 2.27 %	86.40 ± 2.97 %
Global-Local fusion (weighted majority voting)	85.00 ± 2.68 %	87.60 ± 2.70 %
Global-Local fusion (ours)	87.80 ± 2.13%	**89.26 ± 2.70%**

than the performance on the generated images. Example images of the detection results are shown in Fig 6.7 (b).

Because the collected real dataset is under an imbalanced data condition, we select 30 images per class as the test set and conduct a 10-fold cross-validation procedure for the dataset. Table 6.4 presents the classification accuracy comparison on the collected real images. It can be observed that ResNet-50 has better performance than VGGNet-19, and the classification with ensemble learning methods also provided higher accuracy than the classification with only global features. This result means that the feature localization method that we proposed improves the classification of the input image quality. In all the methods in Table 6.4, our method with ResNet-50 achieved the highest classification accuracy of 89.26 %, combining the global and local features by the stacked generalization method. It improved the classification accuracy by 3.26 % and 2.86 % compared with ResNet-50 with only global features and other ensemble learning methods. Unlike other ensemble learning methods, the proposed method concatenates the global and local features and classifies them via three fully connected layers while maintaining the global and local features. However, the classification accuracy on the collected real dataset is lesser than that on the generated dataset. The collected real dataset has more image types and more serious defects. Furthermore, unlike the generated dataset, in which an image is matched one-on-one with a type of defect class, the collected real dataset may have a case where a single image corresponds to a variety of defect types. For example, when a damaged image is occluded by a pen mark and acquired with blurred focus, the image has three types of defects.

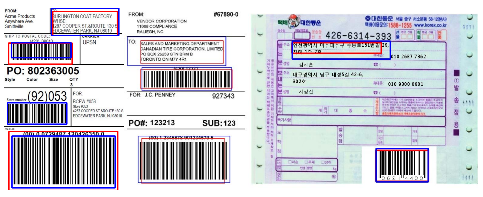
Table 6.5 shows the accuracy of each class on the collected real dataset and improvement of the classification by the global-local fusion method. Owing to the detection of address and barcode areas by YOLO, the classification accuracy for the contaminated class was improved by 6.7%. In addition, the classification accuracy for the unreadable class was improved by 5.0% by the local features extracted by the FAST algorithm and by maintaining the original scale.

Experimental Results by Image Calibration

To test how much the text recognition can benefit from the angle calibration, the dataset is artificially rotated to two cases: 1) randomly rotated within 10 degrees,

Table 6.5: Accuracy of each class on the collected real dataset and comparison with the proposed method with ResNet-50 with only global features

Class	Only global features	Global-local fusion (ours)
Normal	93.33 %	95.00 %
Contaminated	78.33 %	85.00 %
Unreadable	83.33 %	88.33 %
Handwritten	98.33 %	98.33 %
Damaged	76.67 %	79.31 %
Total	86.00 %	89.26 %


(a)

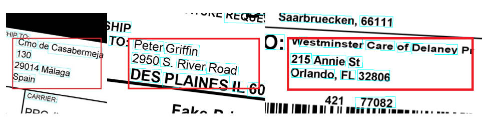

(b)

Figure 6.8: Examples of detected barcode and address areas (a) by the proposed method: blue boxes are ground truth and red boxes are detected areas, (b) by EAST: red boxes are detected addresses and sky blue boxes are detected texts

2) and within 20 degrees. We compare the results of the segmentation and text recognition before and after the angle calibration. The proposed method for the address detection and the text recognition is compared with the automatic text recognition method [41] using EAST [39]. Even though the method using EAST only detects the text area, it is known that it gives the best performance for text detection among the currently available schemes.

Table 6.6 shows the average accuracy results of the barcode and address detection. YOLO-AC is the proposed method that uses YOLO with the angle calibration, which is compared to YOLO without the angle calibration and EAST. The three methods performed similarly on the original dataset and the dataset with minor rotations. However, the accuracy significantly decreases in the case of the dataset rotated within 20 degrees. Among the three methods, our method gives the highest detection accuracy on the dataset with severe rotation errors. The example images of the detection results by the proposed method are shown in Figure 6.8 (a) and the example image of the detection results by EAST are shown in Figure 6.8 (b).

After the address area detection, we evaluate the text recognition accuracy with the cropped address image obtained from three different detection algorithms. Since the dataset is composed of the two-letter types, the Latin alphabet, and Korean, the results of the character recognition for the address area are evaluated separately. As illustrated in Table 6.7 and Figure 6.9, the recognition accuracy after YOLO without

Table 6.6: Detection accuracy comparison

		Total	Barcode	Address
YOLO	original	96.21%	98.05%	97.18%
	10°	95.59%	97.45%	97.17%
	20°	84.48%	91.72%	90.00%
EAST	original	92.44%	-	92.44%
	10°	91.19%	-	91.19%
	20°	76.58%	-	76.58%
YOLO-AC	10°	95.06%	97.93%	96.49 %
	20°	91.78%	94.66%	93.62 %

Table 6.7: Address recognition accuracy comparison

		Total	Alphabet	Korean
YOLO	original	89.87%	94.06%	84.56%
	10°	39.18%	53.36%	26.45%
	20°	26.57%	35.95%	17.32%
EAST	original	87.24%	93.16%	82.29%
	10°	73.18%	88.07%	59.90%
	20°	66.87%	87.85%	50.42%
YOLO-AC	10°	82.93%	96.27%	71.74 %
	20°	83.24%	94.93%	72.63 %

the angle calibration for both Latin alphabet and Korean dramatically decreases as the rotation angle becomes larger. Similarly, the recognition accuracy on EAST results also decreases and the accuracy for the Latin alphabet is also lower than 90%. However, the proposed method for the Latin alphabet gives an accuracy of around 95% even if the images are rotated up to 20 degrees. Our method on Korean data results in at least 12 percent point higher performance than other methods but shows a reduced accuracy compared to the Latin alphabet because the innate performance of the recognition model for Korean provided in Tesseract is low.

Experimental Results by Image Enhancement

The shipping label image dataset contains images of shipping labels with the receiver's address area and the ground truth of the addresses and their text regions. The ground truth is manually masked and can be extracted for validation of the text recognition. In this subsection, we evaluate the proposed method and the state-of-the-art methods by the document image binarization metrics and the Levenshtein distance expressed in percent to measure the extent to which the improved binarization affects the quality of text recognition. For OCR of the text in the shipping addresses, we utilized Tesseract [168], which recognizes characters based on long short-term memory (LSTM) [201]. This engine supports a very large database with

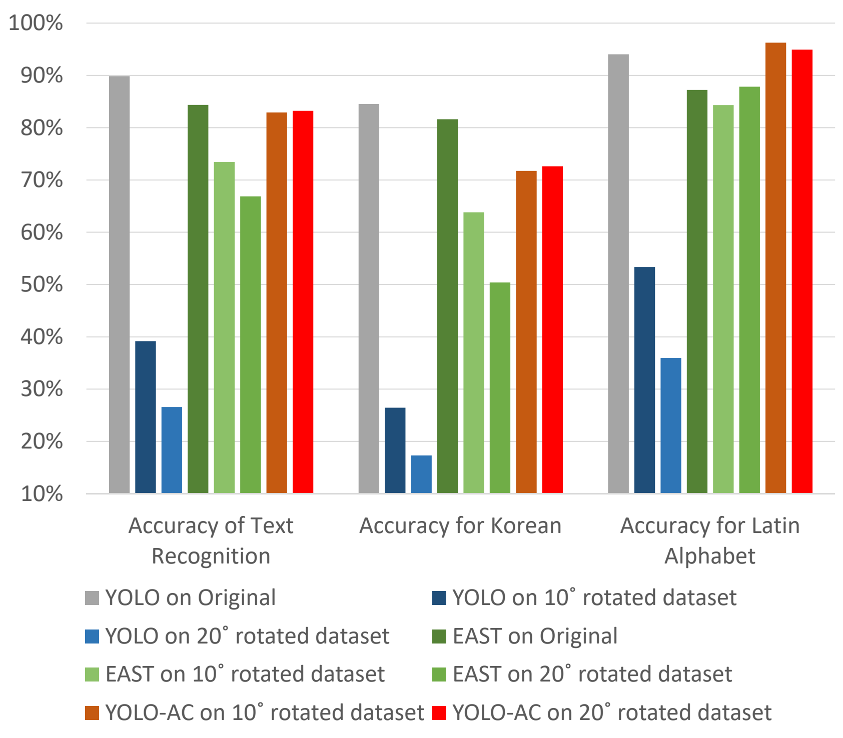


Figure 6.9: Character recognition accuracy for address

Table 6.8: Evaluation of document image binarization on shipping label image dataset.

Methods	FM	p-FM	$PSNR$	DRD
Otsu [175]	88.31	89.42	14.73	6.17
Niblack [176]	86.61	89.46	13.59	6.61
Sauvola [177]	87.67	89.53	14.18	5.75
Vo [160]	91.20	92.92	16.14	2.20
He [159]	91.09	92.26	16.03	2.33
Zhao [182]	92.09	93.83	16.29	2.37
Ours	**94.65**	**95.94**	**18.02**	**1.57**

116 different languages. The shipping label image dataset was collected from eight countries and can be separated into two groups: one of the labels written in the Latin alphabet and the other in Korean. The results of the OCR for the binarized address images were evaluated separately for the two language types.

Table 6.8 shows the mean values of the evaluation measures across the five-fold cross-validation procedure. Even though the shipping label dataset images contain

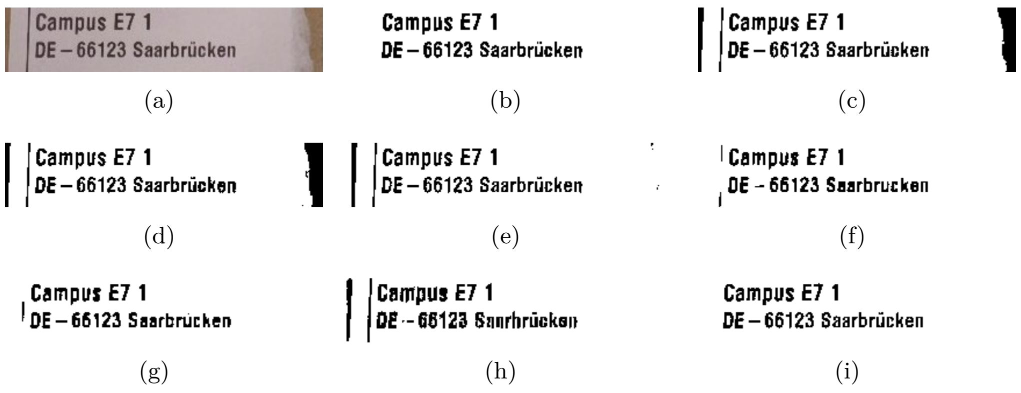


Figure 6.10: Binarization results of the address image Ger142 from the shipping label image dataset. (a) original images, (b) the ground truth, (c) Otsu, (d) Niblack, (e) Sauvola, (f) Vo, (g) He, (h) Zhao, (i) Ours.

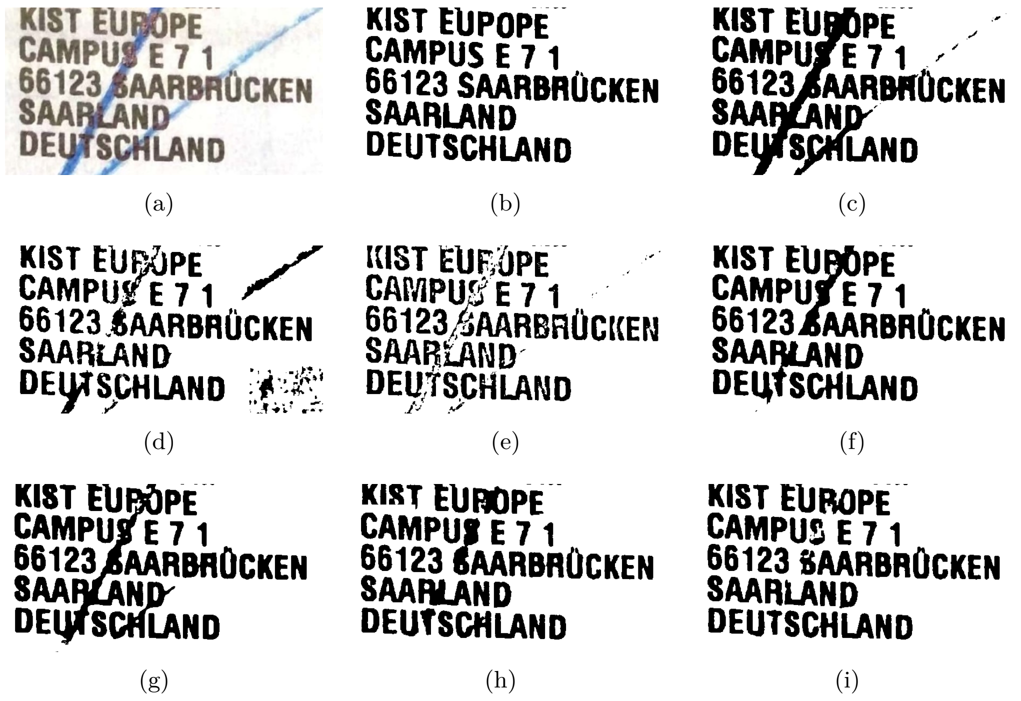


Figure 6.11: Binarization results of the address image Ger177 from the shipping label image dataset. (a) original images, (b) the ground truth, (c) Otsu, (d) Niblack, (e) Sauvola, (f) Vo, (g) He, (h) Zhao, (i) Ours.

colorful backgrounds and contaminants with colors unlike the other datasets, the proposed method outperformed the other methods in terms of all of the metrics. Figs. 6.10, 6.11, 6.12, and 6.13 show results of the image binarization and text

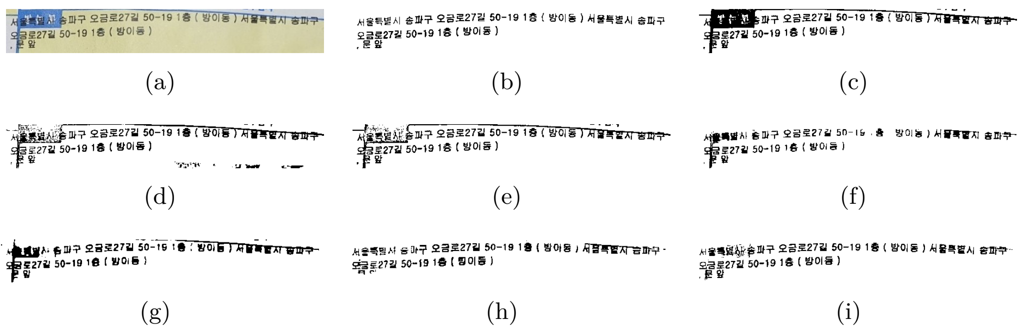


Figure 6.12: Binarization results of the address image Kor19 from the shipping label image dataset. (a) original images, (b) the ground truth, (c) Otsu, (d) Niblack, (e) Sauvola, (f) Vo, (g) He, (h) Zhao, (i) Ours.

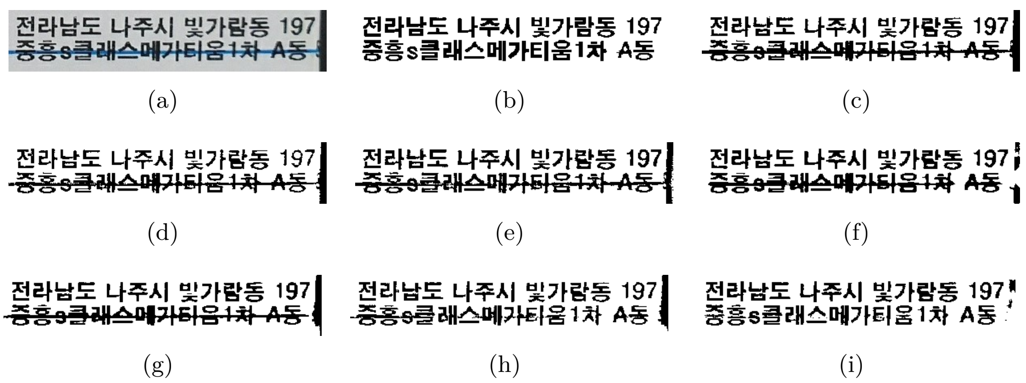


Figure 6.13: Binarization results of the address image Kor230 from the shipping label image dataset. (a) original images, (b) the ground truth, (c) Otsu, (d) Niblack, (e) Sauvola, (f) Vo, (g) He, (h) Zhao, (i) Ours.

region extraction; it can be seen that the deep-learning-based methods can extract the text regions from the background regions, in contrast to the traditional binarization methods. The proposed method can remove the colorful backgrounds and contaminants in the shipping label images, whereas the state-of-the-art methods handle the color information improperly. Table 6.9 shows the average OCR accuracy results using Levenshtein distance expressed in percent. The proposed method improved the accuracy of the input images by 6.20 percentage points. Moreover, the proposed method provided the best performance on both the Korean and Latin alphabet datasets.

To measure the efficiency, the run time on the shipping label image dataset was computed. Table 6.10 shows the average run time of the different methods on the shipping label image dataset. The average run time of the deep-learning-based methods was much greater than that of the traditional image processing methods. Of

Table 6.9: OCR accuracy comparison in Levenshtein distance in percent on the shipping label image dataset.

Methods	Total	Korean	Alphabet
Input image	77.20	73.86	94.47
Ground Truth	87.62	85.88	96.66
Otsu [175]	74.45	70.72	93.79
Niblack [176]	69.00	66.31	82.94
Sauvola [177]	72.84	68.81	93.73
Vo [160]	77.14	74.69	89.86
He [159]	75.15	72.45	89.13
Zhao [182]	77.33	74.56	91.69
Ours	**83.40**	**81.15**	**95.09**

Table 6.10: Average runtime of different methods on the shipping label image dataset in milliseconds

Method	Runtime
Otsu [175]	1.5
Niblack [176]	350.2
Sauvola [177]	15.1
Vo [160]	139.0
He [159]	856.2
Zhao [182]	131.8
Ours	282.8

the deep-learning-based methods, the proposed method took more time than Vo et al.'s and Zhao et al.'s methods but was significantly faster than He and Schomaker's method.

As the final step, we check the recognized address using Google Maps API to verify the address. Most of the addresses in the shipping label dataset are valid ones, but we generated around 10% of the data with invalid addresses. We found that those invalid addresses are correctly identified as erroneous status. Therefore, the cost, such as undeliverable situations and returned products, could be reduced as the proposed method is applied to the inspection machine for logistics or manufacturing companies.

6.5 Conclusions

In this chapter, robust shipping label verification and recognition algorithms for logistics by using deep neural networks have been proposed. The proposed system comprised three steps: 1) input image quality verification, 2) address and barcode areas detection with image calibration, 3) address text recognition with image enhancement. For the input image quality verification, we have presented an input image quality verification method using CNNs combining global and local features for shipping label inspection. We detected and localized the regions of interest using YOLO and FAST, and we extracted the global and local features using several independent CNNs. To combine the global and local features, we adopted a stacked generalization ensemble. Secondly, In order to detect the barcode and address areas and to calibrate the angular error in a shipping label image, we deployed YOLO and measured the angular error from the cropped barcode image by using Hough Transform. Lastly, we have proposed a color document image enhancement and binarization method using multi-scale adversarial neural networks. The proposed method focuses on the multi-color degradation problem with its design consisting of four color-independent networks and combines the local and global binary transformation networks.

The experimental results showed that the proposed input image quality verification method improved the classification accuracy by 3.46% and 3.86%, respectively, compared with the classification model with only global features, and by 2.04% and 2.86%, respectively, compared with the other ensemble learning method combining global and local features. The proposed image calibration method improved the address recognition accuracy by 43 percent point and 59 percent point higher than YOLO without the angle calibration, and by 8 percentage points and 7 percentage points higher than EAST. Furthermore, we evaluated the proposed image enhancement method on the shipping label image dataset. The experimental results demonstrated that the proposed method outperformed traditional and state-of-the-art methods. As the last stage, the recognized address was validated using the Google Maps API. As the invalid addresses were successfully detected, the cost, such as undeliverable situations and returned products, could be reduced.

In future work, the proposed method will be applied to a packaging machine with an industrial camera and will be tested in the logistics industry. For instance, we can deploy the proposed method for automated shipping address recognition and validation based on the enhanced image binarization result, which will apply to the packaging machine and logistics industry. The shipping label image dataset we evaluated in this study will be published on the website for use in further research by the computer vision community.

7 Conclusion

The goal of the dissertation is to improve the performance of the classification under imbalanced data conditions by using GAN and transfer learning. The proposals were deployed to fault detection and diagnosis for rotating machinery, and shipping label recognition and validation for logistics, as real industrial applications. This final chapter summarizes with an overview and achievements of the dissertation and the future directions are briefly discussed.

7.1 Summary of Thesis Achievements

In Chapter 3, a novel data augmentation method named CEGAN, which is composed of three independent networks, was proposed. A novel GAN structure containing a classifier, that induced the generated data, had more features for classification. The classifier had the functionality of reducing the impact of noise input and the ambiguity between classes. Comprehensive experimental results on various benchmark datasets demonstrated that the proposed method achieved promising performance in terms of quantity and quality of data augmentation compared with many classical and state-of-the-art algorithms.

In Chapter 4, a novel feature generation method using GAN structure to unravel the data imbalanced problem and improve neural network performance was proposed. In the proposed GAN structure, the feature extractor and the feature classifier were included to train together with the feature generator and the feature discriminator. For the generation of meaningful features for the classification of small-sized target data, transfer learning with regularization and class-wise attention was adopted. The proposed DFG was evaluated with various pairs of source and target datasets to show general applicability to classification in the class-imbalanced conditions. The experimental results showed that the DFG generator enhanced the augmentation of the label-preserved and diverse features, and the classification results were significantly improved on the target task. The feature generation model could contribute greatly to the development of data augmentation methods through discriminative feature generation and supervised attention methods.

In Chapter 5, a novel feature extraction method to predict the RUL and detect faults of the bearings and a generative oversampling method for imbalanced data on bearing FDD was proposed. The proposed approach was comprised of three algorithms: 1) a supervised HS division using CNN for bearing wear, 2) generalized multiscale feature extraction and RUL prediction method using multiscale GAN, 3) a generative oversampling method to address the data imbalance issue for bearing FDD. The proposed supervised HS division with the combination of NSP and

CNN-HS effectively distinguished healthy and unhealthy states. Secondly, the novel multiscale feature extraction method was designed for the HS division and the RUL prediction. We formulated one-dimensional feature extraction as a principal signal separation task and introduced the use of U-Net to reconstruct the prognostic features for the RUL prediction. The novel domain-invariant generalized solution based on the GAN scheme was introduced to learn the invariant representation and predict the RUL. Lastly, the generative oversampling method generated fault images using DCWGAN-GP to overcome the data imbalance problem. The experimental results showed that the proposed method improved the classification accuracy could capture the degrading patterns effectively, and achieved better RUL prediction results compared to other methods.

In Chapter 6, robust shipping label verification and recognition algorithms for logistics by using deep neural networks have been proposed. The proposed system comprised three steps: 1) input image quality verification, 2) address and barcode areas detection with image calibration, 3) address text recognition with image enhancement. For the input image quality verification, we have presented an input image quality verification method using CNNs combining global and local features for shipping label inspection. To combine the global and local features, a stacked generalization ensemble was adopted. Secondly, To detect the barcode and address areas and to calibrate the angular error in a shipping label image, we deployed YOLO and measured the angular error from the cropped barcode image by using Hough Transform. Lastly, we have proposed a color document image enhancement and binarization method using multi-scale adversarial neural networks. The proposed method focuses on the multi-color degradation problem with its design consisting of four color-independent networks and combines the local and global binary transformation networks. The experimental results showed that the proposed input image verification system improved the classification accuracy and the proposed image enhancement method outperformed traditional and state-of-the-art methods. As the last stage, the recognized address was validated using the Google Maps API. As the invalid addresses were successfully detected, the cost, such as undeliverable situations and returned products, could be reduced.

7.2 Future Outlook

In the future, we will extend the proposed method to real-world problems and contemplate further accuracy improvement in the feature generation method. Although the proposed CEGAN and DFG methods improved the performance of the classification under the class-imbalanced conditions on several generally used datasets, we believe there is room for improvement as a gap still exists between our method and the balanced condition.

For the bearing fault detection, we plan to develop the fault prediction model with an attention mechanism and unsupervised transfer learning, because supervised data with degradation is difficult to collect in real industrial scenarios.

Furthermore, the proposed shipping label recognition system will be applied to a packaging machine with an industrial camera and will be tested in the logistics industry. The shipping label image dataset we evaluated in this study will be published on the website for use in further research by the computer vision community.

Bibliography

[1] X. Li, H. Xiong, H. Wang, Y. Rao, L. Liu, and J. Huan, “DELTA: deep learning transfer using feature map with attention for convolutional networks,” in *International Conference on Learning Representations*, 2019.

[2] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, “Recent advances in convolutional neural networks,” *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[3] N. Japkowicz and S. Stephen, “The class imbalance problem: A systematic study,” *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[4] J. Xie and Z. Qiu, “The effect of imbalanced data sets on LDA: A theoretical and empirical analysis,” *Pattern recognition*, vol. 40, no. 2, pp. 557–562, 2007.

[5] M. Buda, A. Maki, and M. A. Mazurowski, “A systematic study of the class imbalance problem in convolutional neural networks,” *Neural Networks*, vol. 106, pp. 249–259, 2018.

[6] M. Baptista, S. Sankararaman, I. P. de Medeiros, C. Nascimento Jr, H. Prendinger, and E. M. Henriques, “Forecasting fault events for predictive maintenance using data-driven techniques and arma modeling,” *Computers & Industrial Engineering*, vol. 115, pp. 41–53, 2018.

[7] R. K. Singleton, E. G. Strangas, and S. Aviyente, “The use of bearing currents and vibrations in lifetime estimation of bearings,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1301–1309, 2016.

[8] G. Z. Vachtsevanos, *Intelligent fault diagnosis and prognosis for engineering systems.* John Wiley & Sons, 2006.

[9] N. Li, Y. Lei, J. Lin, and S. X. Ding, “An improved exponential model for predicting remaining useful life of rolling element bearings,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 12, pp. 7762–7773, 2015.

[10] Y. Qian, R. Yan, and R. X. Gao, “A multi-time scale approach to remaining useful life prediction in rolling bearing,” *Mechanical Systems and Signal Processing*, vol. 83, pp. 549–567, 2017.

[11] M. Pecht and J. Gu, “Physics-of-failure-based prognostics for electronic products,” *Transactions of the Institute of Measurement and Control*, vol. 31, no. 3-4, pp. 309–322, 2009.

[12] J. Zarei, M. A. Tajeddini, and H. R. Karimi, "Vibration analysis for bearing fault detection and classification using an intelligent filter," *Mechatronics*, vol. 24, no. 2, pp. 151–157, 2014.

[13] X. Guo, L. Chen, and C. Shen, "Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis," *Measurement*, vol. 93, pp. 490–502, 2016.

[14] M. E. H. Benbouzid and G. B. Kliman, "What stator current processing-based technique to use for induction motor rotor faults diagnosis?" *IEEE Transactions on Energy Conversion*, vol. 18, no. 2, pp. 238–244, 2003.

[15] J. A. Antonino-Daviu, M. Riera-Guasp, M. Pineda-Sanchez, and R. B. Perez, "A critical comparison between dwt and hilbert–huang-based methods for the diagnosis of rotor bar failures in induction machines," *IEEE Transactions on Industry Applications*, vol. 45, no. 5, pp. 1794–1803, 2009.

[16] B. Zhang, C. Sconyers, C. Byington, R. Patrick, M. E. Orchard, and G. Vachtsevanos, "A probabilistic fault detection approach: Application to bearing fault detection," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 5, pp. 2011–2018, 2010.

[17] A. P. Ompusunggu, J.-M. Papy, and S. Vandenplas, "Kalman-filtering-based prognostics for automatic transmission clutches," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 1, pp. 419–430, 2015.

[18] M. Jouin, R. Gouriveau, D. Hissel, M.-C. Péra, and N. Zerhouni, "Particle filter-based prognostics: Review, discussion and perspectives," *Mechanical Systems and Signal Processing*, vol. 72, pp. 2–31, 2016.

[19] C. Chen, G. Vachtsevanos, and M. E. Orchard, "Machine remaining useful life prediction: An integrated adaptive neuro-fuzzy and high-order particle filtering approach," *Mechanical Systems and Signal Processing*, vol. 28, pp. 597–607, 2012.

[20] F. Deng, S. Guo, R. Zhou, and J. Chen, "Sensor multifault diagnosis with improved support vector machines," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 1053–1063, 2015.

[21] X. Gu, F. Deng, X. Gao, and R. Zhou, "An improved sensor fault diagnosis scheme based on ta-lssvm and ecoc-svm," *Journal of Systems Science and Complexity*, vol. 31, no. 2, pp. 372–384, 2018.

[22] C. Li, J. V. De Oliveira, M. Cerrada, D. Cabrera, R. V. Sánchez, and G. Zurita, "A systematic review of fuzzy formalisms for bearing fault diagnosis," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 7, pp. 1362–1382, 2018.

[23] E. T. Esfahani, S. Wang, and V. Sundararajan, “Multisensor wireless system for eccentricity and bearing fault detection in induction motors,” *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 3, pp. 818–826, 2013.

[24] Y. O. Lee, J. Jo, and J. Hwang, “Application of deep neural network and generative adversarial network to industrial maintenance: A case study of induction motor fault detection,” in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 3248–3253.

[25] Z. Chen and W. Li, “Multisensor feature fusion for bearing fault diagnosis using sparse autoencoder and deep belief network,” *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 7, pp. 1693–1702, 2017.

[26] Y. Qin, X. Wang, and J. Zou, “The optimized deep belief networks with improved logistic sigmoid units and their application in fault diagnosis for planetary gearboxes of wind turbines,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 5, pp. 3814–3824, 2018.

[27] S. Tang, C. Shen, D. Wang, S. Li, W. Huang, and Z. Zhu, “Adaptive deep feature learning network with nesterov momentum and its application to rotating machinery fault diagnosis,” *Neurocomputing*, vol. 305, pp. 1–14, 2018.

[28] H. Oh, J. H. Jung, B. C. Jeon, and B. D. Youn, “Scalable and unsupervised feature engineering using vibration-imaging and deep learning for rotor system diagnosis,” *IEEE Transactions on Industrial Electronics*, vol. 65, no. 4, pp. 3539–3549, 2017.

[29] S. Guo, T. Yang, W. Gao, and C. Zhang, “A novel fault diagnosis method for rotating machinery based on a convolutional neural network,” *Sensors*, vol. 18, no. 5, p. 1429, 2018.

[30] S. Wang, J. Xiang, Y. Zhong, and Y. Zhou, “Convolutional neural network-based hidden markov models for rolling element bearing fault identification,” *Knowledge-Based Systems*, vol. 144, pp. 65–76, 2018.

[31] T. Liu and G. Li, “The imbalanced data problem in the fault diagnosis of rolling bearing,” *Comput. Eng. Sci.*, vol. 32, no. 5, pp. 150–153, 2010.

[32] E. Ramentol, Y. Caballero, R. Bello, and F. Herrera, “SMOTE-RSB*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory,” *Knowledge and information systems*, vol. 33, no. 2, pp. 245–265, 2012.

[33] W. W. Ng, J. Hu, D. S. Yeung, S. Yin, and F. Roli, “Diversified sensitivity-based undersampling for imbalance classification problems,” *IEEE transactions on cybernetics*, vol. 45, no. 11, pp. 2402–2412, 2014.

[34] X. Lu, M. Chen, J. Wu, and P. Chan, "A Feature-Partition and Under-Sampling Based Ensemble Classifier for Web Spam Detection," *International Journal of Machine Learning and Computing*, vol. 5, no. 6, p. 454, 2015.

[35] U. S. P. S. O. of Inspector General, "Undeliverable as addressed costs more than you think," https://www.uspsoig.gov/, 2015.

[36] Z. Zhang, W. Shen, C. Yao, and X. Bai, "Symmetry-based text line detection in natural scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2558–2567.

[37] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2315–2324.

[38] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 11, pp. 2298–2304, 2017.

[39] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "East:an efficient and accurate scene text detector," *The IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2642–2651, 2017.

[40] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, "Character region awareness for text detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9365–9374.

[41] F. D. S. Ribero, L. Gong, F. Caliva, M. Swainson, K. Gudmundsson, M. Yu, G. Leontidis, X. Ye, and S. Kollias, "An end-to-end deep neural architecture for optical character verification and recognition in retail food packaging," *The IEEE International Conference on Image Processing (ICIP)*, pp. 2376–2380, 2018.

[42] easypost, "easypost shipping api," https://www.easypost.com/docs/api/.

[43] S. Suh, H. Lee, Y. O. Lee, P. Lukowicz, and J. Hwang, "Robust shipping label recognition and validation for logistics by using deep neural networks," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 4509–4513.

[44] S. Suh, P. Lukowicz, and Y. O. Lee, "Fusion of global-local features for image quality inspection of shipping label," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 2643–2649.

[45] G. Van Horn, O. Mac Aodha, Y. Song, A. Shepard, H. Adam, P. Perona, and S. Belongie, "The inaturalist challenge 2017 dataset," *arXiv preprint arXiv:1707.06642*, vol. 1, no. 2, 2017.

[46] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, “Sun database: Large-scale scene recognition from abbey to zoo,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* IEEE, 2010, pp. 3485–3492.

[47] B. A. Johnson, R. Tateishi, and N. T. Hoan, “A hybrid pansharpening approach and multiscale object-based image analysis for mapping diseased pine and oak trees,” *International journal of remote sensing*, vol. 34, no. 20, pp. 6969–6982, 2013.

[48] M. Kubat, R. C. Holte, and S. Matwin, “Machine learning for the detection of oil spills in satellite radar images,” *Machine learning*, vol. 30, no. 2-3, pp. 195–215, 1998.

[49] O. Beijbom, P. J. Edmunds, D. I. Kline, B. G. Mitchell, and D. Kriegman, “Automated annotation of coral reef survey images,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 2012, pp. 1170–1177.

[50] J. W. Grzymala-Busse, L. K. Goodwin, W. J. Grzymala-Busse, and X. Zheng, “An approach to imbalanced data sets based on changing rule strength,” in *Rough-neural computing.* Springer, 2004, pp. 543–553.

[51] B. Mac Namee, P. Cunningham, S. Byrne, and O. I. Corrigan, “The problem of bias in training data in regression problems in medical decision support,” *Artificial intelligence in medicine*, vol. 24, no. 1, pp. 51–70, 2002.

[52] Q. Wang, X. Zhou, C. Wang, Z. Liu, J. Huang, Y. Zhou, C. Li, H. Zhuang, and J.-Z. Cheng, “Wgan-based synthetic minority over-sampling technique: Improving semantic fine-grained classification for lung nodules in ct images,” *IEEE Access*, vol. 7, pp. 18 450–18 463, 2019.

[53] S. Suh, H. Lee, J. Jo, P. Lukowicz, and Y. O. Lee, “Generative Oversampling Method for Imbalanced Data on Bearing Fault Detection and Diagnosis,” *Applied Sciences*, vol. 9, no. 4, p. 746, 2019.

[54] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, “Learning from class-imbalanced data: Review of methods and applications,” *Expert Systems with Applications*, vol. 73, pp. 220–239, 2017.

[55] X.-M. Zhao, X. Li, L. Chen, and K. Aihara, “Protein classification with imbalanced data,” *Proteins: Structure, function, and bioinformatics*, vol. 70, no. 4, pp. 1125–1132, 2008.

[56] S. Graves, G. Asner, R. Martin, C. Anderson, M. Colgan, L. Kalantari, and S. Bohlman, “Tree species abundance predictions in a tropical agricultural landscape with a supervised classification model and imbalanced data,” *Remote Sensing*, vol. 8, no. 2, p. 161, 2016.

[57] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?" in *2016 international conference on digital image computing: techniques and applications (DICTA).* IEEE, 2016, pp. 1–6.

[58] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[59] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[60] P. Jeatrakul, K. W. Wong, and C. C. Fung, "Classification of imbalanced data by combining the complementary neural network and smote algorithm," in *International Conference on Neural Information Processing.* Springer, 2010, pp. 152–159.

[61] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge & Data Engineering*, no. 9, pp. 1263–1284, 2008.

[62] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing.* Springer, 2005, pp. 878–887.

[63] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence).* IEEE, 2008, pp. 1322–1328.

[64] L. Lusa *et al.*, "Evaluation of smote for high-dimensional class-imbalanced microarray data," in *2012 11th International Conference on Machine Learning and Applications*, vol. 2. IEEE, 2012, pp. 89–94.

[65] A. Holzinger, *Machine Learning for Health Informatics: State-of-the-Art and Future Challenges.* Springer, 2016, vol. 9605.

[66] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[67] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[68] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70.* JMLR. org, 2017, pp. 2642–2651.

[69] G. Douzas and F. Bacao, "Effective data generation for imbalanced learning using conditional generative adversarial networks," *Expert Systems with applications*, vol. 91, pp. 464–471, 2018.

[70] S. Suh, H. Lee, P. Lukowicz, and Y. O. Lee, "Cegan: Classification enhancement generative adversarial networks for unraveling data imbalance problems," *Neural Networks*, vol. 133, pp. 69–86, 2021.

[71] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, 2017, pp. 5767–5777.

[72] X. Gao, F. Deng, and X. Yue, "Data augmentation in fault diagnosis based on the wasserstein generative adversarial network with gradient penalty," *Neurocomputing*, 2019.

[73] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[74] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[75] X. Li, Y. Grandvalet, and F. Davoine, "Explicit inductive bias for transfer learning with convolutional networks," in *International Conference on Machine Learning*, 2018, pp. 2825–2834.

[76] S. Suh, P. Lukowicz, and Y. O. Lee, "Discriminative feature generation for classification of imbalanced data," *Pattern Recognition*, vol. 122, p. 108302, 2022.

[77] S. Suh, J. Jang, S. Won, M. S. Jha, and Y. O. Lee, "Supervised health stage prediction using convolutional neural networks for bearing wear," *Sensors*, vol. 20, no. 20, p. 5846, 2020.

[78] S. Suh, P. Lukowicz, and Y. O. Lee, "Generalized multiscale feature extraction for remaining useful life prediction of bearings with generative adversarial networks," *arXiv preprint arXiv:2109.12513*, 2021.

[79] S. Suh, J. Kim, P. Lukowicz, and Y. O. Lee, "Two-stage generative adversarial networks for document image binarization with color noise and background removal," *arXiv preprint arXiv:2010.10103*, 2020.

[80] S. Suh, S. Cheon, D.-J. Chang, D. Lee, and Y. O. Lee, "Sequential lung nodule synthesis using attribute-guided generative adversarial networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention.* Springer, 2021, pp. 402–411.

[81] S. Suh, S. Cheon, W. Choi, Y. W. Chung, W.-K. Cho, J.-S. Paik, S. E. Kim, D.-J. Chang, and Y. O. Lee, "Supervised segmentation with domain adaptation for small sampled orbital ct images," *arXiv preprint arXiv:2107.00418*, 2021.

[82] J. Jang, Y. Kim, K. Choi, and S. Suh, "Sequential targeting: A continual learning approach for data imbalance in text classification," *Expert Systems with Applications*, vol. 179, p. 115067, 2021.

[83] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *The IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[84] T. Jo and N. Japkowicz, "Class imbalances versus small disjuncts," *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 40–49, 2004.

[85] D. Mease, A. J. Wyner, and A. Buja, "Boosted classification trees and class probability/quantile estimation," *Journal of Machine Learning Research*, vol. 8, no. Mar, pp. 409–439, 2007.

[86] R. C. Holte, L. Acker, B. W. Porter *et al.*, "Concept learning and the problem of small disjuncts." in *IJCAI*, vol. 89. Citeseer, 1989, pp. 813–818.

[87] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," in *Pacific-Asia conference on knowledge discovery and data mining.* Springer, 2009, pp. 475–482.

[88] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline over-sampling for imbalanced data classification," *International Journal of Knowledge Engineering and Soft Data Paradigms*, vol. 3, no. 1, pp. 4–21, 2011.

[89] S. Barua, M. M. Islam, X. Yao, and K. Murase, "Mwmote–majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 2, pp. 405–425, 2012.

[90] R. Alejo, V. García, and J. H. Pacheco-Sánchez, "An efficient over-sampling approach based on mean square error back-propagation for dealing with the multi-class imbalance problem," *Neural Processing Letters*, vol. 42, no. 3, pp. 603–617, 2015.

[91] W. A. Rivera, "Noise reduction a priori synthetic over-sampling for class imbalanced data sets," *Information Sciences*, vol. 408, pp. 146–161, 2017.

[92] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[93] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein generative adversarial networks,” in *International Conference on Machine Learning*, 2017, pp. 214–223.

[94] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie, “Stacked generative adversarial networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5077–5086.

[95] T. Guo, X. Zhu, Y. Wang, and F. Chen, “Discriminative sample generation for deep imbalanced learning.” in *IJCAI*, 2019, pp. 2406–2412.

[96] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, “Class-balanced loss based on effective number of samples,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9268–9277.

[97] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, “Are gans created equal? a large-scale study,” in *Advances in neural information processing systems*, 2018, pp. 700–709.

[98] A. Brock, J. Donahue, and K. Simonyan, “Large scale GAN training for high fidelity natural image synthesis,” in *International Conference on Learning Representations*, 2019.

[99] R. Volpi, P. Morerio, S. Savarese, and V. Murino, “Adversarial feature augmentation for unsupervised domain adaptation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5495–5504.

[100] Y. Zhang, B. Sun, Y. Xiao, R. Xiao, and Y. Wei, “Feature augmentation for imbalanced classification with conditional mixture wgans,” *Signal Processing: Image Communication*, vol. 75, pp. 89–99, 2019.

[101] R. Caruana, “Multitask learning,” *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[102] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, “Decaf: A deep convolutional activation feature for generic visual recognition,” in *International conference on machine learning*, 2014, pp. 647–655.

[103] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[104] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *arXiv preprint arXiv:1503.02531*, 2015.

[105] S. Zagoruyko and N. Komodakis, “Paying more attention to attention: Improving the performance of convolutional neural networks via attention

transfer," in *ICLR*, 2017. [Online]. Available: https://arxiv.org/abs/1612.03928

[106] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4133–4141.

[107] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[108] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[109] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[110] M. Lee and J. Seok, "Controllable generative adversarial network," *IEEE Access*, vol. 7, pp. 28 158–28 169, 2019.

[111] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[112] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[113] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[114] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms.* Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[115] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, p. 18, 2010.

[116] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," *arXiv preprint arXiv:1702.05373*, 2017.

[117] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[118] L. N. Darlow, E. J. Crowley, A. Antoniou, and A. J. Storkey, "Cinic-10 is not imagenet or cifar-10," *arXiv preprint arXiv:1810.03505*, 2018.

[119] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[120] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, “Maxout networks,” *arXiv preprint arXiv:1302.4389*, 2013.

[121] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[122] N. Qian, “On the momentum term in gradient descent learning algorithms,” *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[123] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, “Autoencoding beyond pixels using a learned similarity metric,” *arXiv preprint arXiv:1512.09300*, 2015.

[124] G. Mariani, F. Scheidegger, R. Istrate, C. Bekas, and C. Malossi, “Bagan: Data augmentation with balancing gan,” *arXiv preprint arXiv:1803.09655*, 2018.

[125] Z. Wang, E. P. Simoncelli, and A. C. Bovik, “Multiscale structural similarity for image quality assessment,” in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2. Ieee, 2003, pp. 1398–1402.

[126] K. Ma, Q. Wu, Z. Wang, Z. Duanmu, H. Yong, H. Li, and L. Zhang, “Group mad competition-a new methodology to compare objective image quality models,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1664–1673.

[127] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training gans,” in *Advances in neural information processing systems*, 2016, pp. 2234–2242.

[128] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” 2011.

[129] A. Coates, A. Ng, and H. Lee, “An analysis of single-layer networks in unsupervised feature learning,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[130] G. Griffin, A. Holub, and P. Perona, “Caltech-256 object category dataset,” 2007.

[131] L. Bossard, M. Guillaumin, and L. Van Gool, “Food-101–mining discriminative components with random forests,” in *European conference on computer vision.* Springer, 2014, pp. 446–461.

[132] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[133] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[134] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[135] S. Marcel and Y. Rodriguez, "Torchvision the machine-vision package of torch," in *Proceedings of the 18th ACM International Conference on Multimedia*, ser. MM '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 1485–1488. [Online]. Available: https://doi.org/10.1145/1873951.1874254

[136] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[137] L. Ren, J. Cui, Y. Sun, and X. Cheng, "Multi-bearing remaining useful life collaborative prediction: A deep learning approach," *Journal of Manufacturing Systems*, vol. 43, pp. 248–256, 2017.

[138] L. Guo, N. Li, F. Jia, Y. Lei, and J. Lin, "A recurrent neural network based health indicator for remaining useful life prediction of bearings," *Neurocomputing*, vol. 240, pp. 98–109, 2017.

[139] L. Ren, Y. Sun, J. Cui, and L. Zhang, "Bearing remaining useful life prediction based on deep autoencoder and deep neural networks," *Journal of Manufacturing Systems*, vol. 48, pp. 71–77, 2018.

[140] A. Z. Hinchi and M. Tkiouat, "Rolling element bearing remaining useful life estimation based on a convolutional long-short-term memory network," *Procedia Computer Science*, vol. 127, pp. 123–132, 2018.

[141] J. Zhu, N. Chen, and W. Peng, "Estimation of bearing remaining useful life based on multiscale convolutional neural network," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 4, pp. 3208–3216, 2018.

[142] X. Li, W. Zhang, and Q. Ding, "Deep learning-based remaining useful life estimation of bearings using multi-scale feature extraction," *Reliability Engineering & System Safety*, vol. 182, pp. 208–218, 2019.

[143] L. Ren, X. Cheng, X. Wang, J. Cui, and L. Zhang, "Multi-scale dense gate recurrent unit networks for bearing remaining useful life prediction," *Future Generation Computer Systems*, vol. 94, pp. 601–609, 2019.

[144] Y. Qin, D. Chen, S. Xiang, and C. Zhu, "Gated dual attention unit neural networks for remaining useful life prediction of rolling bearings," *IEEE Transactions on Industrial Informatics*, 2020.

[145] P. R. d. O. da Costa, A. Akcay, Y. Zhang, and U. Kaymak, "Remaining useful lifetime prediction via deep domain adaptation," *Reliability Engineering & System Safety*, vol. 195, p. 106682, 2020.

[146] M. Ragab, Z. Chen, M. Wu, C. K. Kwoh, and X. Li, "Adversarial transfer learning for machine remaining useful life prediction," in *2020 IEEE International Conference on Prognostics and Health Management (ICPHM)*. IEEE, 2020, pp. 1–7.

[147] M. Ragab, Z. Chen, M. Wu, C. S. Foo, C. K. Kwoh, R. Yan, and X. Li, "Contrastive adversarial domain adaptation for machine remaining useful life prediction," *IEEE Transactions on Industrial Electronics*, 2020.

[148] X. Li, W. Zhang, H. Ma, Z. Luo, and X. Li, "Data alignments in machinery remaining useful life prediction using deep adversarial neural networks," *Knowledge-Based Systems*, p. 105843, 2020.

[149] E. Principi, D. Rossetti, S. Squartini, and F. Piazza, "Unsupervised electric motor fault detection by using deep autoencoders," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 441–451, 2019.

[150] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[151] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[152] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin, "Machinery health prognostics: A systematic review from data acquisition to rul prediction," *Mechanical Systems and Signal Processing*, vol. 104, pp. 799–834, 2018.

[153] P. Wang, H. Wang, and R. Yan, "Bearing degradation evaluation using improved cross recurrence quantification analysis and nonlinear auto-regressive neural network," *IEEE Access*, vol. 7, pp. 38 937–38 946, 2019.

[154] K. Javed, R. Gouriveau, N. Zerhouni, and P. Nectoux, "A feature extraction procedure based on trigonometric functions and cumulative descriptors to enhance prognostics modeling," in *2013 IEEE Conference on Prognostics and Health Management (PHM)*. IEEE, 2013, pp. 1–7.

[155] B. Wang, Y. Lei, N. Li, and N. Li, "A hybrid prognostics approach for estimating remaining useful life of rolling element bearings," *IEEE Transactions on Reliability*, 2018.

[156] A. Veltman, D. W. Pulle, and R. W. De Doncker, *Fundamentals of electrical drives.* Springer, 2007.

[157] S. Guo, T. Yang, W. Gao, C. Zhang, and Y. Zhang, "An intelligent fault diagnosis method for bearings with variable rotating speed based on pythagorean spatial pyramid pooling cnn," *Sensors*, vol. 18, no. 11, p. 3857, 2018.

[158] H. Li, F. Zhu, and J. Qiu, "Cg-diqa: No-reference document image quality assessment based on character gradient," in *2018 24th International Conference on Pattern Recognition (ICPR).* IEEE, 2018, pp. 3622–3626.

[159] S. He and L. Schomaker, "Deepotsu: Document enhancement and binarization using iterative deep learning," *Pattern recognition*, vol. 91, pp. 379–390, 2019.

[160] Q. N. Vo, S. H. Kim, H. J. Yang, and G. Lee, "Binarization of degraded document images based on hierarchical deep supervised network," *Pattern Recognition*, vol. 74, pp. 568–586, 2018.

[161] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[162] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision.* Springer, 2006, pp. 430–443.

[163] J. Brownlee, *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions.* Machine Learning Mastery, 2018.

[164] M. P. Peterson, *Online maps with APIs and WebServices.* Springer Science & Business Media, 2012.

[165] H. Michalak and K. Okarma, "Region based adaptive binarization for optical character recognition purposes," in *2018 International Interdisciplinary PhD Workshop (IIPhDW).* IEEE, 2018, pp. 361–366.

[166] A. Antonacopoulos, C. Clausner, C. Papadopoulos, and S. Pletschacher, "Historical document layout analysis competition," in *2011 International Conference on Document Analysis and Recognition.* IEEE, 2011, pp. 1516–1520.

[167] N. Stamatopoulos, B. Gatos, G. Louloudis, U. Pal, and A. Alaei, "Icdar 2013 handwriting segmentation contest," in *2013 12th International Conference on Document Analysis and Recognition.* IEEE, 2013, pp. 1402–1406.

[168] R. Smith, “An overview of the tesseract ocr engine,” in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, vol. 2. IEEE, 2007, pp. 629–633.

[169] N. Kligler, S. Katz, and A. Tal, “Document enhancement using visibility detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2374–2382.

[170] A. Sulaiman, K. Omar, and M. F. Nasrudin, “Degraded historical document binarization: A review on issues, challenges, techniques, and future directions,” *Journal of Imaging*, vol. 5, no. 4, p. 48, 2019.

[171] R. F. Moghaddam and M. Cheriet, “A variational approach to degraded document enhancement,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 8, pp. 1347–1361, 2009.

[172] R. Hedjam and M. Cheriet, “Historical document image restoration using multispectral imaging system,” *Pattern Recognition*, vol. 46, no. 8, pp. 2297–2312, 2013.

[173] M. R. Yagoubi, A. Serir, and A. Beghdadi, “A new automatic framework for document image enhancement process based on anisotropic diffusion,” in *2015 13th International Conference on Document Analysis and Recognition (ICDAR).* IEEE, 2015, pp. 1126–1130.

[174] B. Sun, S. Li, X.-P. Zhang, and J. Sun, “Blind bleed-through removal for scanned historical document image with conditional random fields,” *IEEE Transactions on Image Processing*, vol. 25, no. 12, pp. 5702–5712, 2016.

[175] N. Otsu, “A threshold selection method from gray-level histograms,” *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[176] W. Niblack, “An introduction to digital image processing (englewood clivs, nj,” 1986.

[177] J. Sauvola and M. Pietikäinen, “Adaptive document image binarization,” *Pattern recognition*, vol. 33, no. 2, pp. 225–236, 2000.

[178] J. Calvo-Zaragoza and A.-J. Gallego, “A selectional auto-encoder approach for document image binarization,” *Pattern Recognition*, vol. 86, pp. 37–47, 2019.

[179] X. Peng, H. Cao, K. Subramanian, R. Prasad, and P. Natarajan, “Exploiting stroke orientation for crf based binarization of historical documents,” in *2013 12th International Conference on Document Analysis and Recognition.* IEEE, 2013, pp. 1034–1038.

[180] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[181] C. Tensmeyer and T. Martinez, "Document image binarization with fully convolutional neural networks," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 99–104.

[182] J. Zhao, C. Shi, F. Jia, Y. Wang, and B. Xiao, "Document image binarization with cascaded generators of conditional generative adversarial networks," *Pattern Recognition*, vol. 96, p. 106968, 2019.

[183] K. Ashraf, B. Wu, F. N. Iandola, M. W. Moskewicz, and K. Keutzer, "Shallow networks for high-accuracy road object-detection," *arXiv preprint arXiv:1606.01561*, 2016.

[184] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[185] J. Redmon, "Darknet: Open source neural networks in c," http://pjreddie.com/darknet/, 2013-2016.

[186] M. Imaging, *Matrox Imaging Library 9-UserGuide*, 2008, no. Y10513-301-0900.

[187] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *Journal of Machine Learning Research*, vol. 8, no. Dec, pp. 2755–2790, 2007.

[188] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[189] E. R. Bartusiak, S. K. Yarlagadda, D. Güera, P. Bestagini, S. Tubaro, F. M. Zhu, and E. J. Delp, "Splicing detection and localization in satellite imagery using conditional gans," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2019, pp. 91–96.

[190] R. De, A. Chakraborty, and R. Sarkar, "Document image binarization using dual discriminator generative adversarial networks," *IEEE Signal Processing Letters*, 2020.

[191] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, 2019, pp. 6105–6114.

[192] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *European conference on computer vision*. Springer, 2016, pp. 702–716.

[193] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[194] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[195] NiceLabel, "Designer," https://www.nicelabel.com/design-and-print/.

[196] Google, "Google fonts," https://fonts.google.com/.

[197] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8, 1966, pp. 707–710.

[198] I. Pratikakis, B. Gatos, and K. Ntirogiannis, "H-dibco 2010-handwritten document image binarization competition," in *2010 12th International Conference on Frontiers in Handwriting Recognition.* IEEE, 2010, pp. 727–732.

[199] K. Ntirogiannis, B. Gatos, and I. Pratikakis, "Icfhr2014 competition on handwritten document image binarization (h-dibco 2014)," in *2014 14th International conference on frontiers in handwriting recognition.* IEEE, 2014, pp. 809–813.

[200] T.-Y. Liu, "Learning to rank for information retrieval," *Found. Trends Inf. Retr.*, vol. 3, no. 3, p. 225–331, Mar. 2009. [Online]. Available: https://doi.org/10.1561/1500000016

[201] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

Sungho Suh

Department of Embedded Intelligence, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (DFKI)
67663 Kaiserslautern, Germany
+49 631 205752052 / sungho.suh@dfki.de

RESEARCH INTEREST

- Machine learning: Generative adversarial networks, discriminative feature generation
- Computer vision: Image segmentation, enhancement, object classification
- Sensing technology: human activity recognition, wearable computing, prognostics of systems, bearing fault detection, remaining useful life prediction
- Machine vision: Automatic visual inspection, optical character recognition

EDUCATION

Technische Universität Kaiserslautern, Kaiserslautern, Germany — Apr 2018 – Oct 2021
Ph.D., Department of Computer Science
- Dissertation: Improving Classification Performance under Imbalanced Data Conditions using Generative Adversarial Networks
- Advisor: Prof.Dr. Paul Lukowicz

Seoul National University, Seoul, Republic of Korea — Sep 2009 – Aug 2011
M.S., Department of Electrical Engineering and Computer Science
- Dissertation: Robust albedo estimation from a facial image with cast shadow
- Advisor: Prof. Chong-Ho Choi

Seoul National University, Seoul, Republic of Korea — Mar 2005 – Aug 2009
B.S., Department of Electrical and Computer Engineering

WORK EXPERIENCE

German Research Center for Artificial Intelligence (DFKI), Germany — Mar 2021 – Present
Researcher, Research Department of Embedded Intelligence
- STAR: Safe and Trusted Human Centric Artificial Intelligence in Future Manufacturing Lines
- VID-GEN SENSE: Generating synthetic wearable and ubiquitous sensor training data from videos and multimodal online information sources

Korea Institute of Science and Technology Europe, Saarbrücken, Germany — Mar 2018 – Feb 2021
Doctoral Researcher, Smart Convergence Group
- WALL-ET an autonomous robotic unit for transport tasks in logistics
- Developed smart multi-type continuous packaging system
- Enhanced Conditional Generative Adversarial Networks

Samsung Electro-Mechanics, Suwon, Republic of Korea — Aug 2011 – Mar 2018
Research Engineer, Global Technology Center
- Developed multi layer ceramic capacitor (MLCC) visual inspection system
- Developed Samsung Electro-Mechanics Image Processing Library
- Developed PCB via-hole inspection using laser
- Developed Automatic Optical Inspector of printed circuit board (PCB)

PUBLICATIONS

Journal

1. **Sungho Suh**, Paul Lukowicz, and Yong Oh Lee, “Discriminative feature generation for classification of imbalanced data”, *Pattern Recognition*, 2021.
2. Joel Jang, Yoonjeon Kim, Kyoungho Choi, and **Sungho Suh**, “Sequential Targeting: an incremental learning approach for data imbalance in text classification”, *Expert Systems with Applications*, vol. 179, 2021.
3. **Sungho Suh**, Haebom Lee, Paul Lukowicz, and Yong Oh Lee, “CEGAN: Classification Enhancement Generative Adversarial Networks for unraveling data imbalance problems”, *Neural Networks*, vol. 133, pp. 69–86, 2021.
4. **Sungho Suh**, Joel Jang, Seungjae Won, Mayank Shekhar Jha, and Yong Oh Lee, “Supervised Health Stage Prediction Using Convolutional Neural Networks for Bearing Wear”, *Sensors*, 20(20), 5846, 2020.
5. **Sungho Suh**, Haebom Lee, Jun Jo, Paul Lukowicz, and Yong Oh Lee, “Generative oversampling method for imbalanced data on bearing fault detection and diagnosis”, *Applied Sciences*, 9(4), 746, 2019.
6. **Sungho Suh**, Minsik Lee, and Chong-Ho Choi, “Robust Albedo Estimation from a Facial Image with Cast Shadow under General Unknown Lighting”, *IEEE Transactions on Image Processing*, vol. 22, Issue 1, 2013.

Conference

1. Hymalai Bello, Bo Zhou, **Sungho Suh**, and Paul Lukowicz, “Move like the music: posture and gesture detection in loose garments using theremin”, *ACM International Symposium on Wearable Computers (ISWC)*, 2021.
2. **Sungho Suh**, Sojeong Cheon, Dong-Jin Chang, Deukhee Lee, and Yong Oh Lee, “Sequential Lung Nodule Synthesis using Attribute-guided Generative Adversarial Networks”, *In Proceeding of 2021 24th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2021 .
3. **Sungho Suh**, Paul Lukowicz, and Yong Oh Lee, “Fusion of global-local features for image quality inspection of shipping label”, *In Proceedings of 2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, 2021.
4. **Sungho Suh**, Haebom Lee, Yong Oh Lee, Paul Lukowicz, and Jongwoon Hwang, “Robust shipping label recognition and validation for logistics by using deep neural networks”, *In Proceedings of 2019 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2019.
5. **Sungho Suh**, and Moonjoo Kim, “Automatic calibration of the optical system in passive component inspection”, *In Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017)*, 2017.
6. **Sungho Suh**, Hansang Cho, and Donglok Kim, “Robust Registration Method of 3D Point Cloud Data”, *In Proceedings of the 5th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2016)*, 2016.
7. **Sungho Suh**, and Hansang Cho, “Registration of point cloud data for HDD stamped base inspection”, *Applications of Digital Image Processing XXXVIII*, Vol. 9599. International Society for Optics and Photonics, 2015.
8. Hunsue Lee, **Sungho Suh**, and Hansang Cho, ”Robust template matching using run-length encoding”, *Applications of Digital Image Processing XXXVIII*, Vol. 9599. International Society for Optics and Photonics, 2015.
9. **Sungho Suh**, and Hansang Cho, ”Fast Image Alignment for Inspection of FCB Bump Coin System”, *In Proceeding of the 3rd International Conference on Industrial Application Engineering*, 2015.
10. Hyohoon Choi, Ram Mohan Gupta, and **Sungho Suh**, ”Quality measurement of Template Models and Automatic Template Model Selection”, *In Proceeding of the 12th International Conference on Control, Automation and Systems (ICCAS)*, IEEE, 2012.
11. **Sungho Suh**, Minsik Lee, and Chong-Ho Choi, ”Robust Albedo Estimation from a Facial Image with Cast Shadow”, *In Proceeding of the 18th IEEE International Conference on Image Processing*, IEEE, 2011.

PATENTS

1. Yong Oh Lee, Young Jun Kim, Baeckkyoung Sung, Changseon Ryu, Jihoon Park, Jayoung Park, **Sungho Suh**, Wonseo Choi, Taegeol Lee, and Ikhwan Kwon, “Method and apparatus for non-invasive real-time monitoring behavior patterns of ecological species for ecotoxicology risk assessment”, KR 10-2020-0115178, 2020.
2. **Sungho Suh**, Haebom Lee, Yong Oh Lee, and Jongwoon Hwang, “Method and apparatus for recognizing the delivery address printed on mailings”, KR 10-2020-0057685, 2020.
3. **Sungho Suh**, Sanghoon Han, Moonjoo Kim, and Jemin Kang, “Visual inspection apparatus and method of optical calibration of the same”, KR 10-2016-0130015, 2018.
4. **Sungho Suh**, Hansang Cho, and Suhyun Joo, “Apparatus for testing board and testing method thereof”, KR 10-2015-0028302, 2016.