Symbol Detection in a Multi-class Dataset Based on Single Line Diagrams using Deep Learning Models

Hina Bhanbhro[1], Yew Kwang Hooi[2], Worapan Kusakunniran[3], Zaira Hassan Amur[4]
Computer and Information Science Department, Universiti Teknologi,
PETRONAS Seri Iskandar, Perak Darul Ridzuan, Malaysia[1, 2, 4]
Faculty of Information and Communication Technology, Mahidol University, Thailand[3]

***Abstract*—Single Line Diagrams (SLDs) are used in electrical power distribution systems. These diagrams are crucial to engineers during the installation, maintenance, and inspection phases. For the digital interpretation of these documents, deep learning-based object detection methods can be utilized. However, there is a lack of efforts made to digitize the SLDs using deep learning methods, which is due to the class-imbalance problem of these technical drawings. In this paper, a method to address this challenge is proposed. First, we use the latest variant of You Look Only Once (YOLO), YOLO v8 to localize and detect the symbols present in the single-line diagrams. Our experiments determine that the accuracy of symbol detection based on YOLO v8 is almost 95%, which is more satisfactory than its previous versions. Secondly, we use a synthetic dataset generated using multi-fake class generative adversarial network (MFCGAN) and create fake classes to cope with the class imbalance problem. The images generated using the GAN are then combined with the original images to create an augmented dataset, and YOLO v5 is used for the classification of the augmented dataset. The experiments reveal that the GAN model had the capability to learn properly from a small number of complex diagrams. The detection results show that the accuracy of YOLO v5 is more than 96.3%, which is higher than the YOLO v8 accuracy. After analyzing the experiment results, we might deduce that creating multiple fake classes improved the classification of engineering symbols in SLDs.**

Keywords*—Single line diagrams; engineering drawings; synthetic data; symbol detection; deep learning; augmented dataset

I. INTRODUCTION

An engineering drawing (ED) is an illustration of a schematic that demonstrates the operation or construction of an electrical system, procedure, or plant facility [1]. Engineering designs comprise of technical drawings such as mechanical or architectural blueprints, electrical circuits, and drawings [2]. In many different businesses, there is an increasing need for establishing digital systems for processing and analyzing these representations [3]. With such a framework, connected businesses will have the unusual opportunity to make extensive use of diagrams to direct their future practices.

A single-line diagram uses lines and symbols to represent the logical flow of power through physical processes and plant components. Although these components resemble each other in form and shape, they are highly asymmetrical in nature, which makes these documents complex [1]. Distinct power distributions are represented by lines of variable thickness, and each sign stands for a different component such as a transformer, generator, motor, switch, etc. [4]. A typical SLD diagram may have over 50 different symbols, making it an information-rich visual representation. While placing a purchase order or even when project teams are scheduling their work, these drawings are carefully inspected in order to estimate the numbers of various pieces of equipment [5]. When symbols on SLD diagrams are functionally different but visually identical, as in Fig. 1, this process can become considerably more difficult and complex. As a result, distinguishing one symbol from another can be both crucial and difficult. Misreading or omitting any material can also cause severe internal disagreements and be damaging to the progress of a project.

Scientists, on the other hand, are looking into solutions for a power system to transform the conventional power system that existed before into an intelligent power system. The fusion of a power system with artificial intelligence is getting closer and closer as new technologies, like artificial intelligence, arise [6]. It is a common duty in modern businesses and academia to include artificial intelligence technology in power system dispatching software to speed up the process of creating circuit diagrams for power systems. A fundamental document in the power system, the principal wiring diagram of the power station is also commonly needed for viewing and change by the power system's dispatching users [7]. The current power dispatching system relies heavily on the work expertise of dispatchers for the creation and upkeep of station wiring diagrams, which not only raises the danger of safety mishaps in the power grid system but also drives up the cost of wiring diagram maintenance [8, 12]. Therefore, one difficulty facing the contemporary power sector was how to employ artificial intelligence technology to automatically build the station wiring diagram.

Generative models have also undergone significant progress and have been successfully used in numerous areas. One of those is the Generative Adversarial Networks (GAN), which has emerged as a well-known and frequently employed technique for producing content. Ian Goodfellow first introduced GANs in 2014 [9]. We will go over our GAN-based approach to solving the issue of imbalanced classes within the context of Methods section. Another difficult issue that affects a wide range of fields, including engineering drawings [10], is the under-or over-representation of one or more classes of symbols in the diagrams in the dataset [11].

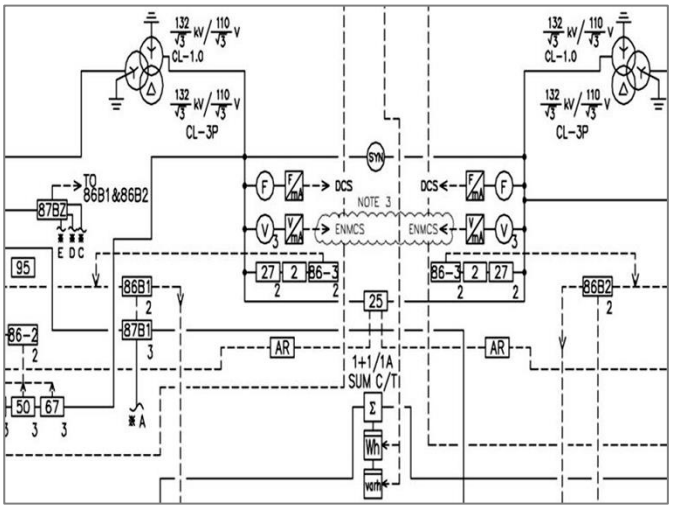


Fig. 1. A part of single line diagram.

It is logical to presume that construction industries have these designs for their on-going projects in a readable electronic format that can be edited with cutting-edge software. However, many businesses continue to maintain these designs as paper copies or in digitized form, particularly for their older projects. Therefore, digitizing these drawings in a way that makes information extraction simple and accessible, may be advantageous [13]. This can make it simple to correct previous designs when the plant's components have been replaced over time owing to maintenance. As a result, project teams will find it simpler to keep track of their instrumentation inventory during the building phase and to create a library of up-to-date drawings for maintenance during the post-installation phase with the help of digitized and updated SLD diagrams. The following restrictions are communicated through the contextualization and digitization of complicated SLDs:

A. Size

According to an estimate in [11], a typical SLD page consists of approximately 50 distinct types of shapes including symbols, connectors, and text. To depict a specific segment of a power system, it may be necessary to utilize anywhere between 100 to 500 pages.

B. Asymmetrical Components

Apart from the typical challenges of classical machine vision such as variations in lighting, scale, and pose, technical drawings utilize equipment symbols that conform to different standards across various industries. Consequently, assembling a precisely labeled dataset that can be employed for symbol classification is a complex undertaking, as mentioned in reference [14]. It is crucial to have a comprehensive assortment of precisely defined symbols that lack symmetry to effectively employ advanced deep learning methods for symbol recognition.

C. Connecting Lines

Connecting lines that indicate the logical and physical relation-ships between symbols are abundant and knotted in complex SLDs. As a result, it is difficult to apply digitization techniques based on thinning [15] or vectorizing [15]. The art-work for line identification are represented by lines of various styles and thicknesses. Furthermore, sophisticated Engineering Drawings (EDs) adhere to rule sets for application-based connectivity. This means that based on a standard that cannot be stated or inferred from the use of the physical lines connecting the symbols, two symbols may or may not be connected. As a result, contextualization is more difficult to implement than when it is applied to simpler drawings, like circuit diagrams [16]. This opens up several intriguing options, such as incorporating human expert knowledge through human-machine inter-action into a potential solution. Another avenue would be interactive learning [17].

D. Labels

Symbols, connectors, and other text characters may overlap; however, symbols and annotations in a variety of scripts and styles are used to distinguish between symbols exhibiting comparable characteristics, to indicate connectors, and to clarify additional information. Symbols with an overlap in drawing sheets are difficult to separate, as demonstrated by techniques like those used by Cao et al. [18] and Roy et al. [19]. Three further challenges have been identified once all of the text elements have been found: As seen in Fig. 1, various lengths and sizes are used to represent text strings that describe symbols and connectors. Additionally, it can be challenging to connect symbols and connectors to their matching text, and text interpretation mistakes could lead to some information being misunderstood.

E. Samples with Inconsistent Occurrence

Inconsistent appearance of symbols within the diagrams is another major issue towards digitization. Deep learning models perform better with large amounts of samples, while in SLDs, symbol frequency is highly imbalanced which creates a class imbalance problem due to the dominance of the majority classes over the minority classes. Hence, deep learning models can be biased towards the majority classes.

A range of methods, especially from the field of machine vision, must be applied to overcome these obstacles. These include symbol detection and localization, as well as feature extraction. The fact that recent advancements in deep learning and machine vision, particularly in the recognition and classification of objects, have not been put to the test against such challenging real-world situations, must be noted [21].

In this article, particularly, the YOLOv8 model for object identification and MFCGAN for class balancing are thoroughly examined. To extract symbols from drawing images, this study aims to use SLDs to create a dataset for model training. The dataset contains 22 different classes of symbols various shapes and sizes.

We were unable to locate a study that assesses a significant amount of deep learning-based detection algorithms that have been particularly designed for the problem do-main of single-line symbol identification while taking into account key variables including Precision, Recall, and F1.

The following are the main contributions of this study:

- Symbols in SLD images are classified using YOLOv8, the latest variant of YOLO model.

- Mixed-quality single-line symbols are synthetically generated using MFC-GAN.
- A GAN-based solution is provided for enhancing the quantity of minority classes to handle the class imbalance problem; along with an expansion of the YOLOv5 training set using newly generated synthetic data.
- We suggested an experimental setup using MFC-GAN for creating synthetic images.
- The accuracy of symbol identification and recognition in Single Line Diagrams (SLDs) is enhanced by using a YOLOv5-based network for object detection.

According to experiments, the *IoU* and performance of the model can be enhanced through the use of synthetic image data generated using different GANs.

The remainder of this paper is structured as follows: In Section II, we delve into the landscape of existing research within the relevant domain, examining both the challenges that have been encountered and the solutions that researchers have put forth in this realm. Moving forward to Section III, we intricately explore the proposed methodology and perform an in-depth analysis of the dataset. Moreover, within this section, we provide a comprehensive exposition of the detection model. The outcomes of our dataset construction and symbol detection are meticulously presented in Section IV. Subsequently, we engage in a thorough discussion of the results in Section V. Lastly, we draw this study to a conclusion in Section VI.

II. Related Work

This section covers recent accomplishments made by the research community in this domain. We discuss single-line engineering drawings, different deep learning techniques used for digitizing the engineering drawings, later we present GANs and discuss the general architecture and recent advancements made to improve the performance of GANs.

A. Single Line Diagrams

In various papers, including [1-4], the problem of recognizing and grouping symbols present in single-line diagrams (SLD) has been raised. The challenge of digitizing SLD, where the aim is to summarize the link between the numerous symbols, served as the inspiration for several of these works. The study in [22] provides an overview of numerous strategies created to digitize ED. In earlier research, including [23-26], symbols were recognized using classifiers that were traditionally based on machine learning and fed hand-crafted characteristics.

SLD digitization has notably drawn a lot of business interest due to the wide range of applications that may be made from a digital output, such as security evaluation, graphic simulations, or data analytics [27]. There are certain strategies developed expressly to handle the digitization of SLDs in the literature. More than 30 years ago, Furuta et al. [48] and Ishii et al. [28] published research on developing software to enable fully automated P&ID digitization. These techniques are currently ineffective due to incompatibility with hardware and software requirements. About ten years later, Howie et al. [29, 30] suggested a semi-automatic technique for localizing symbols of interest using the templates of the symbols as input. Gellaboina et al.'s [49] description of the most recent method for symbol identification uses an iterative learning strategy based on recurrent training of a neural network (NN) with the Hopfield model. This method was developed to pinpoint the most frequently occurring symbols in the artwork that also displayed a prototype pattern. Deep learning models were utilized [31] to build one-line diagrams automatically while generating core power systems.

B. Symbol Detection Using YOLO

Object detection can identify the sort of object present in an image or video and pin-point its location at the same time. In photos and videos, object detection expresses the location information as X and Y coordinate values. Additionally, the width and height values—which represent the object's size—are utilized as label information. Typically, the width and height data are expressed as bounding boxes using the X and Y coordinates.

Recent studies have employed deep neural networks to perform symbol spotting. For instance, researchers in [34] employed the YOLO 32, [33] model to identify symbols in floor plan diagrams. In another study [10], symbol detection was reformulated as a semantic segmentation problem, which led to the development of a pixel-level approach for symbol detection. Researchers are using YOLO for the goal of symbol recognition and classification as a result of the one-stage detection method's growing popularity and success [11]. To do this, the authors of [12] suggested transforming a construction image into a region adjacency network, where each node represented a connected component in the image. These nodes were then categorized using a YOLO. The YOLO and CNN-based technique was put forth in [13] and used to categorize symbols in [14].

Recent research has confirmed the effectiveness of YOLO variants in detecting complicated engineering components [35]. For instance, one-line symbols in substation diagrams were localized and categorized using YOLOv3. The model correctly identified 97% of the symbols. YOLO algorithms demonstrated encouraging results in detecting the symbols in electrical circuits despite the lack of suitable datasets [20]. Additionally, YOLO variations were used to accurately classify hand-drawn electric symbols with a 95% accuracy [11]. To the best of our knowledge, the work detailed here is the first attempt at localizing and matching symbols in a zero-shot method despite the very extensive literature that already exists in this field.

Since 2012, two primary types of deep learning-based object detection models have emerged: one-stage detectors and two-stage detectors, as described in research [32]. Understanding the concepts of region proposal and classification is essential to comprehend the distinction between the two categories. Region proposal refers to an algorithm that quickly identifies possible object locations, while classification is the process of categorizing objects based on their specific type. Although two-stage detectors are better at accurately detecting objects, their slow prediction time restricts their real-time detection ability. To address this issue,

one-stage detectors have been proposed that perform both classification and region proposal simultaneously, resulting in faster object detection. The one-stage detector is a technique that produces results by simultaneously executing classification and region proposal.

As depicted in Fig. 2, upon inputting the image to the model, the Convolutional Layer is employed to extract its features and perform classification. Simultaneously, a region proposal is conducted to generate the output. Models like YOLO, RetinaNet, RefineDet, etc. are good examples [37].

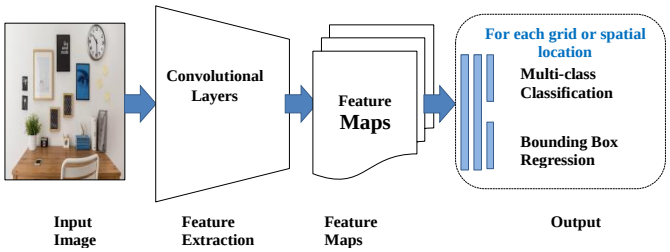


Fig. 2. One-stage model for object detection.

One of these one-stage detectors is YOLO, which integrates the region proposal and classification stages into a single operation. This means that it predicts the position and type of an object simultaneously by treating the bounding box and class probability as a single problem. YOLO divides the image into grids of a predetermined size to forecast the bounding box for each grid, and then trains the bounding-box confidence score and grid cell class score, as mentioned in reference [38].

The YOLO processing procedure is depicted in Fig. 2. First, an SxS grid area is created from the input image. The number of bounding boxes anticipated in each grid cell is equal to the number of bounding boxes that correspond to the area where an object is located. This can be denoted as (x, y, w, h), where (x, y) denotes the center point coordinates of the bounding box, and (w, h) denote its width and height.

Second, the confidence, which stands for the box's dependability and is determined similarly to Equation (1). The *IoU* (Intersection over Union) is used to determine it by computing the ratio of the overlapping area between the predicted and ground truth bounding boxes divided by the probability Pr(Object), which represents the likelihood of an object being present in the grid.

$$^{pred}\Pr(\text{Object}) \times IoU^{truth} \quad (1)$$

The probability of C classes is then determined for each grid and Equation (2) is shown.

$$^{pred}(\text{Class}_i \mid \text{Object}) \quad (2)$$

In this instance, what is strange is that YOLO does not classify the number of classes (background) as an input to a neural network model, although the existing Object detection does [38]. YOLO divides the input image into grids in this manner, performing classification and bounding box calculations for each grid at the same time.

C. Synthetic Data Generation Using GANs

Several studies in the past decade have explored the challenge of identifying symbols in architectural floor plans. To overcome the scarcity of training data available for neural networks, the authors recommended employing a Generative Adversarial Network (GAN) to generate synthetic training data.

Ian Goodfellow first introduced generative adversarial networks (GAN) in 2014. (Goodfellow et al., 2014). These are regarded as generative models that can produce original content. The Generator (G) and the Discriminator (D) are two competing models (such as CNNs, neural networks, etc.) that make up GANs [39]. The discriminator is a classifier that gets input from both the generator and the training set (genuine content). (Fake input). The discriminator will learn how to differentiate between real input samples and bogus input samples during the training phase. However, the generator is trained to provide samples that accurately reflect the fundamental properties of the original data. (Replicating original content). The GAN model is shown in Fig. 3.

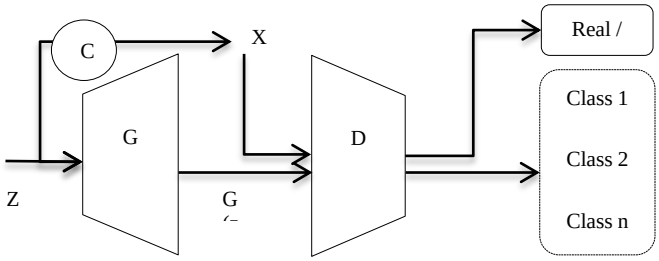


Fig. 3. Architecture of generative adversarial networks.

Equation (3) demonstrates that the value function is employed to perform adversarial training of both models G and D.

$$\min_D \max_G V(D, G) = E_{x\sim pdata(x)}[\log D(x)] + E_{z\sim pz}(z)[\log(1 - D(G(z)))] \quad (3)$$

Where x is a sample from the real training data, Pdata (x) is the probability distribution over the real data, the probability distribution over the noise vector z is referred to as Pz, and the outcome of the generator function G (or generated images) is denoted as G(z). GANs are at the forefront of image generation quality, as per [39].

GANs have been effectively used to solve a variety of issues, including speech synthesis, segmentation, and image production [40]. They have also been successfully used in recent years to address issues with class imbalance. The class mismatch is widespread throughout numerous industries, including banking, security, and health [6]. The issue arises when one or more classes are unequally or excessively represented in the dataset. When dealing with imbalanced datasets, a conventional supervised learning algorithm tends to favor the majority class [41].

By including conditional probabilities in the value function, supervised GANs offer an improvement over the basic GAN architecture. This gives the user more control over the samples that are created and introduces the diversity that is required to supplement synthetic input data for datasets with class

imbalance. Examples of this kind are AC-GAN [10], CGAN [9], and vanilla GAN [8]; even though the literature demonstrates that these models, particularly in extreme situations, can be significantly impacted by class disparity [11].

III. Research Methodologies

We present our method for recognizing the end-to-end symbols from intricate engineering drawings in Section III A. The dataset utilized for the tests will be covered in detail in the subsection that follows. Data exploration and pre-processing will be part of this. The specifics of our suggested approach to dealing with a class imbalance in these drawings are provided in Section IV.

Machine learning is commonly used to classify symbols and texts. Fig. 4 shows a conceptual model for digitizing engineering drawings that includes the essential phases. Such a framework will be extremely useful in fields where schematics can be turned into knowledge.

We determine the characteristics and variety of the created minority samples for our image-generating experiment after each run. In classification studies, we add created minority samples from trained models to the training data. (MFCGAN). The classification performances on the minority classes are then provided after a YOLO classifier has been trained on the expanded dataset.

A. Overview of Symbol Detection Framework

We first look for the areas of an engineering diagram that might contain interesting symbols and attempt to extract all the components from drawing. The next step is to locate and count the interesting symbols that originate from these zones of interest. The vast array of shapes and structures that these symbols emerge in drawings is the task's main problem. Furthermore, as stated in Section I, we cannot anticipate identical depictions of a specific component on all drawings. Additionally, there are a great number of different components and elements that are frequently used in these diagrams. As a result, it is not viable to use a fully supervised technique, training thoroughly to recognize and classifying every single type of object that could be seen in such images.

Information about the symbols that appear in an engineering drawing can be found in a variety of ways, including:

1) A table of legends listing the names of the components represented by the different symbols.

2) A table with numbers that represent the index of a component and the name of the object it represents.

3) There is no tabular data linking the names of objects to the appropriate diagrams.

In the current study, we focus on the first form of drawing, in which the component name and drawing image are both provided. We go into great detail on the various parts of the suggested framework in the sections that follow.

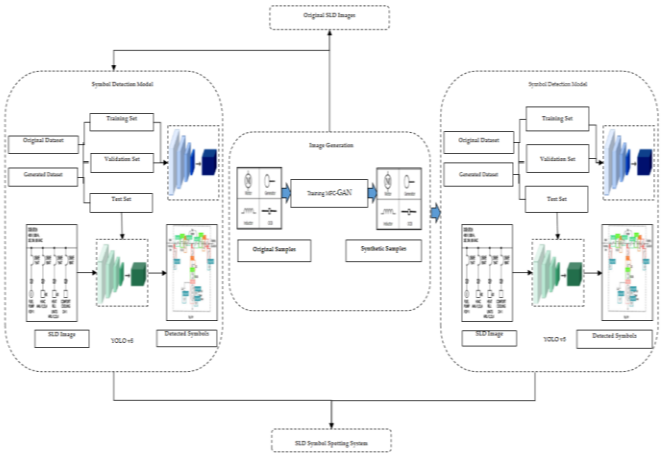


Fig. 4. Schematic of model for digitization of SLDs.

B. Summary of SLD Dataset

For the study in this paper, we chose to employ Single Line Diagrams (SLDs), as shown in Fig. 1. The engineering partner gave a set of 800 sheets for review. These diagrams contain a variety of symbols of varied sizes and dissimilar (asymmetrical) nature, as shown in Fig. 5.

The dataset is suitable for evaluation because the SLDs have a variety of attributes. The numerous electrical system components and connectivity information can be seen schematically represented on the SLD sheets. It is a representation of electrical apparatus and power flow movement, frequently in the form of symbols (represented as various kinds of lines).

In many industries, these diagrams can be found as paper documents or digital photographs. Evaluating and analyzing these materials requires a lot of experience, knowledge, and time [15]. Furthermore, misreading these publications can have disastrous repercussions. For instance, if an engineer needs to modify a wire in an electrical system after installation, they must first verify the associated SLD diagram and decide what safety precautions to take. Therefore, it's important to comprehend these designs correctly.

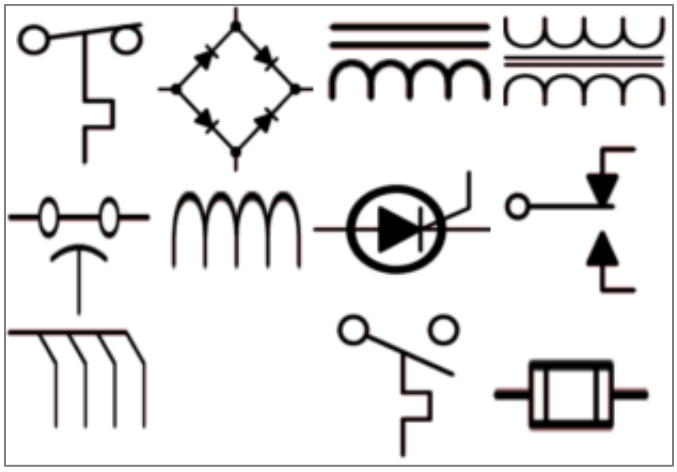


Fig. 5. Example of asymmetrical symbols found in SLDs.

The original data's big SLD sheets are 7500x5250 pixels in size. To expedite training, we divided the sheet into a 6x4 grid, resulting in 24 patches of sub-images that were minuscule in comparison to the original images (1250×1300).

The data generated by the annotation is kept in a file that corresponds to the 22 different classes. The width and height of the symbols that the bounding boxes enclose, as well as the center x and y values of the bounding boxes, were recorded as data. The collection of 800 images includes 12,500 samples, which represent 22 different types of symbols. The initial sample is severely imbalanced, as shown in Fig. 6.

A deep learning model needs to be fully annotated to be ready for training. To do this, we used the LabelBox program to annotate the set of SLD photos, as shown in Fig. 7. Twenty two different symbols in the total collection were annotated. Using the LabelBox tool to record the classes of the associated symbols and their locations is a simple approach for annotating a diagram.

In some instances, the distinctions between the symbols can be very significant. For instance, the dataset contains 1340 instances of generator symbols but only 99 and 117 instances of each disconnect and load symbol. Although delta and capacitor are present in the sample more than 800 times each, inductor and voltmeter are only present 203 and 212 times respectively. Three symbols that were significantly underrepresented overall were not included in the first trial (i.e. appears only once or twice in split sets).

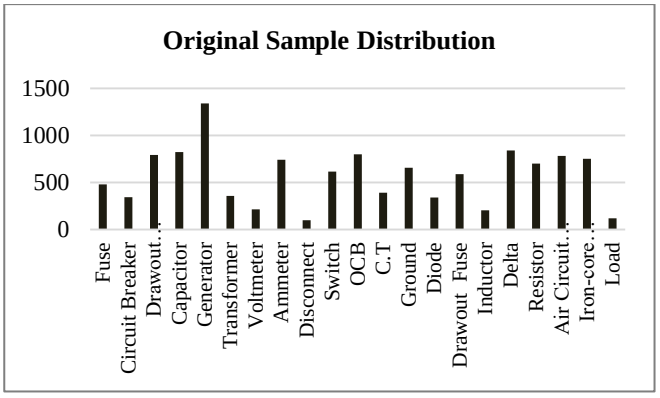


Fig. 6. Sample distribution in the original SLD dataset.

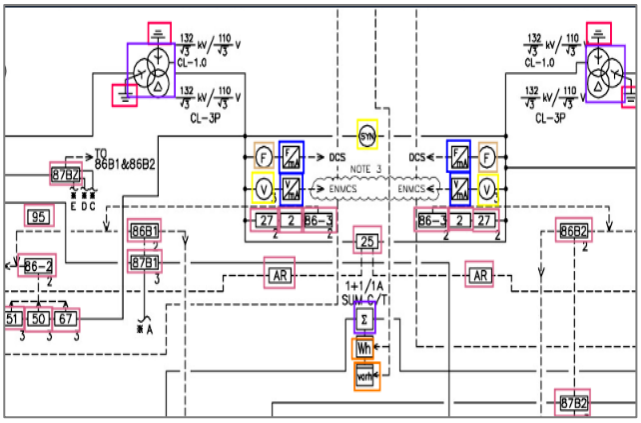


Fig. 7. An SLD annotated using labeling tool.

C. Symbol Detection

The YOLO approach was favored due to two key reasons. Firstly, it has a simple architecture that enables the prediction of multiple bounding boxes and class probabilities simultaneously using a single convolutional neural network. Secondly, YOLO is known for its high speed in comparison to other object detection techniques, which is essential for practical use in testing SLDs that contain an average of 50 engineering symbols.

1) YOLOv8 architecture: At the time this paper was being written, Ultralytics was actively working on YOLOv8 as they addressed community concerns and added new features. Glenn Jocher, the creator of YOLOv8, also discussed the developer-friendly features of YOLOv8 [54]. YOLOv8 comes with a CLI that enables training a model easier, in contrast to other models where chores are separated across numerous executable Python files. The addition of new convolutional layers and YOLOv8's Anchor Free Detection are further features of the software.

Since they may represent the distribution of the boxes from the target benchmark but not the distribution of the custom dataset, anchor boxes were a notoriously difficult component of older YOLO models. YOLOv8 is an anchor-free model, in contrast. In other words, rather than predicting an object's offset from a known anchor box, it predicts the object's center directly. To expedite Non-Maximum Suppression (NMS), a challenging post-processing procedure that sorts through candidate detection following inference, anchor-free detection decreases the number of box predictions [51, 55].

Using Equation (4), the bounding box's position is determined:

$$\cup_{x}^{y} \mathrm{P}\ x,y * IOU_{Predicted}^{Groun\ Truth} \quad (4)$$

According to Equation (4), x and y denote the yth bounding rectangle of the xth grid. The probability value assigned to the yth bounding box of the xth grid is $\cup$. If the yth bounding box contains an object, then Px,y is assigned a value of 1; otherwise, it is assigned a value of 0. The IoU between the predicted class and the actual ground truth is referred to as the IoUgroundtruth, and a greater IoU typically corresponds to more accurate predicted bounding boxes.

The bounding box, categorization, and confidence loss functions are combined to form the YOLOv8 loss function. The total loss function of the YOLOv8 is represented by Equation (5) [42]:

$$loss_{YOLOv5} = loss_{boundingbox} + loss_{classification} + loss_{confidence} \quad (5)$$

The stem's primary construction block, C2f, took the place of C3, and the first 6x6 conv is now a 3x3. Below is a diagram summarizing the module, where "f" represents the number of features, "e" represents the rate of growth, and CBS is a block made up of a conv, a BatchNorm, and a SiLU later. All of the

bottleneck's outputs are concatenated in C2f. C3 merely utilized the output of the previous bottleneck.

The first conv's kernel size was changed from 2x2 to 3x3, but the bottleneck remains the same as in YOLOv8. We might infer from this data that YOLOv8 is beginning to return to the ResNet block that was established in 2015. Features are directly concatenated in the neck without being forced to have the same channel dimensions. By doing this, the parameters count and tensor size as a whole are decreasing. YOLOv8 enhances photos while you're training online. The model views a slightly different variety of the images it has been given at each epoch.

2) Multi-fake class generation: Class imbalance has been a subject of extensive research, and various techniques have been developed, ranging from simple data augmentation and sampling to more sophisticated approaches like GAN [56]. In this study, we are utilizing MFC-GAN to generate more classes to handle the imbalance problem.

Our goal is to adopt a method similar to the MFC-GAN approach introduced in [57] to tackle the problem of class imbalance in the dataset of engineering symbols, specifically at the classification level.

The very little and occasionally subtle differences between the various classes of symbols led to the selection of this paradigm. We may train the discriminator using the MFC-GAN model to categorize both actual and false symbols, which allows for more precise discrimination across cases, as seen in Fig. 8.

By conditioning the generator on attribute labels, control generation was accomplished. Numerous studies involving various sample sizes in the minority classes, notably the goatee and eyeglass classes, were conducted. The MFC-GAN model is trained from scratch for each run, and samples are created following the end of the training.

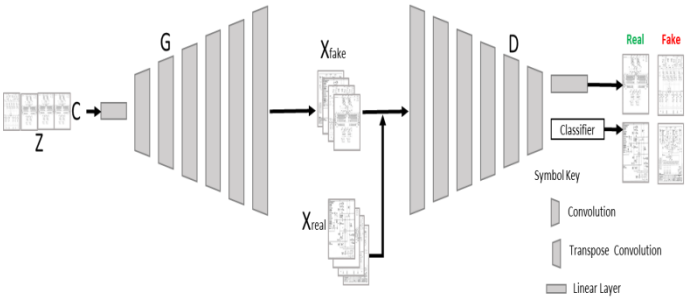


Fig. 8. Framework design for multi-fake class GAN.

The discriminator network for this study is built with four convolutional layers with two-stride spacing and uses batch normalization in between layers. Leaky ReLu with an alpha of 0.2 is used to activate all convolution layers, and the Sigmoid function is employed as the activation function in the final layer.

The classifier model generates a 2xN soft-max output for N classes and shares the discriminator layers with it. The generator is constructed using five transpose convolution layers with a stride of two and one linear layer. All the layers except the last one are activated using Leaky ReLu, and the final layer is activated using a sigmoid function. Batch normalization is applied between adjacent layers.

The generator of the GAN model takes a noise vector with a size of 100 as input along with symbol label encoding, which is similar to the input of most GAN models. The label encoding is important for class-specific generation, which is a significant aspect of our experiment.

The generator produces a 64x64 image of greyscale symbols. A batch size of 100 and a learning rate of 0.001 were used, which were selected through experimentation. Both the discriminator and the generator employed spectral normalization. Eq. (6), (7), and (8) will be used to train the suggested model.

$$Ls=E[logP(S=real|Xreal)]+E[logP(S=fake|Xfake)] \quad (6)$$

$$Lcd=E[logP(C=c|Xreal)]+E[logP(C'=c'|Xfake)] \quad (7)$$

$$Lcg=E[logP(C=c|Xreal)]+E[logP(C=c|Xfake)] \quad (8)$$

Where Ls denotes the chance that the sample is real or fraudulent and is used to determine the sampling loss. The losses for classification of both the generator and discriminator are calculated using Lcd and Lcg. The set of created images is called Xfake, and Xreal represents the training data.

3) Architecture of YOLOv5: In this study, the YOLOv5 algorithm was utilized, which is one of the most recent variations of the YOLO algorithm [44]. This algorithm is a speedy and effective system for identifying objects and locating them instantly. Since the symbols present in SLDs have a high degree of similarity, rapid detection is also necessary, which the YOLOv5 algorithm can fulfill. The system was built using the PyTorch deep learning framework, which has excellent detection performance and has simplified the process of training and testing specialized datasets. The YOLOv5 algorithm comprises three components: the head, the neck, and the backbone [45].

For our investigation, we opted to utilize the YOLOv5 detection model because of its straightforwardness and transparency. YOLOv5 created CSPDarknet, which formed the core of the network [58], by combining Darknet with the cross-stage partial network (CSPNet) [43]. CSPNet addresses the issue of recurrent gradient information in large-scale backbones by integrating gradient changes into the feature map, which decreases the model's parameters and FLOPS (floating-point operations per second). This ensures inference speed and accuracy while also reducing model size, which is crucial for accurate and speedy recognition of sperm cells. Furthermore, the YOLOv5 incorporates a path aggregation network (PANet) [59] as its neck to improve information flow. PANet employs a novel feature pyramid network (FPN) architecture with an enhanced bottom-up method-ology to increase low-level feature propagation. Adaptive feature sharing connects the feature grid to each feature level, ensuring that the downstream subnetwork receives meaningful data from every feature level. In addition, PANet enhances precise

localization signals at lower levels, considerably improving the object's location accuracy. The head of YOLOv5, the YOLO layer, generates three different sizes of feature maps to enable multi-scale prediction, allowing the YOLO model to handle small, medium, and large objects [58].

The CSPNet provides the framework for this algorithm. Because of the simplified model of CSPNet, fewer hyperparameters and FLOPS are produced, and the disappearing and ballooning gradient issues caused by complex neural networks are addressed. These enhancements improve the effectiveness and accuracy of object recognition inference. CSPNet has various features, including multiple convolutional layers, three convolutions in four CSP blocks, and spatial pyramid sharing. The CSPNet is responsible for extracting features from an input image, pooling and convolving that data to create a feature map. Consequently, in YOLOv5, the backbone serves as a feature generator [60].

The neck or core segment of YOLOv5 is referred to as the PANet. Its main function is to collect all the features obtained from the backbone, maintain them, and send them to the deeper layers to perform feature fusions. These feature fusions are then passed on to the head for object recognition, allowing the output layer to be aware of the high-level characteristics.

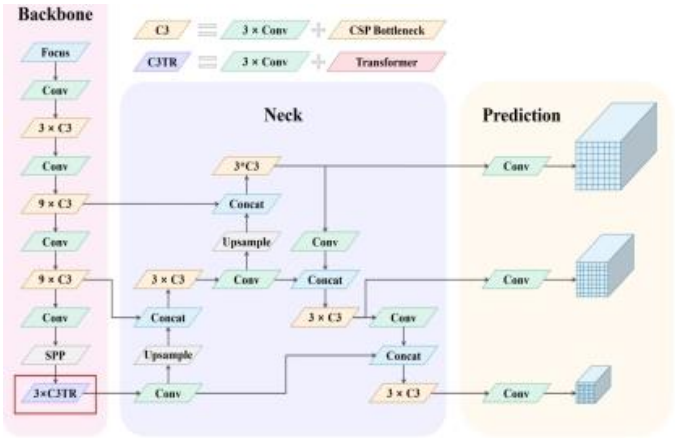


Fig. 9. Network architecture of YOLOv5 model.

The YOLOv5's head is responsible for identifying objects. It places bounding boxes and a class probability score around the target item, which is determined by 1x1 convolutions. The overall architecture of YOLOv5 is depicted in Fig. 9.

The position of the bounding box is established using Equation (9):

$$\cup_x^y = \text{P x,y} * IOU_{Predicted}^{Groun\ Truth} \quad (9)$$

Equation (9) demonstrates that the yth bounding box of the xth grid is defined by x and y. The yth bounding box of the xth grid has probability value. Px,y equals 1 when a subject is present within yth bounding box; otherwise, it is equal to 0. The IoUgroundtruth is the IoU that exists among the predicted class and the actual data. Higher IoUs are related to more accurate predicted bounding boxes.

The loss function of YOLOv5 is produced by merging the bounding box, classification, and confidence loss functions. The combined loss function of YOLOv5 is shown in Equation (10) [46].

$$\text{loss}_{\text{YOLOv5}} = \text{loss}_{\text{bounding box}} + \text{loss}_{\text{classi f ication}} + \text{loss}_{\text{con f idence}} \quad (10)$$

Equation (11) is used to determine the $\text{loss}_{\text{boundingbox}}$.

$$\text{loss}_{\text{bounding box}} = \lambda_{\text{if}} \sum_{a=0}^{b^2} \sum_{c=0}^{d} E_{a,c}^{g}, \text{h}_g(2\text{-k}_a\text{k}_{na})[(\text{x}_a\text{-}x'^c_a)^2 + (\text{y}_a\text{-}y'^c_a)^2 + (\text{w}_a\text{-}w'^c_a)^2 + (\text{h}_a\text{-}h'^c_a)^2] \quad (11)$$

Equation (11) uses h' and w' to denote the width and height of the target item, while xa and ya denote the coordinates of the target object in an image. Lastly, the indicator function (λif) shows whether the bounding box contains the target object.

The $\text{loss}_{\text{classification}}$ method is shown in equation (12):

$$\text{loss}_{\text{classi f ication}} = \lambda_{\text{classification}} \sum_{c=0}^{b^2} \sum_{c=0}^{d} E_{a,c}^{g} \sum_{\text{CCc1}} \text{L}_a(\text{c}) \log(\text{LL}_a(\text{c})) \quad (12)$$

$\text{loss}_{\text{confidence}}$ is determined using Equation (13):

$$\text{loss}_{\text{confidence}} = \lambda_{\text{confidence}} \sum_{c=0}^{b^2} \sum_{c=0}^{d} E_{a,c}^{Confidence} (\text{c}_i - \text{c}_l)^2 + \lambda_g \sum_{c=0}^{b^2} \sum_{c=0}^{d} E_{a,c}^{Confidence} (\text{c}_i - \text{c}_l)^2 \quad (13)$$

In Equations (12) and (13) show the symbols that represent confidence and signify the category loss coefficient λ, classification loss coefficient, and confidence score.

IV. Results and Experiments

This section can be separated into two experiments and should present a clear and accurate depiction of the experimental findings, their analysis, and the conclusions that can be drawn from the experiments.

There were two experiments done. The initial test was created to assess a complete method for identifying symbols in engineering drawings. In this context, the aim is to enhance the overall efficiency of analyzing a collection of drawings by detecting and identifying symbols, which is an important task as symbols make up a significant portion of these drawings. This can aid in completing other tasks, such as detecting text, pipelines, etc. The second experiment is different from the first, as it concentrates on using GAN-based methods to deal with the problem of class imbalance.

A. Symbol Detection Using YOLOv8

The SLD sheets in our dataset had a size of about 7500x5250 pixels. To avoid using computationally expensive training data, the SLDs were divided into 24 patches by multiplying their original width by 6 and their original height by 4. The resulting patch size was approximately 1250×1300 pixels. The annotations for the entire SLD were used to retrieve data for each patch's annotations, as described in the preceding section.

Symbols that spanned multiple patches were excluded from the training phase. After extensive testing, the third version of the YOLO framework was selected as it showed better detection rates for small objects compared to the previous versions. It should be emphasized that, when compared to the

entire image size, the technical symbols in our dataset are relatively small.

To conduct the experiment, the researchers utilized a recent version of YOLO architecture. Initially, they configured the total number of classes to be 22 in all three YOLO layers, and then adjusted the number of filters to 3 (referred to as $Class_{no}$ 5), where $Class_{no}$ represents the complete number of classes present in the dataset.

The dataset was divided into two sets: training set with 640 SLDs and test set with 160 SLDs, with a ratio of approximately 80:20. A pre-trained YOLO network was used and fine-tuned on our dataset by adjusting all layers. YOLO was implemented using PyTorch. To enhance object detection for various object sizes, it was observed that changing the input size during training is effective [47]. In this study, the network input size was modified to 416x416 after every 10 batches, and the training stopped after 10,000 batches. The learning rate was 0.001, and the batch size was 64.

During the testing phase, the model input size was increased from 416×416 to 2400×2400. This enabled us to perform symbol detection on the original SLD images instead of integrating detection from the SLD patches, thereby simplifying the symbol detection process for an entire SLD diagram in one step. To evaluate the model, we experimentally set the Intersection over Union (IoU) threshold to 0.5 and compared the detected symbols with the ground truth. A Python-based front-end was developed utilizing OpenCV and other libraries to analyze and display manual errors.

B. Training Evaluation of YOLOv8

1) Computer hardware configuration: GPU computing is a preferred choice for processing deep learning on a PC [50], and therefore, strong hardware support is required for deep learning networks. The training and generation processes were conducted on a GPU workstation that ran on Linux, CUDA 11.1, Python 3.8, and PyTorch 1.8.0, and was equipped with an Nvidia A40 4 48 GB GPU.

C. YOLOv8 Detection Results

The training phase produced an accuracy of 96%, while the testing phase produced an accuracy of 95.9%, with 11987 out of 12500 symbols in the test set correctly detected and recognized. The loss matrix for the training and validation sets indicates that the most of the instances of the classes were identified and detected accurately, indicating that symbols with sufficient training instances were correctly identified. An example output from the proposed methods is shown in Fig. 10, with different symbols highlighted in different colors.

In this case, the identified symbols were labeled with numbers, and the labels predicted by the models were noted down to compare them with the actual labels later. These symbols comprised various electrical components like switches, generators, motor, re-lays, inductors, as well as input/output labels such as "label_to" and "label_from".

Table I in the paper contains details about the number of symbols in both the training and testing datasets, with columns labeled "No. of Training Samples" and "No. of Testing Samples". It also displays the accuracy achieved for each class in the testing set, along with the number of symbols that were correctly identified, listed under "Correctly Detected Samples" or "Class Accuracy".

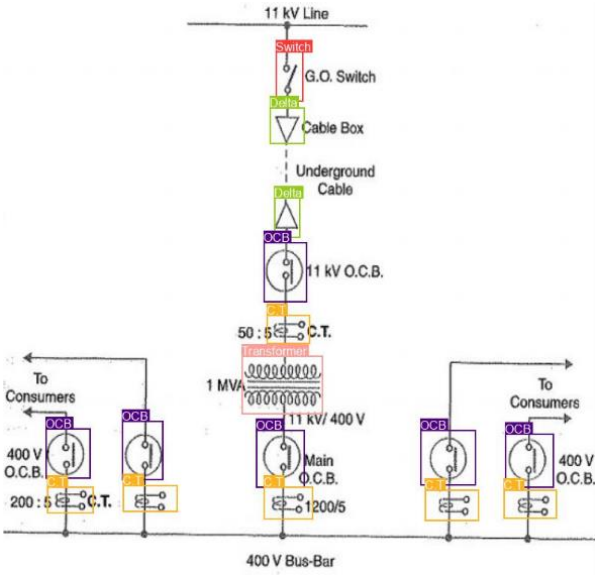


Fig. 10. Symbol detection using YOLOv8.

TABLE I. YOLOV8 DETECTION RESULTS BASED ON ORIGINAL SLD DATASET

Symbols	No. of Training Samples	No. of Testing Samples	Class Accuracy
Fuse	400	80	100%
Circuit Breaker	300	52	100%
Drawout Circuit Breaker	600	190	99%
Capacitor	601	20	99%
Generator	940	400	100%
Transformer	305	50	97%
Voltmeter	182	20	94%
Ammeter	550	190	100%
Disconnect	89	10	94%
Switch	501	115	100%
OCB	509	290	100%
C.T	309	80	100%
Ground	515	140	100%
Diode	310	30	98%
Drawout Fuse	679	110	100%
Inductor	185	18	98%

The results of the study show that most instances (11987 out of 12500) were accurately detected and identified. Fig. 10 displays different symbols from various SLD diagrams and how they can be accurately recognized regardless of their orientation. Some symbols, such as transformers, OCB, C.T, ground, and delta components, have various orientations, but the suggested approach can correctly detect and identify them. Even in situations where the text overlaps, resistors and ground

symbols are accurately detected, demonstrating the approach's resilience to innate visual issues, at least in this context, as opposed to conventional methods.

The round-shaped motor symbol in the testing set was misclassified as a generator symbol in all five instances due to their similarity. It is expected that increasing the number of training examples for this symbol, as well as other symbols in the majority class (such as switch, relay, C.T, inductor, voltmeter, load, etc.), would improve their detection rates.

To conclude, based on the results presented in Table I, it can be inferred that the detection rate for the symbols load, resistor, and motor was very low. This could be attributed to the fact that there were only a few training samples for these symbols. The complexity of the problem notwithstanding, the average accuracy of the remaining 22 symbols in the dataset was above 95%, which is a positive result, although these symbols were excluded.

D. MFC-GAN for Sample Generation and Symbol Detection

The purpose of this study is to evaluate a GAN-based model that can tackle the class imbalance problem in the dataset, which is a classification problem as opposed to a recognition task like in the first experiment. The main aim of this experiment is to utilize the MFC-GAN model to generate more symbols, which can then be added to the training set to improve classification accuracy.

Experiment 2 made use of a dataset that was very similar to the one utilized in Experiment 1, with all symbols being resized to 64x64 grayscale images. The problem was framed as a supervised learning task, where the goal was to learn a function f(x) that maps a given engineering symbol instance (xi) to its corresponding class (yi). In this case, the 22 symbols in the dataset were represented as a discrete set of classes denoted as Y, where yi belongs to Y. The dataset suffered from severe class imbalance, as previously mentioned, with some instances such as the angle choke valve being present in less than 0.01% of the dataset.

In this experiment, the MFC-GAN model was employed in two stages, namely the GAN training stage and the classification stage for this experiment. The purpose was to address the issue of class imbalance in the dataset. The MFC-GAN model was initially trained using all the samples in the dataset, with a focus on the symbols with the least representation. These symbols included fuse, circuit breaker, drawout circuit breaker, capacitor, generator, transformer, voltmeter, ammeter, disconnect, and switch. The numbers of occurrences of these symbols in the training set were 48, 344, 790, 1340, 821, 355, 212, 740, 99, and 616 respectively. The model underwent training only once on this dataset, and the generated samples were obtained after the training. To improve the learning of minority instance structure while training, the less represented classes were resampled.

Using the MFC-GAN model trained on the least represented symbols in the dataset, symbols from the minority class were generated. Eight symbols with the least representation were selected. To create a balanced dataset, 80% of the original dataset was used for training, and the remaining 20% was kept for testing. The artificially generated symbols were added to the training set, providing more than 4,000 additional synthetic samples for each minority class. This allowed the dataset to be rebalanced by increasing the prevalence of the least represented symbols.

Our goal is to evaluate the effectiveness of the synthesized symbols by comparing the performance of a classification model trained before and after the inclusion of these sym-bols in the training set.

The experiment employed a four-layer CNN classification model, consisting of three convolution layers with 32, 64, and 128 outputs, respectively. The kernel sizes for these layers were 3x3, 2x2, and max-pooling. The fourth layer was a fully connected layer with 256 units, which represented the 22 symbol classes, and fed into a 22-way Soft-max out-put. The model was trained using SGD with 64 batches and a learning rate of 0.001. The model's classification performance was assessed using standard measures like true positive rate, balanced accuracy, G-mean, and F1-Score, with the aim of comparing the model's performance before and after incorporating the generated symbols into the training dataset.

1) Results: Fig. 11 displays a comparison between the original symbols in the diagram and the symbols generated by the MFC-GAN model.

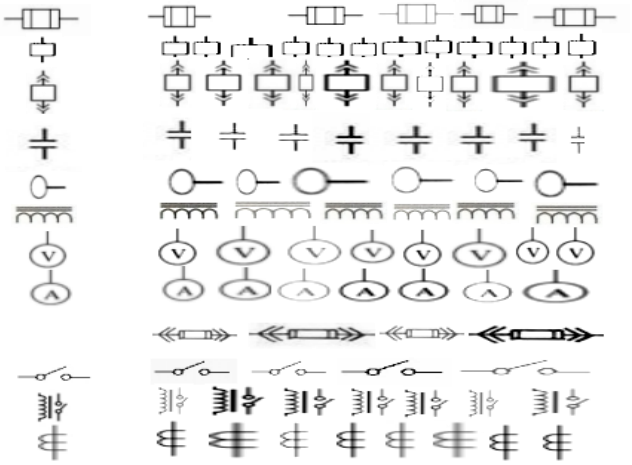


Fig. 11. Original SLD samples compared with MFC-GAN generated samples.

The comparison between the MFC-GAN generated symbols and the original symbols of the diagram is depicted in Fig. 11. The generated symbols by MFC-GAN were found to be more accurate and precise compared to symbols generated by other methods. The generated samples had clear symbol traits and distinct categories formed in each instance. Moreover, these high-quality samples resulted in an improved performance of the classifier. For example, Table III shows that for the angle disconnect, which had only 99 instances of the class, the accuracy improved from 0 to 94%. Similarly, seven out of eight minority classes demonstrated similar improvements. However, the MFC-GAN model did not improve the baseline in the load and inductor classes. It was observed that certain symbols such as OCB, capacitor, voltmeter, and ammeter exhibited significant similarity despite being uniquely generated, which hindered the classifier's ability to classify the load and inductor classes. This

observation was further supported by the low precision data in Table III for these classes.

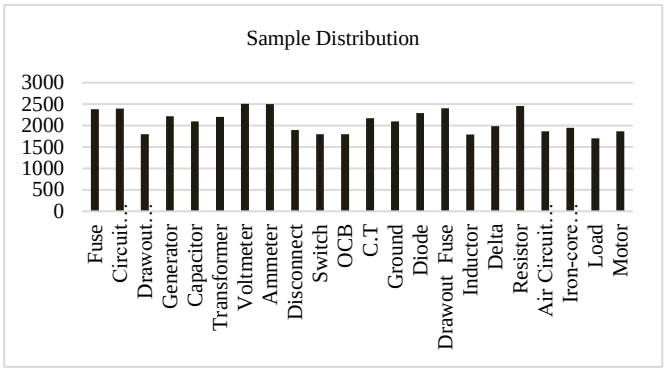


Fig. 12. Sample distribution in MFC-GAN generated dataset.

In this study, MFC-GAN models were able to produce occurrences of minority classes that were significantly underrepresented in the dataset, as demonstrated in Fig. 12. Both subjective and objective methods were used to evaluate the generated samples, including an assessment of the classifier's performance before and after incorporating the samples into the training sets. The results showed an improvement in performance across several commonly used evaluation parameters. However, it should be acknowledged that the class imbalance issue can only be addressed to some extent by MFC-GAN, and other strategies may need to be explored and utilized.

The study categorizes the outcome of classification predictions into four groups based on the relationship between the predicted output and the actual value: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). To assess the efficacy of defect detection, the study calculates the precision, recall, and F1 score of the model for different types of faults. Precision rate, which measures the accuracy of detection findings, is computed by dividing the number of symbols expected to be positive by the total number of symbols predicted to be positive. Recall rate, which gauges the thoroughness of detection findings, is calculated by dividing the number of samples expected to be positive by the total number of samples that actually have a positive value. Precision and recall are given in Equations (14) and (15):

$$\text{Precision} = \frac{TP}{TP + FP} \tag{14}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{15}$$

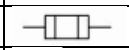
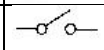
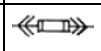
To evaluate classification problems, it is essential to take into account both the accuracy of identification and the completeness of detection. The model is evaluated using the F1 score, which considers both precision and recall. Equations (16) and (17) express accuracy as the number of symbols that are correctly identified.

$$\text{F1-score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \,.\text{Precision} \,.\text{Recall}}{\text{Precision} + \text{Recall}} \tag{16}$$

$$\text{Accuracy} = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 * Precision * Recall}{Precision + Recall}\text{s} \tag{17}$$

The results of the detection data are shown in Table II. The attributes represent the actual class, and each row represents the predicted category. The total number of symbols for each category is calculated by adding up the numbers in each column. The predicted category and the total number of predicted symbols for that category are shown in each row. Our proposed method can accurately detect the majority of single line-based engineering symbols. In this study, the precision of symbol detection for all types is over 95%, the average recall rate is 93.67%, and the F1 score is above 0.9. The average frame detection time is 0.074 seconds, while the average recall and precision rates are 90.67% and 0.074 seconds, respectively.

TABLE II. SYMBOLS GENERATED USING MFC-GAN

Symbol Name	Symbol	Original Instances	Generated Instances
Fuse		480	2380
Circuit Breaker		344	2400
Drawout Circuit Breaker		790	1800
Capacitor		821	2011
Generator		1340	2219
Transformer		355	2200
Voltmeter		212	2587
Ammeter		740	2500
Disconnect		99	1900
Switch		616	1800
OCB		799	1800
C.T		389	2178
Ground		655	2100
Diode		340	2291
Drawout Fuse		589	2408
Inductor		203	1790
Delta		840	1990
Resistor		700	2455
Air Circuit Breaker		780	1870
Iron-core Inductor		750	1950
Load		117	1701
Motor		541	1870

V. DISCUSSION

This section may be divided into two subsections which provide comparisons and conclusive remarks on the experiments.

A. Comparison with Different Data Augmentation Techniques

Table II displays the frequency of the different classes in both datasets. After generating synthetic SLD images using MFC-GAN, it can be observed that the new samples are approximately balanced. However, to address the issue of class imbalance, the original dataset could be improved by including additional distinct and separate photos.

TABLE III. YOLOV5 CLASSIFICATION PERFORMANCE OF SYMBOLS ON AUGMENTED DATASET

Metric	Switch	Relay	Motor	Generator	Load	Inductor	Fuse	Resistor
Precision	1.00	1.00	0.85	0.89	1.00	1.00	1.00	1.00
Recall	0.94	0.89	0.92	1.00	1.00	0.90	0.91	0.87
F1-score	0.89	0.94	0.91	0.95	0.88	0.90	0.92	0.91
Accuracy	1.00	0.98	0.99	1.00	1.00	1.00	1.00	1.00

We conducted an experiment to test the accuracy and performance of YOLOv8 in various conditions and configurations using 800 images, and an example of the detection results can be seen in Fig. 13. The accuracy testing and performance of the experiment using images from our datasets are displayed in Table IV. YOLOv5 is generally more accurate than its recent version. Group 2, which is the augmented dataset, had the highest average accuracy of 96% when using YOLOv5, with only five detection errors. YOLO's performance can be improved by utilizing a large dataset that includes both real and synthetic images generated by GANs. When a deep learning-based method is trained on a small and insufficient dataset, it may result in overfitting and difficulties in mapping the object [52]. Adding noise or generating fake images during training can make the process of learning the input image from the output image easier, reducing general errors, and enhancing the training component [53]. Thus, to improve item identification accuracy, it is necessary to include synthetic images in addition to actual photographs.

B. Missed Detection

Table V presents the outcomes of the evaluation of the model on the SLD components dataset, revealing that there were a total of 52 instances where the SLD components were either absent or wrongly classified as some other symbols. Out of these occurrences, eight symbols were misclassified, while the remaining 46 symbols were not detected entirely. This issue can be attributed in part to the nature of some drawings, where symbols are nearly completely obscured by text and comments. The Intersection over Union (IOU) metric in Table V confirms that these missed symbols have a zero IOU, indicating that they were not detected by the model.

TABLE IV. COMPARISON OF YOLOV8 AND YOLOV5 CLASS ACCURACY

Dataset	Accuracy	Wrong Detection	Missed Detection
Original	95%	8	46
Augmented	96.3%	3	2

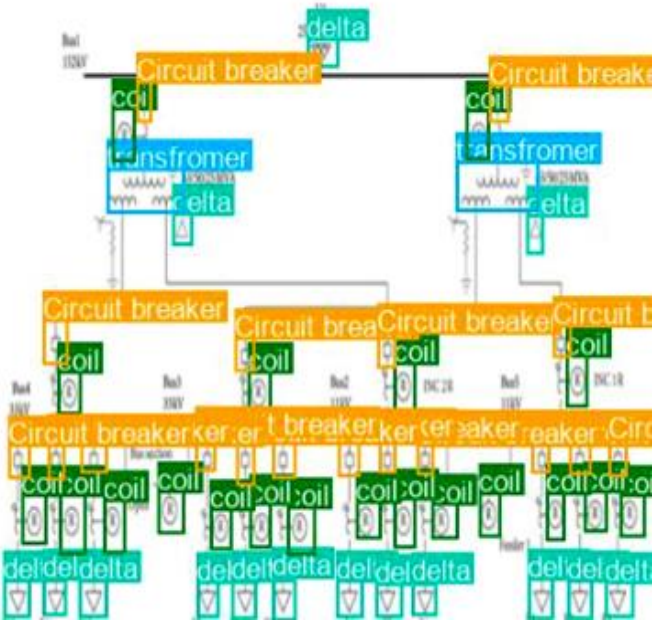


Fig. 13. Symbols detected in augmented dataset.

After conducting a visual inspection of the data in Table V, some symbols were found to be mislabeled. Specifically, when reviewing the results for the switch representation, it was observed that the algorithm had predicted the correct class for symbols with an incorrect label. However, for some symbols, the algorithm had predicted the wrong class label.

TABLE V. SYMBOLS MISSED OR OVERLOOKED BY THE CLASSIFIERS

Class	No of Sample Occurrence		Predicted Sample Class		IOU	
	YOLOv8	YOLOv5	YOLOv8	YOLOv5	YOLOv8	YOLOv5
Disconnect	2	5	Switch	Ground	0.50	0.91
Load	1	3	-	Fuse	0.56	1.00
Inductor	2	4	-	-	-	0.00
Voltmeter	-	16	-	-	-	0.00
Diode	-	10	-	-	-	0.00
Circuit Breaker	-	8	-	-	-	0.00

VI. CONCLUSION AND FUTURE RECOMMENDATION

In this study, we proposed a system for analyzing and processing complex engineering drawings. Our approach achieved more than 96% accuracy in recognizing symbols on the drawings, based on extensive testing on a large collection of SLD sheets provided by an industry partner. We utilized advanced bounding-box detection techniques, which

demonstrated high accuracy in identifying symbols from 22 different categories, despite some of these symbols having only minor differences. To address class imbalance in the symbol dataset, we suggested a GAN-based model. Our experiments showed that our system could generate realistic engineering symbols, and adding this synthetic data to the training set improved classification accuracy. According to the experimental results, the proposed GAN model was capable of learning from a smaller amount of training in-stances.

The subsequent emphasis of this study will be on the application of GANs to the creation of symbols in a schematic environment. In future development, an integrated system will be created using the recommended methods to enable thorough analysis and processing of technical diagrams like SLD. This approach will make it much easier to per-form additional tasks such as line detection or text localization. Furthermore, future work will involve combining Explainable AI (XAI) and other GAN methods such as WGAN, CycleGAN, PCCGAN, and StyleGAN with other detection techniques.

ACKNOWLEDGMENT

We appreciate Yayasan UTP FRG (YUTP-FRG), grant number 015LC0-280, and Computer and Information Science Department of Universiti Teknologi PETRONAS for providing funding and support for this research.

REFERENCES

[1] BHANBHRO, H., HOOI, Y. K., HASSAN, Z., & SOHU, N., "Modern Approaches towards Object Detection of Complex Engineering Drawings," in Proc. International Conference on Digital Transformation and Intelligence (ICDI), IEEE, 2022.

[2] E. ELYAN, L. JAMIESON, AND A. ALI-GOMBE, "Deep learning for symbols detection and classification in engineering drawings," Neural networks, vol. 129, pp. 91-102, 2020.

[3] X. Y. , G. F. MENG, AND C. H. PAN, "Scene text detection and recognition with advances in deep learning: a survey," (in English), Int J Doc Anal Recog, vol. 22, no. 2, pp. 143-162, Jun 2019, doi: 10.1007/s10032-019-00320-5.

[4] S. MANI, M. A. HADDAD, D. CONSTANTINI, W. DOUHARD, Q. W. LI, and L. Poirier, "Automatic Digitization of Engineering Diagrams using Deep Learning and Graph Search," (in English), Ieee Comput Soc Conf, pp. 673-679, 2020, doi: 10.1109/Cvprw50498.2020.00096.

[5] C. F. MORENO-GARCIA, E. ELYAN, AND C. JAYNE, "New trends on digitisation of complex engineering drawings," (in English), Neural Comput Appl, vol. 31, no. 6, pp. 1695-1712, Jun 2019, doi: 10.1007/s00521-018-3583-1.

[6] T. M. NGUYEN, L. V. PHAM, C. C. NGUYEN, AND V. V. NGUYEN, "Object Detection and Text Recognition in Large-scale Technical Drawings," (in English), Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods (Icpram), pp. 612-619, 2021, doi: 10.5220/0010314406120619.

[7] J. K. NURMINEN, K. RAINIO, J.-P. NUMMINEN, T. SYRJÄNEN, N. PAGANUS, AND K. HONKOILA, "Object detection in design diagrams with machine learning," in International Conference on Computer Recognition Systems, 2019: Springer, pp. 27-36.

[8] A. REZVANIFAR, M. COTE, AND A. B. ALBU, "Symbol Spotting on Digital Architectural Floor Plans Using a Deep Learning-based Framework," (in English), Ieee Comput Soc Conf, pp. 2419-2428, 2020, doi: 10.1109/Cvprw50498.2020.00292.

[9] S. SARKAR, P. PANDEY, AND S. KAR, "Automatic Detection and Classification of Symbols in Engineering Drawings," arXiv preprint arXiv: 2204.13277, 2022.

[10] Q. S. Wang, F. S. Wang, J. G. Chen, and F. R. Liu, "Faster R-CNN Target-Detection Algorithm Fused with Adaptive Attention Mechanism," (in Chinese), Laser Optoelectron P, vol. 59, no. 12, Jun 2022, doi: 10.3788/Lop202259.1215016.

[11] L. H. WEN AND K. H. JO, "Fast LiDAR R-CNN: Residual Relation-Aware Region Proposal Networks for Multiclass 3-D Object Detection," (in English), Ieee Sens J, vol. 22, no. 12, pp. 12323-12331, Jun 15 2022, doi: 10.1109/Jsen.2022.3172446.

[12] CINTRA, R. J., DUFFNER, S., GARCIA, C., AND LEITE, AL., "Low-complexity Approximate Convolutional Neural Networks," IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 12, pp. 5981–5992, 2018.

[13] KHAN, S. H., HAYAT, M., BENNAMOUN, M., SOHEL, F. A., AND TOGNERI, R., "Cost-sensitive Learning of Deep Feature Representations from Imbalanced Data," IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 8, pp. 3573–3587, Aug. 2018.

[14] STUHLSATZ, A., LIPPEL, J., & ZIELKE, "Feature Extraction with Deep Neural Networks by a Generalized Discriminant Analysis," IEEE Trans. Neural Netw. Learn. Syst., vol. 23, no. 4, pp. 596–608, Apr. 2012.

[15] REN, S., HE, K., GIRSHICK, R., & SUN, J., "Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks," in Proc. NIPS, 2015, pp. 91–99.

[16] REDMON, J., DIVVALA, S., GIRSHICK, R., & FARHADI, A.., "You Only Look Once: Unified, Real-time Object Detection," in Proc. CVPR, 2016, pp. 779–788.

[17] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in Proc. CVPR, 2005, pp. 886–893.

[18] P. F. FELZENSZWALB et al., "Object Detection with discriminatively Trained Part-based Models," IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[19] M. EVERINGHAM et al., "The Pascal Visual Object Classes (VOC) Challenge," Int. J. Comput. Vis., vol. 88, no. 2, pp. 303–338, 2008.

[20] G. E. HINTON AND R. R. SALAKHUTDINOV, "Reducing the Dimensionality of Data with Neural Networks," Science, vol. 313, no. 5786, pp. 504507, 2006.

[21] VIG, E., DORR, M., & COX, D., "Large-scale Optimization of Hierarchical Features for Saliency Prediction in Natural Images," in Proc. CVPR, 2014, pp. 2798–2805.

[22] BHANBHRO, H., HOOI, Y. K., HASSAN, Z., & SOHU, N., "Modern Deep Learning Approaches for Symbol Detection in Complex Engineering Drawings," in Proc. International Conference on Digital Transformation and Intelligence (ICDI), IEEE, 2022.

[23] ELYAN, E., MORENO-GARCÍA, C. F., & JOHNSTON, P., "Symbols in Engineering Drawings (SIED): An Imbalanced Dataset benchmarked by Convolutional Neural Networks," in Proc. 21st EANN (Engineering Applications of Neural Networks), 2020.

[24] RICA, E., ALVAREZ, S., MORENO-GARCIA, C. F., & SERRATOSA, F., "Zero-Error Digitisation and Contextualisation of Piping and Instrumentation Diagrams Using Node Classification and Sub-graph Search," Springer International Publishing, August 26–27, 2023.2.6

[25] GUPTA, M., WEI, C., & CZERNIAWSKI, T., "Automated Valve Detection in Piping and Instrumentation (P&ID) Diagrams," in Proc. International Symposium on Automation and Robotics in Construction. Vol. 39. IAARC Publications, 2022.

[26] SHEN, C., LV, P., MAO, M., LI, W., ZHAO, K., & YAN, Z., "Substation One-Line Diagram Automatic Generation Based On Image Recongnition," in Proc. Global Conference on obotics, Artificial Intelligence and Information Technology (GCRAIT). IEEE, 2022.

[27] A. ALI-GOMBE AND E. ELYAN, "MFC-GAN: Class-imbalanced dataset classification using Multiple Fake Class Generative Adversarial Network," (in English), Neurocomputing, vol. 361, pp. 212-221, Oct 7 2019, doi: 10.1016/j.neucom.2019.06.043.

[28] E. ELYAN, L. JAMIESON, AND A. ALI-GOMBE, "Deep learning for symbols detection and classification in engineering drawings," (in English), Neural Networks, vol. 129, pp. 91-102, Sep 2020, doi: 10.1016/j.neunet.2020.05.025.

[29] V. NAOSEKPAM AND N. SAHU, "Text detection, recognition, and script identification in natural scene images: a Review," (in English), Int

J Multimed Inf R, vol. 11, no. 3, pp. 291-314, Sep 2022, doi: 10.1007/s13735-022-00243-8.

[30] H. BHANBHRO, S. R. HASSAN, S. Z. NIZAMANI, S. T. BAKHSH, AND M. O. ALASSAFI, "Enhanced Textual Password Scheme for Better Security and Memorability," (in English), Int J Adv Comput Sc, vol. 9, no. 7, pp. 209-215, Jul 2018.

[31] R. HUANG, J. GU, X. SUN, Y. HOU, AND S. UDDIN, "A rapid recognition method for electronic components based on the improved YOLO-V3 network," Electronics, vol. 8, no. 8, p. 825, 2019.

[32] H. LEE, J. LEE, H. KIM, AND D. MUN, "Dataset and method for deep learning-based reconstruction of 3D CAD models containing machining features for mechanical parts," Journal of Computational Design and Engineering, vol. 9, no. 1, pp. 114-127, 2022.

[33] S. E. WHANG, Y. ROH, H. SONG, AND J.-G. LEE, "Data collection and quality challenges in deep learning: A data-centric ai perspective," The VLDB Journal, pp. 1-23, 2023.

[34] J. WANG, Y. CHEN, Z. DONG, AND M. GAO, "Improved YOLOv5 network for real-time multi-scale traffic sign detection," Neural Computing and Applications, pp. 1-13, 2022.

[35] C. F. MORENO-GARCÍA, E. ELYAN, AND C. JAYNE, "Heuristics-based detection to improve text/graphics segmentation in complex engineering drawings," in Engineering Applications of Neural Networks: 18th International Conference, EANN 2017, Athens, Greece, August 25–27, 2017, Proceedings, 2017: Springer, pp. 87-98.

[36] L. JAMIESON, C. F. MORENO-GARCIA, AND E. ELYAN, "Deep learning for text detection and recognition in complex engineering diagrams," in 2020 International Joint Conference on Neural Networks (IJCNN), 2020: IEEE, pp. 1-7.

[37] M. F. THEISEN, K. N. FLORES, L. S. BALHORN, AND A. M. SCHWEIDTMANN, "Digitization of chemical process flow diagrams using deep convolutional neural networks," Digital Chemical Engineering, vol. 6, p. 100072, 2023.

[38] M. KARTHI, V. MUTHULAKSHMI, R. PRISCILLA, P. PRAVEEN, AND K. VANISRI, "Evolution of yolo-v5 algorithm for object detection: automated detection of library books and performace validation of dataset," in 2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), 2021: IEEE, pp. 1-6.

[39] Naosekpam, V.; Sahu, N. Text detection, recognition, and script identification in natural scene images: a Review. Int J Multimed Inf R 2022, 11 (3), 291-314. DOI: 10.1007/s13735-022-00243-8.

[40] Zhang, Q. R.; Zhang, M.; Chen, T. H.; Sun, Z. F.; Ma, Y. Z.; Yu, B. Recent advances in convolutional neural network acceleration. Neurocomputing 2019, 323, 37-51. DOI: 10.1016/j.neucom.2018.09.038.

[41] Antoniou, A.; Storkey, A.; Edwards, H. Data augmentation generative adversarial networks. arXiv preprint arXiv:1711.04340 2017.

[42] Amur, Z. H.; Hooi, Y.; Sodhar, I. N.; Bhanbhro, H.; Dahri, K. State-of-the Art: Short Text Semantic Similarity (STSS) Techniques in Question Answering Systems (QAS). In International Conference on Artificial Intelligence for Smart Community: AISC 2020, 17–18 December, Universiti Teknologi Petronas, Malaysia, 2022; Springer: pp 1033-1044.

[43] Amur, Z. H.; Kwang Hooi, Y.; Bhanbhro, H.; Dahri, K.; Soomro, G. M. Short-Text Semantic Similarity (STSS): Techniques, Challenges and Future Perspectives. Applied Sciences 2023, 13 (6), 3911.

[44] Baur, C.; Albarqouni, S.; Navab, N. MelanoGANs: high resolution skin lesion synthesis with GANs. arXiv preprint arXiv:1804.04338 2018.

[45] Buda, M.; Maki, A.; Mazurowski, M. A. A systematic study of the class imbalance problem in convolutional neural networks. Neural networks 2018, 106, 249-259.

[46] Denton, E. L.; Chintala, S.; Fergus, R. Deep generative image models using a laplacian pyramid of adversarial networks. Advances in neural information processing systems 2015, 28.

[47] Dong, Q.; Gong, S.; Zhu, X. Class rectification hard mining for imbalanced deep learning. In Proceedings of the IEEE international conference on computer vision, 2017; pp 1851-1860.

[48] Dosovitskiy, A.; Springenberg, J. T.; Riedmiller, M.; Brox, T. Discriminative unsupervised feature learning with convolutional neural networks. Advances in neural information processing systems 2014, 27. Douzas, G.; Bacao, F. Effective data generation for imbalanced learning using conditional generative adversarial networks. Expert Syst Appl 2018, 91, 464-471.

[49] Fernández, A.; López, V.; Galar, M.; Del Jesus, M. J.; Herrera, F. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. Knowledge-based systems 2013, 42, 97-110.

[50] Frid-Adar, M.; Klang, E.; Amitai, M.; Goldberger, J.; Greenspan, H. Synthetic data augmentation using GAN for improved liver lesion classification. In 2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018), 2018; IEEE: pp 289-293.

[51] He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2016; pp 770-778.

[52] Huang, C.; Li, Y.; Loy, C. C.; Tang, X. Learning deep representation for imbalanced classification. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2016; pp 5375-5384.

[53] Inoue, H. Data augmentation by pairing samples for images classification. arXiv preprint arXiv:1801.02929 2018.

[54] Karras, T.; Aila, T.; Laine, S.; Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 2017.

[55] Mariani, G.; Scheidegger, F.; Istrate, R.; Bekas, C.; Malossi, C. Bagan: Data augmentation with balancing gan. arXiv preprint arXiv:1803.09655 2018.

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

[57] Odena, A. Semi-supervised learning with generative adversarial networks. arXiv preprint arXiv:1606.01583 2016.

[58] Wan, L.; Wan, J.; Jin, Y.; Tan, Z.; Li, S. Z. Fine-grained multi-attribute adversarial learning for face generation of age, gender and ethnicity. In 2018 International Conference on Biometrics (ICB), 2018; IEEE: pp 98-103.

[59] Yue, Y.; Liu, H.; Meng, X.; Li, Y.; Du, Y. Generation of high-precision ground penetrating radar images using improved least square generative adversarial networks. Remote Sensing 2021, 13 (22), 4590.

[60] Thuan, D. Evolution of Yolo algorithm and Yolov5: The State-of-the-Art object detention algorithm. 2021.