A Fault Detection Model with Imbalanced Data using Machine Learning and Deep Learning

2022-1nd Semester

Interdisciplinary Project

Student ID	20181353
Name	Kim Yejin
School	Department of Industrial Engineering
1 track	Industiral Engineering
2 track	Computer Science and Engineering
Advisor 1	Sunhoon Lim
Advisor 2	Ilwoo Lyu

I. Introduction

1) Topic and Purpose of Research

In the recent years, as the amount of data increases, there has been a growing interest in the use of machine learning and deep learning in manufacturing industry and it is called smart manufacturing.[1] Smart manufacturing refers to the use of advanced data analytics to improve system performance and decision-making. It is an emerging field that combines traditional manufacturing with ICT(Information and Communication Technology) to the planning, design, production, distribution, and sales of products.

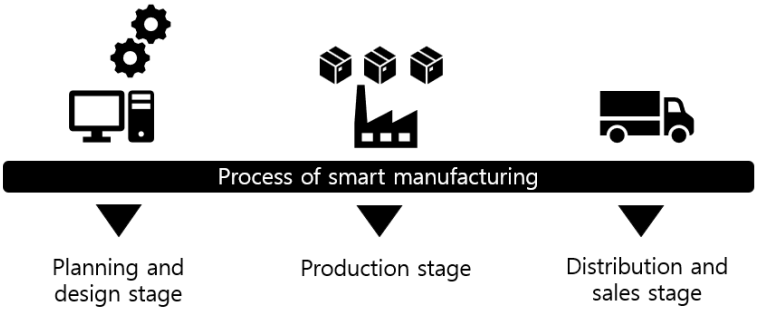

Figure 1 schematization of smart manufacturing

In the planning and design stage, it is possible to shorten the period and develop customized products by simulating before production in a virtual space. In the production stage, by exchanging information between facilities, materials, and systems in real time, mass production of multiple varieties and energy facility efficiency can be reconsidered. In the distribution and sales stages, real-time automatic orders can reduce inventory costs and allow cooperation in all fields from quality to logistics. Because of these advantages, many companies are seeking to change to smart factories. In particular, SMEs(Small and Medium Enterprise) are establishing a basic stage that can easily start with relatively low cost, and are satisfied with their performance beyond expectations. The aim of smart manufacturing is to create a more flexible and responsive manufacturing system that can rapidly adapt to changing market demands.

The main motivation for this new industrial revolutions is that these techniques offer the potential to automatically learn from data, and thereby improve the efficiency and effectiveness of manufacturing processes. Especially in the 21st century, data is more important than ever before. In the past, manufacturing was a process that was largely based on trial and error. This meant that a lot of time and resources were wasted in the production process. However, with the advent of smart manufacturing, this is no longer the case.

Then how to utilize the data more effectively? Machine learning and deep learning can be the powerful tools. It can learn from data and improve the efficiency of manufacturing processes. In recent years, there has been an explosion of interest in machine learning and deep learning, with these technologies being applied in a wide variety of fields such as computer vision, natural language processing, and robotics. Smart manufacturing is an area where these technologies can be used to great effect, and there are a number of different ways in which machine learning and deep learning can be used in this domain [2-4]. The benefits of smart manufacturing with machine learning and deep

learning are already being realized by many manufacturers. In the future, machine learning and deep learning in of smart manufacturing is likely to become increasingly important as the technology continues to develop and become more widely adopted.

One way in which machine learning and deep learning can be used in smart manufacturing is to detect faults in the production process and take corrective action in real-time. This can help to avoid costly production downtime and defects. Fault detection in manufacturing is typically done manually, which can be time-consuming and error-prone. Smart manufacturing systems can automate fault detection using sensors and data analytics. This can help to improve the accuracy and speed of fault detection, and enable corrective action to be taken more quickly.

However, It is often difficult and expensive to collect data on defective products. The reason is that normal product data and defective product data must be obtained through experiments, and there are overwhelmingly many normal product data. So the data is often unbalanced, with a large number of positive examples and a small number of negative examples.

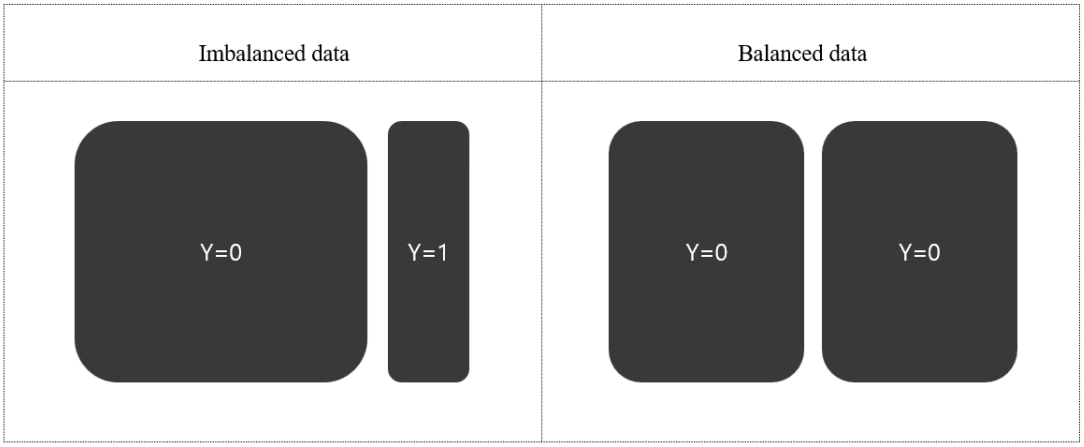

Figure 2 Diagram of imbalanced and balanced data

Unbalanced data and balanced data can be examined through Figure 2. Through this, it can be seen that unbalanced data means that data in which the target variable is 0 is much more than data in which the target variable is 1. It can also be seen that the balanced data have same data distribution regardless of the target variable.

The challenge of working with imbalanced data is that most machine learning algorithms are designed to work with data that is balanced, meaning that the classes are roughly equally represented. When the data is imbalanced, as is often the case in real-world applications, the minority class is at a disadvantage.[5] This can lead to poor performance on minority class examples, even if the overall accuracy is high. Therefore, several studies are being conducted to solve this problem.[6]

So in this research, we study how to apply machine learning and deep learning technique in real manufacturing data, especially for classification problem with imbalanced dataset. We propose some of the machine learning and deep neural network-based fault detection method in the field of smart manufactory. There are a number of ways to deal with imbalanced data, but we focus on oversample the minority class because of efficient. In addition, different types of method and model are employed in real-data and compared based on the way to estimate binary classification problem. According to this, we can find a way to solve the classification problem with the imbalanced data which is occurs a lot in real smart manufacturing situation.

II. Main Subject

1) Research Planning

A. Data Description

This section introduces a case study involving real-world data that have imbalanced problem. The dataset used in this research is collected by UV lamp company, Panasia. Panasia is a global eco-friendly facility company with air, water and hydrogen business solutions. Among them, the UV lamp that we will use is deployed to protect the marine ecosystem by sterilizing microorganisms in the equilibrium water of the ship. The process of processing UV lamps is very complicated. Figure 3 is a simple schema of this process.

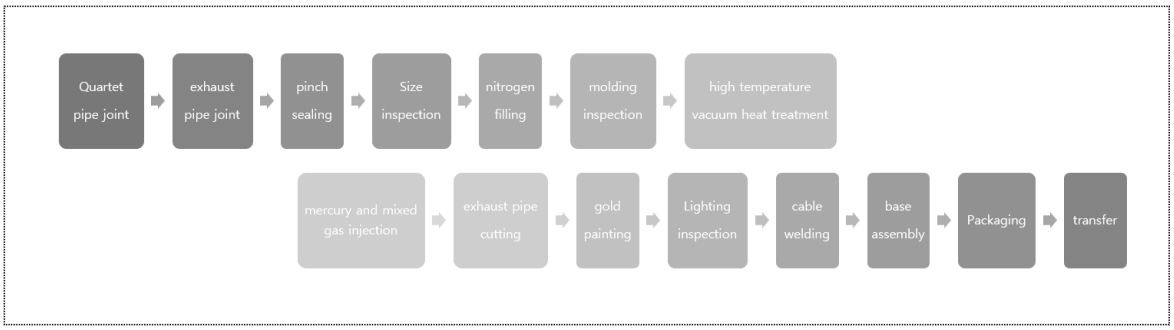

Figure 3 process of processing UV lamps

Among these processes, the data we will use are collected in pinch sealing, size inspection, high-temperature vacuum heat treatment, mercury and mixed gas injection, and lighting inspection. Pinch sealing is a process in which electrodes are inserted at both ends of the bonded quartz tube and heated, compressed, and sealed. Dimensional inspection is a work of measuring the thickness of a pinch part that has been pinched and performing a molded size inspection. The high-temperature vacuum heat treatment is a process of performing an annealing process to relieve stress on the heated and compressed quartz tube. The mercury and mixed gas injection process is a process of injecting mercury and mixed gas into a quartz tube in a vacuum state using an exhaust pipe for UV lamp emission. The lighting test is a process of determining whether it is defective after examining the UV lamp. Most of these processes are worked automatically, so that data is collected automatically. The data generated in the above process are described in Table 1.

Table 1 Description of variables

Variable	
Positive/negative	Pinch thickness-upper
Space between front and back	Pinch thickness-lower
Space between left and right	Electrode gap
Time between front and back	Front deviation-upper
Time between left and right	Front deviation-lower
Amount of hydrogen	Front deviation-gap
Amount of oxygen	Side deviation-upper
Burner revolution	Side deviation-lower
Burner speed	Side deviation-gap

B. Problem Setting

Our research purpose is to develop a model that predicts defects in products in advance. Therefore, the problem we have to solve is the binary classification problem that determines whether it is defective or not.

However, there were many problems with the data. First of all, the data we used was obtained from small and medium-sized manufacturing companies, so the number of data is small because of the inconvenient to collect data and lack of inspection personnel. Therefore, the number of data we used to develop the model is only 1607. In addition, there is not much data that is defective, so it is unbalanced data. As can be seen from the figure 4, the number of defective data is significantly smaller than that of good quality data.

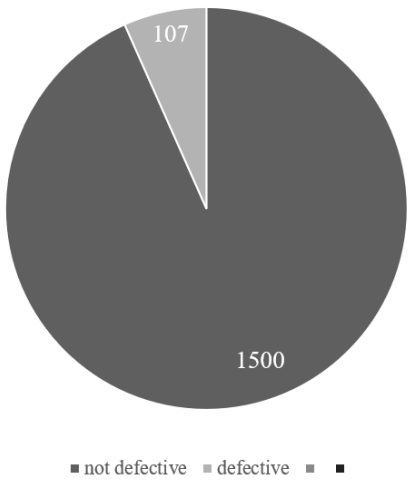

Figure 4 the number of data

2) Method

In this work, various method to oversample the minority class to detect defected product is proposed. Considering the imbalanced data, several machine learning and deep learning methods are selected and combined to enhance the performance of the fault detection model. At first, Random Forest which is basic machine learning technique is described and then deep learning model is also described. Next, we combined deep learning model and several method to oversample the minority class which is called Synthetic Minority Over-sampling Technique (SMOTE) and Adaptive Synthetic Sampling Approach(ADASYN) is explained. we explain how to solve the problem of imbalanced data by giving weight to the loss function when develop the deep learning model. Finally, we will also describe a method of oversampling using the state-of-the-art deep learning technique, Generative Adversarial Network.

A. Random Forest

Random forest is a machine learning algorithm that belongs to the ensemble learning family. Ensemble learning is a type of learning where you combine multiple models to get better results. Random Forest is an example of a bagging algorithm. Bagging is a type of ensemble learning where you train each model in the ensemble independently and then combine the results. Random Forest is a powerful machine learning algorithm that is very versatile and can be used for a variety of tasks such as regression, classification, and feature selection. It is a relatively simple algorithm to understand and implement and has been shown to outperform more complex algorithms such as support vector machines.

The key to the success of Random Forest is the fact that it is a very robust algorithm that is not susceptible to overfitting. Overfitting is a problem that occurs when a machine learning algorithm is too closely fit to the training data and does not generalize well to new data. This can be a problem with more complex algorithms that have a lot of parameters that can be tuned. Random Forest is less likely to overfit because it creates multiple decision trees and then combines them to form a single model. This means that the model is not as dependent on any single decision tree and can better handle new data. Another advantage of Random Forest is that it can handle both numerical and categorical data. This is important because many machine learning algorithms can only handle one type of data. This can be a problem when you have a dataset that contains both types of data. Random Forest can also handle missing data without having to impute the missing values.

One potential disadvantage of Random Forest is that it can be a bit slow to train, especially if you have a large dataset. However, once the model is trained, it can be very fast to make predictions. Another potential disadvantage is that Random Forest is not as easy to interpret as some other machine learning algorithms. This can be a problem if you need to explain your results to a non-technical audience.

Overall, Random Forest is a powerful and versatile machine learning algorithm that is well suited for a variety of tasks. It is robust and can handle both numerical and categorical data. It is also relatively easy to understand and implement.

B. Deep Neural Network

Deep neural networks (DNNs) are a powerful class of machine learning models that have achieved state-of-the-art performance on a variety of tasks, such as image classification, object detection, and machine translation.

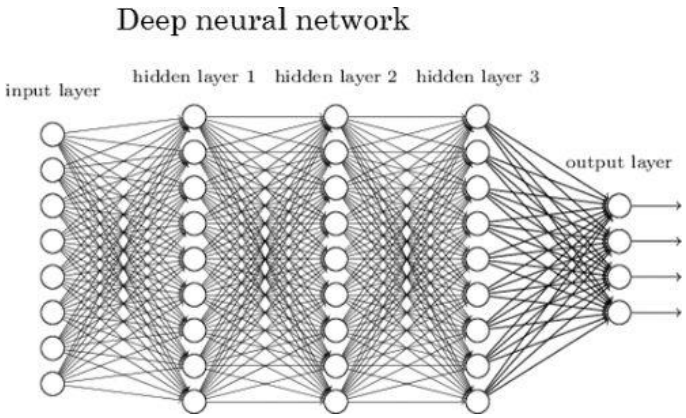

Figure 5 The general model of DNN

As you can see in Figure 5, DNNs are composed of multiple layers of artificial neurons, or "nodes", each of which performs a simple mathematical operation on its input. The output of each node is fed as input into the next layer, until the final layer produces the output of the DNN. The nodes in the hidden layers of a DNN can learn to recognize patterns of input data, which allows the DNN to perform complex tasks such as image classification and object detection. DNNs are trained using a variety of different algorithms, including backpropagation, which adjusts the weights of the nodes in the network in order to minimize the error of the DNN's predictions. DNNs are a powerful tool for machine learning, but they are also complex and can cause overfitting.

DNN can be used for binary classification. To train a deep neural network for binary classification, we first need to define the network architecture. The most common architecture for deep neural networks is the feed-forward neural network. In a feed-forward neural network, the information flows through the network in one direction, from the input layer to the output layer. The input layer of a deep neural network for binary classification contains the number of neurons as same as the number of features. The output layer contains a single neuron that predicts the class of the input. In between the input and output layer, there are a number of hidden layers. The number of hidden layers is a hyperparameter that we can tune.

Once we have defined the network architecture, we need to train the network. To do this, we need to specify the loss function that we want to minimize. The most common loss function for binary classification is the cross-entropy loss. The cross-entropy loss is defined as this.

$$L = -(ylog(p) + (1-y)\log(1-p)$$

Where p is the predicted probability of the input belonging to the positive class and y is the true label. The cross-entropy loss is a convex function and so it can be minimized using gradient descent. In gradient descent, we compute the gradient of the loss function with respect to the weights of the network. We then update the weights in the direction that decreases the loss. We repeat this process until the loss converges to a minimum.

Once the network has been trained, we can use it to make predictions on new data. To do this, we simply need to forward propagate the input through the network to obtain the predicted probability. We can then threshold the predicted probability to obtain the predicted class. Deep neural networks for binary classification are powerful models that can learn complex decision boundaries. They are easy to train and can be used to make predictions on new data.

C. Synthetic Minority Over-sampling Technique

The Synthetic Minority Over-sampling Technique (SMOTE) is a data pre-processing method used to address the issue of class imbalance in machine learning [7]. SMOTE works by creating synthetic samples of the minority class. This is done by selecting a point from the minority class and finding its k nearest neighbors. The synthetic points are then created by taking the vector between the selected point and one of its neighbors, and multiplying it by a random number between 0 and 1. This new point is then added to the minority class. This process is repeated until the desired number of synthetic samples has been created. The Pseudo-code of SMOTE is described in Figure 6.

Algorithm *SMOTE*(T, N, k)
Input: Number of minority class samples T; Amount of SMOTE $N\%$; Number of nearest neighbors k
Output: $(N/100)$ * T synthetic minority class samples
1. (∗ *If N is less than 100%, randomize the minority class samples as only a random percent of them will be SMOTEd.* ∗)
2. **if** $N < 100$
3. **then** Randomize the T minority class samples
4. $T = (N/100) * T$
5. $N = 100$
6. **endif**
7. $N = (int)(N/100)$ (∗ *The amount of SMOTE is assumed to be in integral multiples of 100.* ∗)
8. k = Number of nearest neighbors
9. $numattrs$ = Number of attributes
10. $Sample[\,][\,]$: array for original minority class samples
11. $newindex$: keeps a count of number of synthetic samples generated, initialized to 0
12. $Synthetic[\,][\,]$: array for synthetic samples
(∗ *Compute k nearest neighbors for each minority class sample only.* ∗)
13. **for** $i \leftarrow 1$ **to** T
14. Compute k nearest neighbors for i, and save the indices in the $nnarray$
15. Populate(N, i, $nnarray$)
16. **endfor**

Populate(N, i, $nnarray$) (∗ *Function to generate the synthetic samples.* ∗)
17. **while** $N \neq 0$
18. Choose a random number between 1 and k, call it nn. This step chooses one of the k nearest neighbors of i.
19. **for** $attr \leftarrow 1$ **to** $numattrs$
20. Compute: $dif = Sample[nnarray[nn]][attr] - Sample[i][attr]$
21. Compute: gap = random number between 0 and 1
22. $Synthetic[newindex][attr] = Sample[i][attr] + gap * dif$
23. **endfor**
24. $newindex{+}{+}$
25. $N = N - 1$
26. **endwhile**
27. **return** (∗ *End of Populate.* ∗)

Figure 6 Pseudo-code of SMOTE

SMOTE has been shown to be effective at reducing the class imbalance and improving the performance of machine learning models. It is important to note that SMOTE should only be used on the training data, and not on the test data. This is because SMOTE creates synthetic samples, which are not present in the real world and therefore would not be seen in the test data.

SMOTE is a powerful tool that can be used to improve the performance of machine learning models. However, it is important to use it carefully, as it can sometimes lead to overfitting.

D. Adaptive Synthetic Sampling Approach

Adaptive Synthetic Sampling Approach(ADASYN) is an advanced version of SMOTE [8]. ADASYN creates synthetic samples by interpolation between a minority class point and its nearest neighbors that have a higher density of minority class points. This means that ADASYN is more likely to create synthetic samples that are closer to the decision boundary than SMOTE. Also, ADASYN uses a variable number of nearest neighbors. The number of nearest neighbors used by ADASYN is inversely proportional to the density of minority class points. The more minority class points there are, the fewer nearest neighbors are used. The biggest difference between SMOTE and ADASYN is that SMOTE can only be used with binary classification, while ADASYN can be used with both binary and multi-class classification. Therefore, ADASYN is more improved version of SMOTE.

E. Deep Neural Network with Weighted Loss Functions

The weighted loss function is a deep learning technique that allows for the training of a model with a higher accuracy than would be possible with a standard loss function [9]. The weighted loss function is a modification of the standard loss function that assigns a higher weight to samples that are more difficult to classify. This results in the model being more accurate on these difficult samples, and thus overall. The weighted loss function has been shown to be particularly effective in deep learning models that are trained on imbalanced datasets. An imbalanced dataset is one where the classes are not evenly distributed, and so some classes are more represented than others. This can be a problem with standard loss functions, as the model can learn to simply predict the majority class, and so not be accurate on the minority class. The weighted loss function helps to alleviate this issue by giving more weight to the minority class, and so the model is forced to learn to better classify them.

There are a few different ways to weight the loss function, and the choice of method will depend on the dataset and the model being used. One common method is to use the inverse class frequencies, so that the minority class is given more weight. Another is to use a cost-sensitive method, where a cost is assigned to each class and the weights are chosen so that the model minimizes the cost. The weighted loss function is a powerful tool that can help to improve the accuracy of deep learning models. When training on imbalanced datasets, it is often the best choice of loss function to use. It is also a good choice when the model is expected to be used on data that is not evenly distributed.

F. Generative Adversarial Networks

Generative adversarial networks (GANs) are a type of artificial intelligence algorithm used to generate new data samples from a training dataset [6]. GANs are composed of two neural networks, a generator and a discriminator, that compete with each other in a zero-sum game. The generator network generates new data samples, while the discriminator network tries to classify them as either real or fake.

The training process for a GAN is an iterative process, where the generator and discriminator networks are trained alternately. The generator network is first trained to generate new data samples that are similar to the real data in the training dataset. The discriminator network is then trained to classify these generated data samples as fake. The generator network is then trained again to generate data samples that are even more similar to the real data, and so on.

Eventually, the generator network will learn to generate data samples that are indistinguishable from the real data, and the discriminator network will learn to always classify these generated data samples as real. At this point, the GAN has converged and the training process is complete.

GANs have been used to generate realistic images, videos, and text. They have also been used to generate new data samples from a limited number of training examples, which is known as few-shot learning. GANs have also been used to improve the performance of other machine learning models, such as image classifiers.

3) Model Evaluation Method

In this chapter, we will describe the evaluation method that used to evaluate the performance of the model. Because we use imbalanced data, high accuracy does not mean a good model. This is because the model may perform well on a large number of data, but poorly on a small number of data. Therefore, we evaluated the model from various aspects using various evaluation method.

A. Confusion matrix

A confusion matrix is a table that is used to evaluate the performance of a binary classification model. You can see the table in Figure 7.

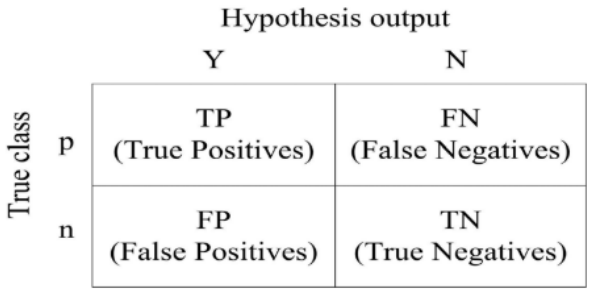

Figure 7 Confusion matrix

The table is made up of four rows and four columns, where the rows represent the actual values and the columns represent the predicted values. The first row represents the true positives, the second row represents the false positives, the third row represents the false negatives, and the fourth row represents the true negatives.

The confusion matrix can be used to calculate a number of different metrics, such as accuracy, precision, recall, and specificity. In next, we will explain how to calculate each of it.

B. Accuracy, precision, recall, F1 score

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$Precision = \frac{T_p}{T_p + F_p}$$

$$Recall = \frac{T_p}{T_p + T_n}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Figure 8 The formula of Accuracy, Precision, Recall and F1 score

Accuracy is simply the number of correct predictions divided by the total number of predictions. While accuracy is a useful metric, it can be misleading in some cases. For example, imagine we have a classifier that always predicts the positive class. In this case, the classifier would have an accuracy of 100%, even though it is clearly not making accurate predictions.

The precision of a confusion matrix is the number of correctly classified examples divided by the total number of examples classified. This metric is important because it gives you a sense of how well the classifier is able to correctly identify examples of each class. A high precision means that when the classifier predicts an example is of a certain class, it is very likely to be correct. A low precision means that when the classifier predicts an example is of a certain class, it is very likely to be incorrect.

The recall of the confusion matrix is the ratio of the number of true positives to the number of false negatives. The recall can be thought of as the ability of the model to correctly identify positive examples. The recall is calculated by dividing the number of true positives by the sum of the true positives and the false negatives. The recall is an important metric for evaluating the performance of a machine learning model, as it gives us a measure of how well the model is able to identify positive examples. A high recall indicates that the model is good at identifying positive examples, while a low recall indicates that the model is not good at identifying positive examples.

The F1 score is a measure of a classifier's accuracy. It is the harmonic mean of the precision and recall, where precision is the number of true positives divided by the sum of the true positives and false positives, and recall is the number of true positives divided by the sum of the true positives and false negatives. The F1 score is between 0 and 1, with 1 being the best possible score.

C. ROC curve

A ROC curve is a graphical representation of the performance of a binary classification model at various thresholds.

Figure 9 ROC Curve

The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at different thresholds. The true positive rate is the proportion of positive examples that are correctly classified as positive, while the false positive rate is the proportion of negative examples that are incorrectly classified as positive. The ROC curve is a useful tool for comparing the performance of different binary classification models, as it allows for easy visualization of the trade-off between the true positive rate and the false positive rate. The higher the true positive rate, the more positive examples are correctly classified, but the trade-off is that the false positive rate will also increase. The ideal point on the ROC curve is the top-left corner, where the true positive rate is 1 and the false positive rate is 0.

The area under the ROC curve (AUC) is a measure of the overall performance of the model, and can be used to compare different models. A model with a higher AUC will have a better performance than a model with a lower AUC.

The ROC curve is a valuable tool for evaluating the performance of binary classification models. It is easy to interpret and can be used to compare the performance of different models. The AUC is a useful measure of the overall performance of the model, and can be used to compare different models.

III. Conclusion and Discussion

1) Research Results

The experiment is conducted by Google Colab environment with python. Random Forest use scikit-earn, Deep Neural Network(DNN) and Generative Adversarial Network(GAN) use TensorFlow, Synthetic Minority Over-sampling Technique(SMOTE) and Adaptive Synthetic Sampling Approach(ADASYN) use python code to implement to data. Thy hyperparameters for each algorithms are tuned for train set, and the details of the model specifications are shown in Tables.

Table 2 Detailed hyperparameters of the used machine learning models and some of methods

Model	Detailed hyperparameter values
Random Forest	Criterion = entropy, max depth = 13, # estimators = 17
SMOTE	The amount of SMOTE = 1401, The number of nearest neighbors = 5
ADASYN	Balance level = 1, The number of nearest neighbors = 8
Weight loss function	Weight for class 0 = 0.54, Weight for class 1 = 7.51

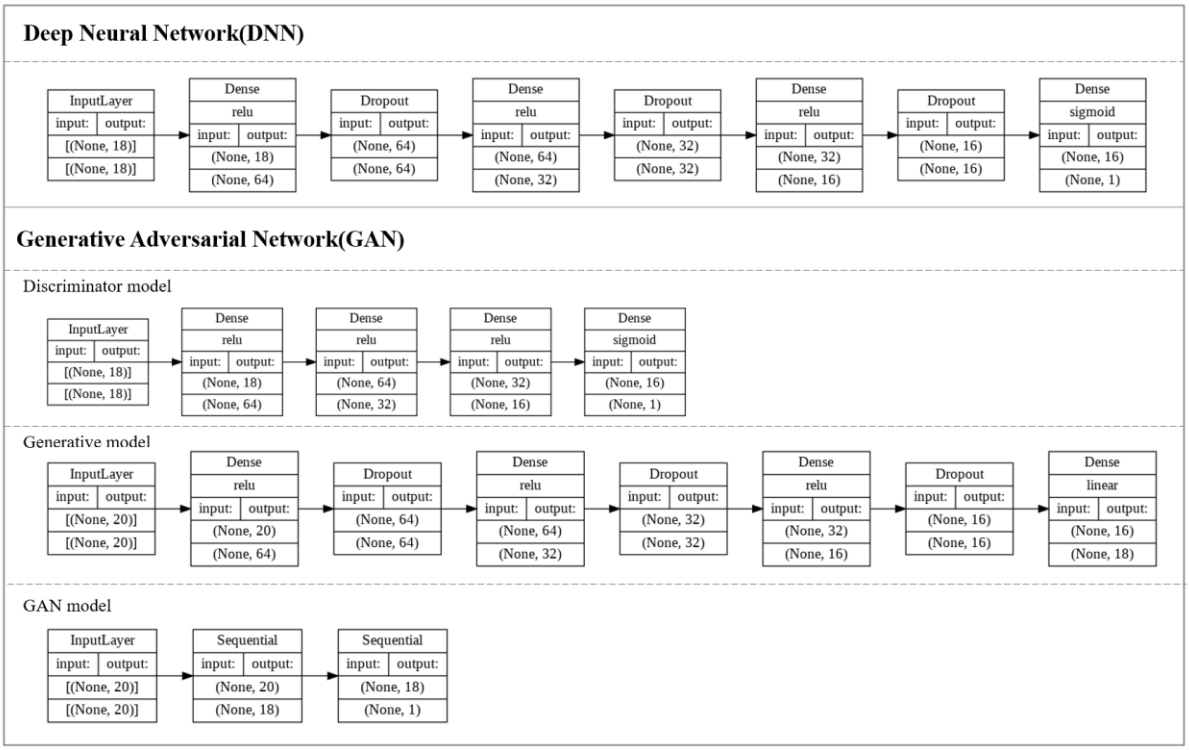

Figure 10 Detailed architectures of the used deep learning models

The following is a comparison of each methods, including machine learning methods, method to increase minority class and deep learning. Table 3 show the results of fault detection measured with matric, accuracy, precision, recall and F1 score. According to Table 3, Random Forest and Deep Neural Network show highest accuracy among other methods. the Adaptive Synthetic Sampling approach with Deep Neural Network shows the highest F1 score among other methods. However, the two models have very low ability to detect defective products. So it shows a low recall. On the other hand, Weighted loss function show highest recall among other methods. But there was a trade-off in this. So it score lowest accuracy and precision. The model with the highest f1 score considering both of these trade-off is the Synthetic Minority Over-sampling Technique(SMOTE) + DNN, Adaptive Synthetic Sampling Approach(ADASYN) + DNN and Generative Adversarial Network(GAN) + DNN. Through this, we can find that the method of oversampling a minority class shows the better performance than other.

Table 3 Performance comparison of methods

Model	TP	FN	FP	TN	Accuracy	Precision	Recall	F1 score
Random Forest	3	18	1	300	0.94	0.75	0.14	0.24
Deep Neural Network(DNN)	4	17	1	300	0.94	0.80	0.19	0.31
Synthetic Minority Over-sampling Technique + DNN	10	11	20	281	0.90	0.33	0.47	0.39
Adaptive Synthetic Sampling Approach + DNN	10	11	19	282	0.90	0.34	0.48	0.40
Weighted loss function	13	8	44	257	0.83	0.23	0.62	0.34
Generative Adversarial Network(GAN) + DNN	7	14	9	292	0.92	0.44	0.33	0.38

\

Figure 11 show ROC curve of each model, and Table 4 show AUC score of each model. According to the figure 11, we can see that the more the ROC curve of the test dataset bends to above, the less the ROC curve of the train dataset bends. Through this, it can be seen that the overfitting model does not have good actual performance.

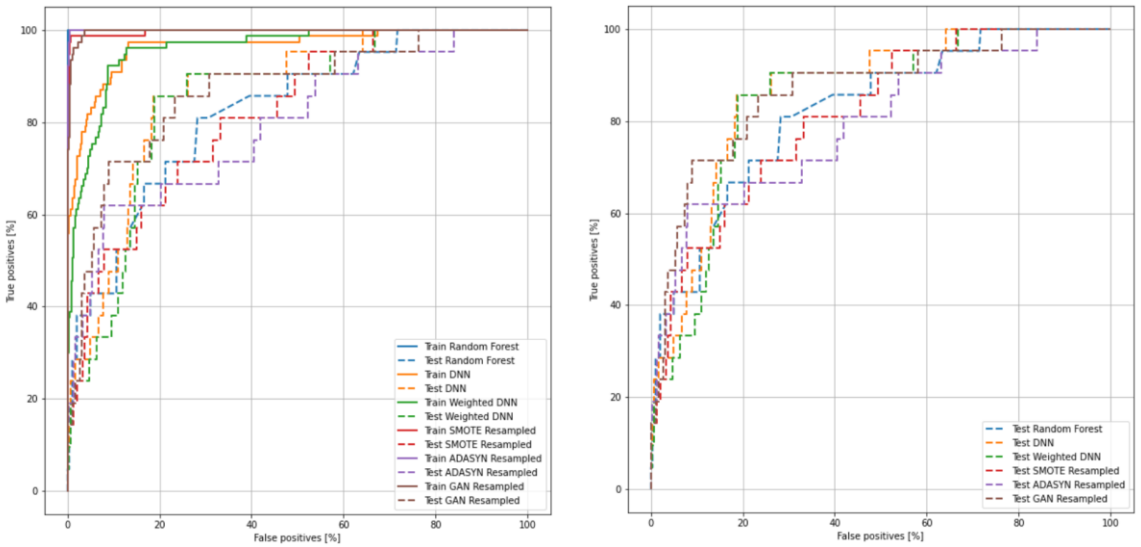

Figure 11 ROC curve of each model

According to Table 4, Generative Adversarial Network(GAN) + DNN model shows the highest score among other methods. So, the multi-model deep learning-based fault detection model have highest capable of distinguishing between classes.

Table 4 AUC score of each model

Model	AUC
Random Forest	0.822
Deep Neural Network(DNN)	0.862
Synthetic Minority Over-sampling Technique + DNN	0.815
Adaptive Synthetic Sampling Approach + DNN	0.795
Weighted loss function	0.846
Generative Adversarial Network(GAN) + DNN	0.866

2) Discussion

In this research, we propose a fault detection model with imbalanced data using Machine Learning and Deep Learning. Detecting fault in advance is the most important problem in manufacturing. Because it is cost-effective and time-consuming. In previous, Fault detection is typically done manually. But, with the growing interest in Smart manufacturing, systems can detect fault product automatically using sensors and data analytics. Therefore, research in this area is essential.

However, It is often difficult and expensive to collect data on defective products. So the data is often unbalanced, with a large number of positive examples and a small number of negative examples. The challenge of working with imbalanced data is that most machine learning algorithms are designed to work with data that is balanced. So in this research, we study how to apply machine learning and deep learning technique in real manufacturing data, especially for classification problem with imbalanced dataset.

The proposed model are six, Random forest, Deep Neural Network(DNN), Synthetic Minority Over-sampling Technique(SMOTE), Adaptive Synthetic Sampling Approach(ADASYN), Deep Neural Network with Weighted Loss Functions and Generative Adversarial Networks(GAN). Each model is evaluated by confusion matrix, Accuracy, Precision, Recall, F1 score and ROC curve.

Experimental results illustrate the superiority of oversampling a minority class according to F1 score. Furthermore, multimodal setting, GAN+DNN, have shown best performance in AUC score among other methods. In particular, due to the small number of data, the model, which could not avoid overfitting, showed poor performance.

The proposed approach range from basic machine learning technique to state-of-the-art deep neural networks in the field of smart manufacturing. Future work will be conducted about the model that can solve both the imbalance and the overfitting problem at the same time. It will also be necessary to study ways to reflect more various forms of data.

※ References

1. Zheng, P., et al., Smart manufacturing systems for Industry 4.0: Conceptual framework, scenarios, and future perspectives. Frontiers of Mechanical Engineering, 2018. **13**(2): p. 137-150.
2. Wang, J., et al., Deep learning for smart manufacturing: Methods and applications. Journal of Manufacturing Systems, 2018. **48**: p. 144-156.
3. Kotsiopoulos, T., et al., Machine Learning and Deep Learning in smart manufacturing: The Smart Grid paradigm. Computer Science Review, 2021. **40**.
4. Çınar, Z.M., et al., Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0. Sustainability, 2020. **12**(19).
5. Wang, L., et al., Review of Classification Methods on Unbalanced Data Sets. IEEE Access, 2021. **9**: p. 64606-64628.
6. Zhang, W., et al., Machinery fault diagnosis with imbalanced data using deep generative adversarial networks. Measurement, 2020. **152**.
7. SMOTE_Synthetic Minority Over-sampling Technique.
8. ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced.
9. DNN AND CNN WITH WEIGHTED AND MULTI-TASK LOSS FUNCTIONS FOR AUDIO.