Juanda Airport Runway Visibility Modeling Using Gan Based on Imbalanced Dataset

Nusaibatul Zahra[1], I. G. P. Asto Buditjahjanto[2], Unit Three Kartini[3], Hapsari Peni[4]
[1,2,3,4] Electrical Engineering Department, Universitas Negeri Surabaya
[1,2,3,4] A5 Building Ketintang Campus, Surabaya 60231, Indonesia
Email: nusaibatul.20037@mhs.unesa.ac.id, asto@unesa.ac.id, unitthree@unesa.ac.id

Abstract – Aviation safety and security are heavily influenced by airport visibility, as pilots require clear visual references for landing. However, poor weather conditions can reduce visibility and increase the risk of accidents. Therefore, an automated system is needed to classify visibility levels quickly and accurately, even when faced with the challenge of imbalanced datasets. This study employs a Generative Adversarial Network (GAN) approach, focusing on Vanilla GAN, DCGAN, and StyleGAN models. The data used is sourced from CCTV AWOS at Runway 10 of Juanda Airport, encompassing 14,458 images from the period of August 13 to 31, 2023. The models are evaluated using SSIM scores and feature extraction of color, texture, and HOG at various epochs. The results indicate that the Vanilla GAN model at 60 epochs is the most suitable for the minority class compared to the other models, based on feature evaluation, SSIM scores, synthetic image quality, and loss pattern outcomes. Its simple architecture aids in capturing low variation in the dataset, making it superior to more complex architectures like DCGAN and StyleGAN. Further optimization and architectural adjustments could enhance the results, especially for datasets with low variation like the one used in this study.
Keywords: Aviation, GAN, Dataset

I. INTRODUCTION

Security and safety are crucial in the world of transportation, especially in aviation or air transportation. Several factors, including visibility, affect safety and security, with the most critical being visibility range, wind direction, wind speed, temperature, dew point, and barometric pressure [1]. Visibility is the maximum horizontal distance a person with normal vision can see from the background under current weather conditions [2]. It is crucial to pay attention to visibility range when operating an aircraft. The Minister of Transportation Regulation Number KM 18 of 2010 stipulates that no one can operate an aircraft under VFR conditions if the visibility or distance from clouds exceeds the specified requirements [3].

Visibility at the airport is crucial for pilots because they need visual references for landing. In aeronautical-meteorological documentation, visibility is a measurable characteristic provided by an observer or an automated system and expressed in meters or kilometers. It defines the farthest distance from which an object with specific characteristics can be seen and recognized [4].

Poor weather conditions, such as dense fog or heavy rain, can reduce visibility and increase the risk of aviation accidents. Therefore, an automated system capable of quickly and accurately classifying visibility levels on the runway is necessary. However, classifying visibility levels from an imbalanced dataset is one of the main challenges in this research, as it can cause the classification model to be less effective in recognizing and classifying rare visibility situations. Consequently, specific strategies, such as oversampling, undersampling, or other dataset imbalance handling techniques, are needed to address this imbalance. The dataset used in this research is an imbalanced dataset obtained from the CCTV data of AWOS (Automatic Weather Observing System) at Juanda Airport, where the researchers focus on images from runway 10 of Juanda Airport.

Researching with an imbalanced dataset becomes problematic because, in real life, many classifications have similar problems. To resolve this, it is necessary to find the right way to balance the dataset before classifying [5]. Several techniques have been used to deal with imbalanced datasets, such as under- and oversampling. However, there are drawbacks to both methods. Minority data points are

randomly duplicated throughout the oversampling procedure in order to enhance their count [6]. Overfitting is frequently the result, while undersampling might eliminate significant components of the majority class, making it more challenging to learn the decision border between classes [7].

Due to the common occurrence of data imbalance in real-world situations, a dataset with imbalanced data was selected as the basis for studying visibility modeling using Generative Adversarial Networks (GAN). The availability of data limits most image processing and classification studies, and real-world events often lead to imbalanced class distributions. Considering the prevailing visibility circumstances, it is likely that high visibility scenarios will occur more frequently than bad visibility conditions, which aligns with the general patterns of weather.

Furthermore, the utilization of imbalanced datasets also takes into account the necessity of constructing models that may offer pragmatic resolutions for real-life scenarios. By prioritizing research on data imbalance, the results are anticipated to have greater relevance in tackling intricate everyday problems, particularly when the minority class exerts a substantial influence on decisions or actions. This research addresses both the technical difficulties in identifying visibility and the real-world dynamics where data imbalance is a crucial factor to consider in creating effective models.

The research aims to identify the optimal model by utilizing Generative Adversarial Networks (GANs) to generate synthetic images, hence addressing the issue of imbalanced datasets. Image quality assessment can be categorized into two methods: subjective and objective. Subjective assessment is a method of evaluating image quality that relies on human perception. This approach is not as successful and efficient, and it necessitates a significant amount of time to locate evaluators and wait for their assessments based on personal viewpoints. Objective assessment, in contrast, employs mathematical algorithms based on precise criteria to evaluate image quality [8].

Augmenting the minority class to attain equilibrium or approach the quantity of data points in the majority class is a crucial measure in tackling imbalanced datasets. This research entails conducting meticulous tests to assess the efficacy of different strategies in enhancing dataset balance.

II. METHODS

Imbalanced dataset

Data imbalance refers to a situation where certain classes have a substantially unequal amount of data in comparison to other classes. The class containing a larger amount of data is commonly referred to as the majority class, whilst the class containing a smaller amount of data is known as the minority class. The attributes of imbalanced data can undeniably impact the forecasting outcomes of an algorithm. The Imbalance Ratio (IR) can be calculated to assess the extent of data imbalance, as described in reference [9]. Equation 1 presents the mathematical expression used to determine the proportion between the minority class and the majority class.

$$Imbalanced\ Ratio\ (IR) = \frac{n_{majority}}{n_{minority}} \quad (1)$$

Generative adversarial network

Over the past twenty years, various techniques have been developed to study class imbalance. However, most of these techniques have focused on studying binary single-label class imbalance in small-scale data with a low-class imbalance ratio, typically in the range of 1:100 [10].

In 2014, Goodfellow and his colleagues developed the fundamental Generative Adversarial Network (GAN) using the multi-layer perceptron (MLP) network. The GAN framework comprises two components, namely a generator (G) and a discriminator (D), which together form a class of deep generative models. These two components can be regarded as differentiable systems, although they are commonly employed as the generator and discriminator in neural networks [11].

The neural network game is a contest where two neural networks compete against each other. The generative model has the ability to generate new instances of data, whilst the discriminative model is designed to differentiate between different types of data instances [12]. Researchers in diverse fields have acknowledged the significance of Generative Adversarial Networks (GANs) because of their capacity to represent intricate real-world image data. GANs are important because they have the capability to create artificial images and their adversarial learning notion is exciting. This concept shows that GANs have the capacity to rectify skewed datasets [13].

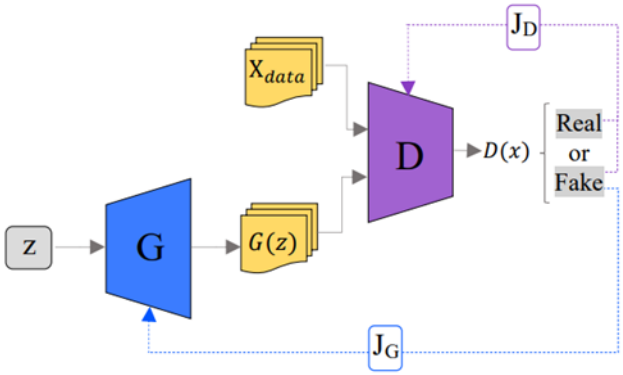

Figure 1. GAN block diagram

Figure 1 depicts a block diagram that illustrates the GAN employed in this research. The discriminator (D) serves as a binary classifier, differentiating between synthetic samples G(z) and authentic samples Xdata. The discriminator is trained to classify both genuine and counterfeit data with maximum accuracy. Put simply, when given actual data Xdata as input, the discriminator categorizes it as real data and outputs a numerical value that is very near to 1. On the other hand, if the input is data

that was created by the generator, the discriminator will classify it as fake data and provide a numerical value that is very near to 0 [14].

Feature extraction is the procedure of categorizing a picture database based on its content. Mathematically, every feature extraction is represented as an n-dimensional vector known as a feature vector. The feature vector's components are calculated by image processing and analysis techniques, enabling the comparison of one image to another. Feature extraction can be categorized into three distinct types: low-level, middle-level, and high-level. Low-level feature extraction involves analyzing visual elements such as color and texture. Middle-level extraction focuses on specific areas of a picture specified by segmentation. High-level extraction, on the other hand, relies on the semantic information included in the image [15].

Color feature extraction entails the examination of an image's color, which is composed of pixels that possess distinct color intensities. A histogram is used to represent the color distribution in each pixel. Histograms are employed as a technique for extracting features by leveraging the variations in pixel distribution within each image. The color histogram of an image is generated by converting the colors in the image into binary values and then tallying the number of pixels for each binary value. A color histogram is a visual depiction of the frequency of different colors contained in an image.

A color histogram of an image is created by dividing the colors in the image into separate groups and calculating the number of pixels in each group. The image data will be represented using color histograms in the RGB and HSV color spaces, with 8 bins in each color channel [3].

Hue, saturation, and value (HSV) are a collection of color characteristics that provide insight into how colors are perceived by the human eye. The HSV color properties can be understood as follows: hue represents colors in the range from red to green, saturation represents colors from red to pink, and value represents colors from black to white [16]. In order to ascertain the outcomes in the HSV color space, there exist multiple formulas for computing the values of the HSV color space as presented in Equations 2 to 5.

$$R = \frac{R}{255} \; G = \frac{G}{255} \; B = \frac{B}{255} \tag{2}$$

$$V\,(value) = max\; value \tag{3}$$

$$S\begin{Bmatrix} 0, if\; max = min \\ \frac{max-min}{v}, if\; max > 0 \end{Bmatrix} \tag{4}$$

$$H\begin{Bmatrix} 0, if\; max = min \\ 60°\left(\frac{G-B}{max-min} mod\; 6\right), if\; max = R \\ 60°\left(\frac{B-R}{max-min} + 2\right), if\; max = G \\ 60°\left(\frac{R-G}{max-min} + 4\right), if\; max = B \end{Bmatrix} \tag{5}$$

GLCM, short for Gray-Level Co-occurrence Matrix, is a statistical technique employed in texture extraction. It takes into account the spatial arrangement of pixels within a picture [17]. Out of the 14 features that may be computed using GLCM, three significant ones are contrast, homogeneity, and entropy.

Contrast is a quantitative measure of the degree of variance in the grey levels found in the co-occurrence matrix. When the intensity values of a pixel and its adjoining pixel are comparable, the texture contrast is significantly reduced. Homogeneity quantifies the degree of similarity in the gray-level intensity variations inside the picture co-occurrence matrix. Homogeneity is considered high when pixel pairings exhibit consistent gray-level values.

Entropy, which is the antithesis of homogeneity, quantifies the quantity of unpredictability exhibited by the gray-level intensities in the image co-occurrence matrix. The entropy value indicates the degree of roughness or smoothness of the surface texture [18]. In order to ascertain the outcomes of texture feature extraction, many equations can be employed, as demonstrated in Equations 6-8.

$$Contrast = \sum_i^{Ng} \sum_j^{Ng} (i-j)^2 p(i,j) \tag{6}$$

$$Homogeneity = \sum_i^{Ng} \sum_j^{Ng} \frac{p(i,j)}{1+(i-j)2} \tag{7}$$

$$Entropy = -\sum_i^{Ng} \sum_j^{Ng} p(i,j)\, log(p(i,j))^2 \tag{8}$$

The Histogram of Oriented Gradient (HOG) is a feature descriptor used in computer vision and image processing for object detection and shape recognition. HOG works by capturing the histogram of gradient orientations within localized portions of an image. It was developed by Navneet Dalal and Bill Triggs in 2005 to address the need for a feature that could distinguish intricate details in images, especially under variations in lighting, pose, and scale.

The main advantage of HOG is its ability to capture edge or gradient structures that are highly characteristic of local shapes [4]. HOG operates by dividing an image into small connected regions called cells and then calculating the histogram of gradient directions within each cell [19]. Essentially, HOG relies on the distribution of gradient orientations within an image [20]. To determine the results of HOG feature extraction, several formulas can be used, as shown in Equations 9 and 10.

$$G = \sqrt{Gx^2 + Gy^2} \quad (9)$$

$$\theta = arctan\frac{Gy}{Gx} \quad (10)$$

G = gradient magnitude
θ = gradient orientation

The Structural Similarity Index Method (SSIM) is a quantitative measure employed to assess the likeness between two images and is thought to be associated with the perceptual quality of the Human Visual System (HVS). The SSIM model takes into account three factors: loss of correlation, luminance distortion, and contrast distortion [21]. The SSIM value is a numerical measure that varies from 0 to 1. A value of 0 signifies no connection between the compared images, while a value of 1 signifies complete resemblance.

SSIM is a perceptual model that takes into account the perception of changes in structural information while considering picture degradation. This approach works in conjunction with other significant perception-related factors, such as luminance masking and contrast masking. The phrase "structural information" highlights the interdependence of pixels or their spatial adjacency. The term "dependent pixels" pertains to significant details on visual objects within the image domain. Luminance masking pertains to the portions of the distortion that are less discernible toward the margins of the image. Contrast masking, however, pertains to aberrations that are less conspicuous in the texture of the image. The SSIM metric quantifies the perceived quality of images and videos by assessing the similarity between two images: the original and the synthesized [22]. The SSIM score is determined using many formulas, including those presented in Equations 11 and 14.

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \quad (11)$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \quad (12)$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \quad (13)$$

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_x\sigma_y + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (14)$$

Figure 2 depicts a schematic representation of the planned research in the form of a block diagram. This study consists of two distinct parts. The first phase involves generating images using three different techniques: Vanilla GAN, StyleGAN, and DCGAN. The second phase focuses on extracting features related to color, texture, and HOG.

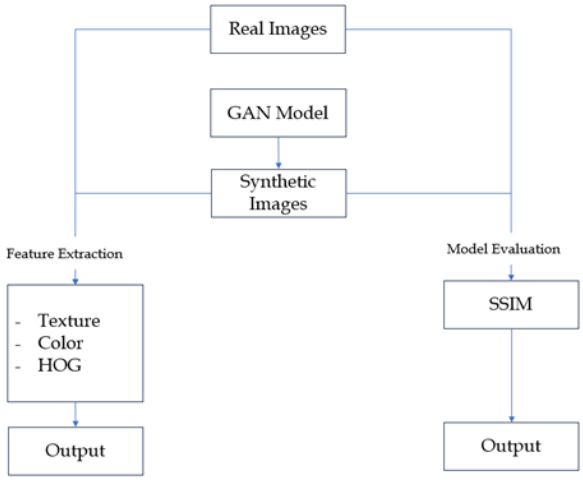

Figure 2. Research blok diagram

During the image generating step, the model design, depicted in Figure 2, comprises two components: feature learning and image generation using the learned model. The components of this system consist of the input layer, generator, discriminator, GAN model, training loop, saving generated images, and saving trained models for the training process. Afterwards, the steps include loading the generator model, generating synthetic images, and saving them for the image generation phase.

Feature learning

During the feature learning phase, the GAN model discerns and isolates significant characteristics from the image data utilized for training. The method commences with a preprocessing phase, in which photos are loaded, scaled, and normalized to guarantee uniformity in the utilized data. Subsequently, the image data undergoes a sequence of convolutional and activation layers specifically engineered to extract significant characteristics. The Vanilla GAN model employs a straightforward design consisting of convolutional layers followed by Leaky ReLU activations.

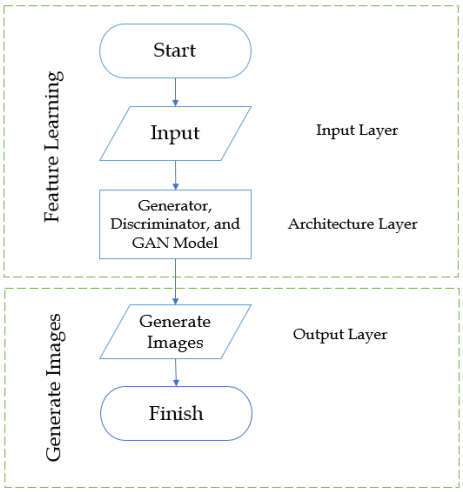

Figure 3. Proposed GAN model architecture

In contrast, the StyleGAN and DCGAN models have more intricate structures with deeper layers and variations in activations, including Leaky ReLU. These disparities

affect the model's ability to comprehend information and extract intricate characteristics from the photos. In general, the purpose of the feature learning stage is to produce superior feature representations from the image data.

During the feature learning step of Vanilla GAN, the image data is loaded and processed using a straightforward normalization technique that scales the pixel values between 0 and 1. The generator model, depicted in Figure 4, initiates with a Dense layer to convert the latent vector into a tensor with dimensions 75x100x128. The Reshape layer transforms the tensor into a multi-dimensional tensor of dimensions (75, 100, 128). The multi-dimensional tensor is upsampled using Conv2DTranspose layers until the image dimensions reach the required output of (600,800,3).

The design of the Vanilla GAN discriminator is illustrated in Figure 4, which utilizes convolutional layers with Leaky ReLU activation. The initial Conv2D layer does feature extraction using 64 feature maps, which are subsequently augmented to 128 feature maps in the following layer. The multi-dimensional tensor obtained from feature extraction is subsequently transformed into a vector by flattening it. This vector is then used as the input for a Dense layer that consists of a single neural network. Next, the resulting output is transformed into a probability value ranging from 0 to 1 by using a sigmoid activation function.

The generator architecture of DCGAN has a layer arrangement that closely resembles that of the Vanilla GAN. The process begins by transforming the latent vector into a tensor with one dimension, and then further converting it into a tensor with several dimensions using the Reshape layer. This design incorporates four Conv2DTranspose layers, with the inclusion of Batch Normalization at each upsampling level to enhance the efficiency of training. The initial upsampling stage employs 256 filters, contrasting with the Vanilla GAN's utilization of 128 filters.

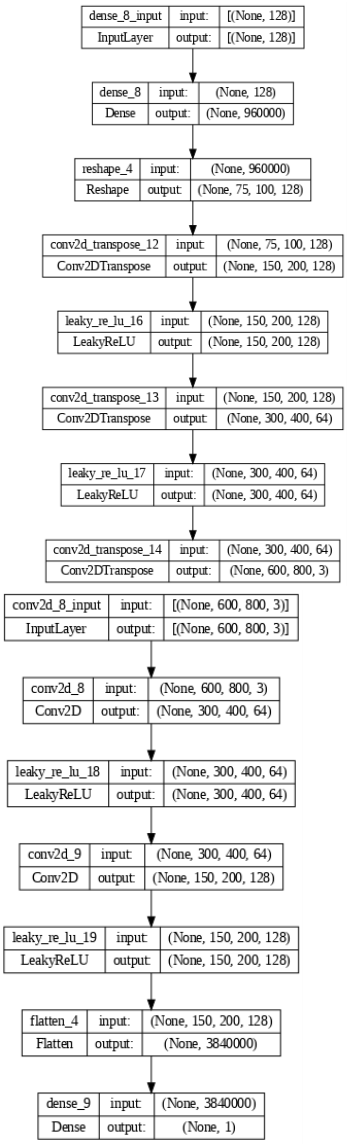

Figure 4. Architecture of the Generator and Discriminator in Vanilla GAN

The discriminator architecture of the DCGAN model, consisting of three Conv2D layers for the convolution stages with feature maps of 64, 128, and 256. The activation function utilized is Leaky ReLU with a parameter value of 0.2. Once the convolution stages are finished, the output is converted from a tensor to a vector so that it may be processed by the Dense layer. This processing results in an output in the form of probabilities, specifically 0 and 1, using the Sigmoid activation function.

During the feature learning stage of StyleGAN, the model utilizes a more intricate architecture with increased depth and employs normalizing techniques such as Batch normalizing to accelerate the training process. Style modulation is employed to regulate the styles at different levels of intricacy in the produced images. The process begins by converting the latent vector into a tensor to allow for noise injection and style modulation. These tensors are then multiplied together to create a modulated tensor, which serves as the input for the upsampling stage. This model employs three convolutional transpose layers, with

the inclusion of Batch Normalization in each Conv2DTranspose layer.

The discriminator architecture of the StyleGAN is more intricate than Vanilla GAN and DCGAN, as it consists of four convolutional layers with 64, 128, 256, and 512 feature maps respectively. Due to the abundance of feature maps, the discriminator has sufficient capacity to extract features from the input.

Generate images

During the Generate Images phase of the three implemented GAN models, the generator that has been trained is utilized to produce novel synthetic images. In the Vanilla GAN framework, this procedure entails using the `predict` function on the generator model and supplying randomly produced noise vectors as input. The generated artificial photos are thereafter stored in a designated directory.

In StyleGAN, the generator not only takes randomly produced noise vectors as input, but also applies a truncation approach to the noise vector before processing it. Afterwards, the synthetic photos that are produced are stored in the designated directory.

In DCGAN, the procedure is almost same to Vanilla GAN. Randomly produced noise vectors are inputted directly into the generator model to generate artificial images. The outcomes are thereafter stored in the designated directory. Therefore, the Generate Images stage in these three GAN models has the identical objective: to produce artificial images using the acquired patterns from the training process.

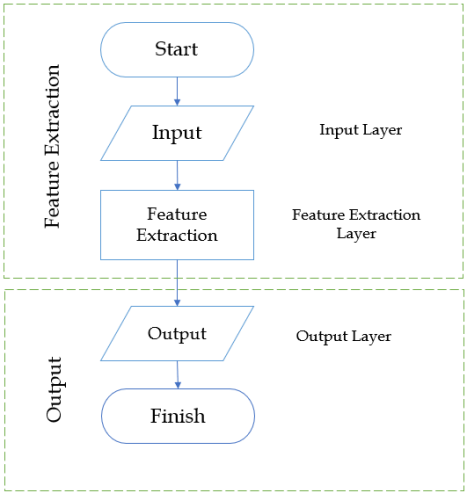

Figure 5. Proposed feature extraction model architecture

The second step involves the phase of feature extraction, which includes the construction of a model as depicted in Figure 5. This stage comprises the extraction of features and the generation of output. The feature extraction component consists of the input layer and the process of extracting features. The output section generates feature extraction results from each model.

Feature extraction

The architecture design of the feature extraction model includes an initial input layer that is responsible for preprocessing the images from the dataset prior to being inputted into the model. Initially, the photos are processed and transformed into the suitable format with OpenCV. Subsequently, the dimensions of the photographs are modified to match the expected size by the model through the utilization of resizing techniques. Subsequently, the pixel values in the photos are standardized to the interval [0, 1] in order to assist the training process.

Afterwards, the images are forwarded to the feature extraction layer, where significant features are extracted. Every method of feature extraction possesses a distinct layer architecture. For example, when extracting texture, the GLCM (Grey-Level Co-occurrence Matrix) is calculated using the Greycomatrix function from the skimage library. The GLCM algorithm generates features such as contrast, dissimilarity, homogeneity, energy, and correlation. In addition, to extract color information, a color histogram is created from the histogram layer using the histogram function from the OpenCV package.

This color histogram represents the distribution of pixel frequencies in the RGB color space. Meanwhile, the HOG Descriptor Layer utilizes the OpenCV library to extract the HOG descriptor for HOG feature extraction. This layer utilizes the Histogram of Oriented Gradients (HOG) method for extracting features. HOG examines the distribution of intensity gradients within blocks of an image. Every layer architecture requires the adjustment of specific parameters, including the number of histogram bins, HOG cell size, and other factors.

Output

Within the Output part, every feature extraction model showcases the extracted features and conducts an analysis of the disparities between datasets, using the retrieved features as a basis. In the Texture Feature Extraction model (GLCM), a feature vector that represents the texture properties of the images is recorded. Additionally, a histogram of the texture features is generated to visually illustrate the distribution of these feature values. Subsequently, the mean texture attributes are calculated for each dataset, and the disparity in mean texture attributes between the two datasets is quantified to emphasize the distinctions in texture characteristics.

Additionally, in the Color Feature Extraction model (HSV Histogram), histograms of colors are extracted for each HSV color channel from the photos. These histogram findings are then shown to visually represent the distribution of colors within the dataset. The color histograms are analyzed to identify the highest values, which represent the dominating colors in the photos. Moreover, the disparities in the maximum values of the color histograms of the two datasets are determined, and the % difference is generated to illustrate the variances in

color between the datasets.

Additionally, the HOG Feature Extraction model presents the recovered HOG features from the photos in conjunction with the original images, allowing for the visualization of the extracted texture patterns. The matrix values of the Histogram of Oriented Gradient (HOG) features are also displayed to offer additional understanding of the feature representation.

In addition, the HOG feature values from both datasets are calculated and utilized to determine the cosine similarity between the HOG feature matrices of the two datasets. The cosine similarity values are utilized to quantify the percentage disparity between the datasets, based on the recovered HOG features, hence emphasizing the variations in texture patterns between the two datasets. Thus, in the Output phase, every feature extraction model provides pertinent information and crucial analysis of disparities to comprehend the properties of the dataset.

III. RESULT AND DISCUSSION

GAN model loss result

The GAN modeling was performed for 20, 40, 60, 80, and 100 epochs, with the goal of optimizing the performance of each model in generating images in the minority class below 9999.

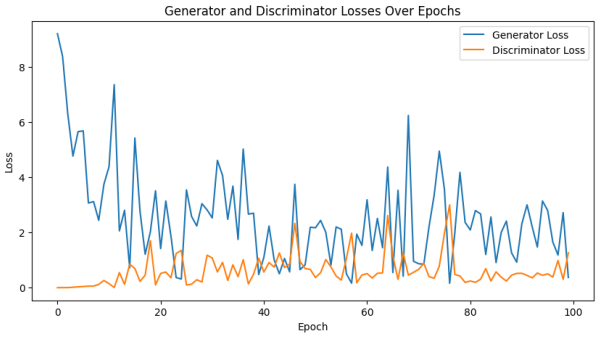

Figure 6. Plot of Generator and Discriminator Loss for Vanilla GAN Model over 100 Epochs

The Vanilla GAN model exhibits diverse patterns in the performance of both the Generator and Discriminator during 100 epochs, as depicted in Figure 6. At the beginning, the Generator shows a substantial reduction in loss, dropping from roughly 9.19 in the first epoch to around 3.74 by the 10th epoch. This suggests an enhancement in the quality of the generated images. Nevertheless, the Discriminator continuously displays a minimal and stable loss, indicating its proficient capability to accurately differentiate between authentic and counterfeit photos. Generator and Discriminator loss exhibit changes throughout training, with certain instances displaying abrupt variations, such as at epoch 47. These fluctuations could potentially signify problems such as mode collapse or training instability.

Loss fluctuations can arise due to issues such as inadequate learning rates, insufficient data variance, or imbalanced strength between the Generator and Discriminator during training.

By the end of the training process, the Vanilla GAN model demonstrates that the Generator has achieved a minimal loss of around 0.37, indicating that it has converged towards generating images of the desired quality. However, the Discriminator currently had a loss of approximately 1.26. Although there was some progress from its initial loss, there were still variations that suggested difficulties in reliably differentiating between real and false images.

Figure 7 displays the loss plot of the generator and discriminator components of the DCGAN model across 100 epochs. The loss reported throughout the 100 epochs of DCGAN training exhibits a noteworthy trend.

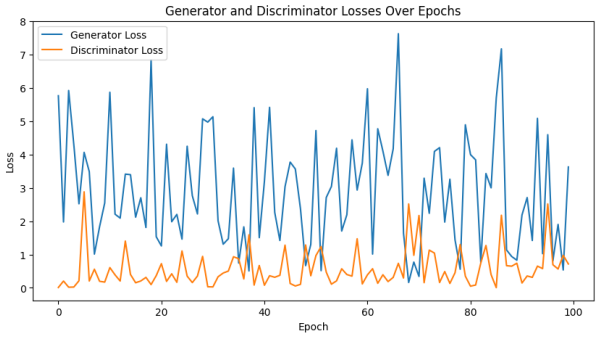

Figure 7. Plot of Generator and Discriminator Loss for DCGAN Model over 100 Epochs

During the initial phase of training, the generator exhibited a significant loss of around 5.76 in the first epoch. However, this loss rapidly diminished as subsequent epochs were completed. This suggests that the generator began to generate images that closely resembled the actual data as its capacity to alter the noise input enhanced.

Nevertheless, there were instances where there were sudden increases or decreases in the loss of both the generator and discriminator. During the 6th epoch, the generator encountered a rapid increase in loss, reaching a value of 4.06. This suggests the occurrence of a mode collapse, where the generator fails to generate diverse images that can deceive the discriminator.

In the last epoch, the generator had a loss of 3.62 while the discriminator had a loss of 0.72. Overall, this suggests that the generator has acquired the ability to generate superior images in comparison to the initial stages of training, while there is still potential for further enhancement. Conversely, the discriminator has effectively acquired the ability to differentiate between authentic and synthesized images, as evidenced by its minimal loss, which indicates its proficiency in this task.

DCGAN tends to show greater fluctuations in loss during training. This can be attributed to the higher complexity of the DCGAN architecture, which requires more time and data to stabilize during training. Vanilla GAN, with its simpler architecture, tends to have more stable and controlled loss.

Figure 8 displays the loss plot for the generator and discriminator components of the StyleGAN model across 100 epochs. An intriguing trend emerges from the documented losses across the 100 epochs of StyleGAN

training. During the first stage of the training, the generator exhibits a significant loss of approximately 7.23 in the first epoch. However, this loss gradually diminishes as the subsequent epochs unfold. This suggests that the generator is beginning to generate images that have a stronger resemblance to the original data as its capacity to modify the noise input enhances.

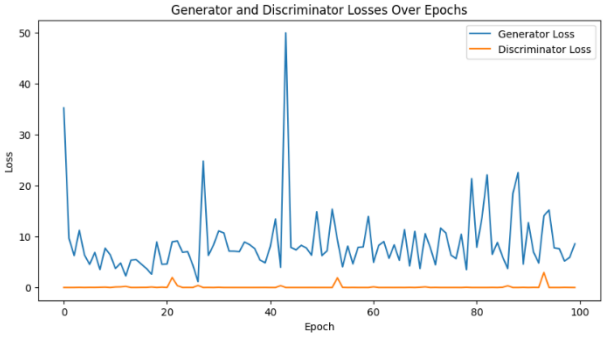

Figure 8. Plot of Generator and Discriminator Loss for StyleGAN Model over 100 Epochs

Nevertheless, there are intermittent instances where the generator and discriminator losses experience abrupt increases or decreases. At epoch 15, the generator encounters a sharp increase in loss, reaching 6.80, suggesting a potential occurrence of mode collapse or a situation where the generator fails to generate enough diverse images to deceive the discriminator. The variations seen also indicate the difficulties encountered in training the StyleGAN model, particularly when using a dataset consisting of photos with limited diversity.

In the last epoch, the generator had a loss of 2.45 while the discriminator had a loss of 0.65. Overall, this suggests that the generator has acquired the ability to generate superior images in comparison to the initial stages of training, while there is still potential for further enhancement. The discriminator, however, has achieved a high level of proficiency in distinguishing between authentic and synthesized images, as seen by a loss value close to 0.5, which signifies its effectiveness in this objective.

StyleGAN tends to show more significant fluctuations in loss during training compared to DCGAN. This can be attributed to the higher complexity of the StyleGAN architecture and the use of techniques such as style modulation and noise injection, which require more time and data to stabilize during training. DCGAN, with its simpler architecture, tends to have more stable and controlled loss. However, StyleGAN has the potential to generate higher quality images with greater variation if the challenges in training stability can be overcome.

The StyleGAN model encounters challenges in generalizing and producing images with adequate variation when working with a dataset that exhibits low variability. Mode collapse can occur when the generator consistently produces images that are highly similar, resulting in a decrease in the quality and variety of the created images.

Table 1 displays the scores obtained from the Structural Similarity Index Method (SSIM). Figure 9 illustrates the graph of SSIM scores for each GAN model, which were calculated using Equations 11-14. The GAN model exhibits a progressive rise in SSIM scores from epoch 20 to 80, reaching a maximum of 0.5261. Nevertheless, by epoch 100, the Structural Similarity Index (SSIM) drops to 0.4986, suggesting the presence of overfitting or fluctuations in the generated image's quality. The DCGAN model consistently maintains steady scores throughout all epochs, exhibiting somewhat higher values in comparison to the Vanilla GAN. The epoch 60 yielded the maximum score of 0.5333, indicating a marginal enhancement in the quality of the created photos. DCGAN has superior consistency in comparison to Vanilla GAN.

SSIM score

Table 1. SSIM score

Model	Epoch	Score SSIM
Vanilla GAN	20	0.4908
	40	0.5007
	60	0.5158
	80	0.5261
	100	0.4986
DCGAN	20	0.5280
	40	0.5281
	60	0.5333
	80	0.5281
	100	0.5319
StyleGAN	20	0.5098
	40	0.5202
	60	0.5177
	80	0.4674
	100	0.5221

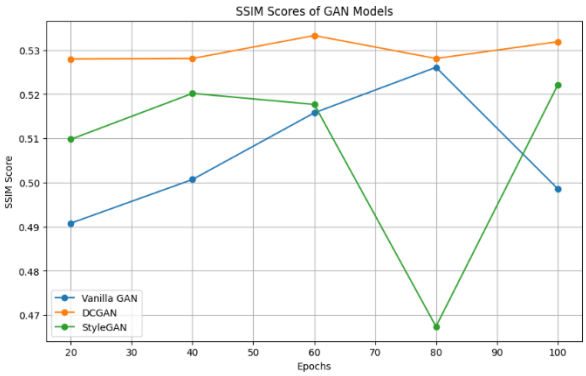

Figure 9. Plot of SSIM Scores for GAN Models

The StyleGAN model exhibits an upward trend in SSIM scores from epoch 20 to 40, followed by a modest decline at epoch 60. Epoch 80 has a notable decrease, with a score of 0.4674, suggesting possible problems in training or image quality during that epoch. At epoch 100, the SSIM score increases to 0.5221, indicating a restoration in image quality.

Texture

Table 2. Results of texture feature extraction

Model	Epoch	Difference Under 9999 (%)			Difference Over 9999 (%)			Relative Similarity
		C	H	E	C	H	E	(%)
Vanilla GAN	20	10,73	35,46	16,32	37,95	21,88	0,26	51
	40	42,42	6,29	1,36	50,59	4,15	3,59	46
	60	10,67	12,84	2,42	20,36	1,28	1,23	53
	80	11,46	23,84	5,02	33,18	26,45	3,83	39
	100	11,79	12,35	2,28	24,94	23,52	0,01	35
DCGAN	20	73,93	50,88	11,85	82,42	53,04	14,32	48
	40	77,21	26,59	9,87	80,9	54,44	16,32	43
	60	76,67	11,57	7,95	78,85	43,74	14,66	41
	80	77,82	56,22	13,95	85,28	54,6	16,25	49
	100	80,37	49,11	14,37	85,8	59,27	18,73	47
StyleGAN	20	72,54	12,16	11,64	74,32	1,91	11,82	52
	40	80,37	71,42	22,26	82,02	75,91	22,67	49
	60	87,28	44,02	21,69	90,89	44,06	24,82	49
	80	67,71	5,65	10,57	66,94	13,6	11,6	48
	100	80,81	5,13	7,32	84,28	49,09	18,15	38

The texture feature extraction findings in Table 2, derived using Equations 6-8, which measure contrast, homogeneity, and entropy, indicate that Vanilla GAN achieves a relative similarity score of 53% at epoch 60, indicating its best performance. At this particular point in time, the images produced by Vanilla GAN bear the closest resemblance to the class labeled as 9999.

The Vanilla GAN achieves its lowest relative similarity score at epoch 100, with a value of 35%. Currently, the generated photos exhibit the least resemblance to the class below 9999. The observed outcome is consistent with the SSIM analysis, which indicates that the score varies but exhibits an overall increase and subsequent decrease by epoch 100. Although there were variations in the relative similarity scores, epoch 60 exhibited the highest level of performance.

The optimal epoch for texture feature extraction in DCGAN is epoch 80, yielding a relative similarity score of 49%. The current outcome is marginally greater than the one obtained at epoch 20, which reached a 48% success rate. These statistics are still lower than the maximum results attained by the Vanilla GAN model.

DCGAN has superior stability in comparison to Vanilla GAN and StyleGAN, while achieving a maximum score of just 49% and a minimum score of 41% at epoch 60. The StyleGAN model at Epoch 20 excels at extracting texture features, achieving a relative similarity of 52%. Nevertheless, the performance of StyleGAN exhibits a deterioration, as the results consistently diminish from epoch 20 to epoch 100.

Color

Table 3. Results of color feature extraction

Model	Epoch	Difference Under 9999 (%)			Difference Over 9999 (%)			Relative Similarity
		H	S	V	H	S	V	(%)
Vanilla GAN	20	11,16	35,58	23,43	8,97	34,57	23,66	51
	40	4,79	32,75	34,24	2,76	31,85	29,49	53
	60	44,75	33,66	27,89	24,52	7,66	9,91	72
	80	2,93	19,1	37,52	20,91	9,2	5,43	63
	100	28,71	5,63	23,27	25,76	2,37	24,51	52
DCGAN	20	3,33	82,88	94,77	16,19	82,7	55,44	54
	40	2,33	72,99	77,75	0,16	81,88	51,93	53
	60	0,19	80,65	54,67	9,59	86,53	37,87	50
	80	4,46	76,53	64,11	7,08	81,99	56,41	50
	100	17,23	75,32	102,3	7,46	87,53	32,18	61
StyleGAN	20	11,88	76,21	62,66	26,13	85,97	36,22	50
	40	14,77	86,64	49,65	17,98	91,7	24,75	53
	60	9,96	76,63	67,15	16,11	83,31	42,07	52
	80	0,27	68,49	70,77	8,25	78,88	37,57	53
	100	56,64	80,41	63,92	63,77	80,72	76,65	48

Table 3 displays the outcomes of extracting color features from the three GAN models, as derived using Equations 2-5. The ideal number of epochs for Vanilla GAN in color feature extraction is 60, resulting in a relative similarity of 72%. The relative similarity value increases steadily from epoch 20 to epoch 60, but then declines between epochs 80 and 100.

According to the findings, the model achieved a minimum relative similarity value of 51% at epoch 20. The current result is only 1 percentage point lower than the result at epoch 100, which shown a decrease beginning at epoch 80. The degree of similarity at epoch 100 is 52%. The optimal epoch for the DCGAN model is 100, achieving a relative similarity value of 61%. The value reached its peak in epoch 100, following a fall in performance from epoch 20 to 80.

The optimal value in this model remains lower than the comparative similarity value of Vanilla GAN. The epoch at which the highest relative similarity value is observed in StyleGAN is epoch 40, with a value of 53%. StyleGAN ranks last among the three models in terms of relative similarity for color characteristics. This model has a very consistent behavior, with its peak similarity value reaching just 53% and its lowest similarity value occurring at epoch 100, with a value of 48%.

The results of Histogram of Oriented Gradient (HOG) feature extraction for each GAN model derived using Equations 9 and 10 are displayed in Table 4.4. The Vanilla GAN model produces surprising results, with a relatively high similarity score of up to 90% at epoch 80. The accuracy reached 75% at epoch 20, but experienced a considerable decline to only 10% at epoch 40. Subsequently, there was a fluctuation in performance, with the similarity reaching 86% at epoch 100, which was lower than the previous 90%.

HOG

Based on the findings shown in Table 4, it is evident that the Vanilla GAN model encountered instability throughout the training process.

Table 4. Results of HOG feature extraction

Model	Epoch	Difference Under 9999(%)	Difference Over (%)	Relative Similarity (%)
Vanilla GAN	20	0,3	0,1	75
	40	0,2	1,72	10
	60	6,23	5,97	51
	80	5,94	0,66	90
	100	1,26	0,2	86
DCGAN	20	16,55	17,04	49
	40	14,02	14,63	49
	60	10,61	9,79	52
	80	14,77	14,74	50
	100	13,96	13,83	50
StyleGAN	20	0,81	1,38	37
	40	11,69	13,91	46
	60	4,72	4,37	52
	80	4,18	3,48	55
	100	9,6	7,55	56

The DCGAN model exhibits stability across a spectrum of relative similarity values ranging from 49% to 52%. The similarity between epochs 20 and 60 shows a significant increase, reaching a peak relative similarity of 52%. Nevertheless, epochs 80 and 100 observed a marginal decline to 50%.

The DCGAN model continues to exhibit relatively low similarity results compared to the Vanilla GAN model, ranging from 75% to 90%. The StyleGAN model exhibits a gradual and consistent trend in the outcomes. The initial level of similarity is 37% at epoch 20 and gradually rises to 56% by epoch 100. Epoch 100 exhibits the highest performance, achieving a relative similarity of 56%. Although there is a small variation in results, this still represents the greatest relative similarity across all epochs for StyleGAN. Based on these findings, StyleGAN exhibits a positive trend in performance, however its relative resemblance to Vanilla GAN is still distant.

IV. CONCLUSION

Conclusion

The Vanilla GAN model at epoch 60 has been determined to be the most effective at generating images with a base of 9999, based on feature evaluation, SSIM scores, synthesis image quality, and loss pattern. The architectural simplicity of this model allows it to effectively capture minimal variations in the dataset, giving it an advantage over more intricate designs such as DCGAN and StyleGAN. Additional optimization and architectural tweaks could further increase results, particularly for datasets with minimal variability, as utilized in this work.

Suggestion

Future studies should prioritize increasing image variation to enhance the model's capacity to collect diverse and pertinent information, based on the completed research. To improve VanillaGAN, one can better its performance by improving the architecture, increasing the number of epochs and batch size, and implementing regularization techniques to address overfitting. To strengthen the power of the DCGAN model to capture more complicated aspects, one can incorporate additional layers and utilize more advanced architectural techniques. StyleGAN necessitates targeted refinements for color characteristics by incorporating additional layers that specifically address color aspects and pixel values inside the photos.

ACKNOWLEDGMENT

I would like to express my sincere gratitude to Professor Asto, for his invaluable guidance and support throughout this research. His insightful feedback and encouragement were instrumental in shaping the direction of my study.

REFERENCES

[1] X. Zhou, Y. Hu, J. Wu, W. Liang, J. Ma, and Q. Jin, "Distribution Bias Aware Collaborative Generative Adversarial Network for Imbalanced Deep Learning in Industrial IoT," *IEEE Trans. Ind. Informatics*, vol. 19, no. 1, pp. 570–580, 2023.

[2] H. Achicanoy, D. Chaves, and M. Trujillo, "StyleGANs and transfer learning for generating synthetic images in industrial applications," *Symmetry (Basel).*, vol. 13, no. 8, pp. 1–16, 2021.

[3] L. Alwi, A. T. Hermawan, and Y. Kristian, "Identifikasi Biji-Bijian Berdasarkan Ekstraksi Fitur Warna, Bentuk dan Tekstur Menggunakan Random Forest," *J. Intell. Syst. Comput.*, vol. 1, no. 2, pp. 92–98, 2019.

[4] T. A. Mutiara and Q. N. Azizah, "Klasifikasi Tumor Otak Menggunakan Ekstraksi Fitur HOG dan Support Vector Machine," *J. Infortech*, vol. 4, no. 1, pp. 45–50, 2022.

[5] A. D. I. M. S. P. M. B. Saputra, "Pengaruh Kondisi Cuaca Penerbangan (Aviation Weather) Pengaruh Kondisi Cuaca Penerbangan (Aviation Weather) Terhadap Beban Kerja," *J. Transp.*, vol. 15, no. November, pp. 159–168, 2015.

[6] A. Islam, S. B. Belhaouari, A. U. Rehman, and H. Bensmail, "KNNOR: An oversampling technique for imbalanced datasets[Formula presented]," *Appl. Soft Comput.*, vol. 115, 2022.

[7] A. Indrawati, "Penerapan Teknik Kombinasi Oversampling Dan Undersampling Untuk Mengatasi Permasalahan Imbalanced Dataset," *JIKO (Jurnal Inform. dan Komputer)*, vol. 4, no. 1, pp. 38–43, 2021.

[8] M. Wulandari, "Index quality assesment citra terinterpolasi (SSIM dan FSIM)," *J. Terap. Teknol. Inf.*, vol. 1, no. 1, pp. 11–20, 2017.

[9] K. Akbar and M. Hayaty, "Data Balancing untuk Mengatasi Imbalance Dataset pada Prediksi Produksi Padi," *J. Ilm. Intech Inf. Technol. J. UMUS*, vol. 2, no. 02, 2020.

[10] Q. Dong, S. Gong, and X. Zhu, "Imbalanced Deep Learning by Minority Class Incremental Rectification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 6, pp. 1367–1381, 2019.

[11] S. Jozdani, D. Chen, D. Pouliot, and B. Alan Johnson, "A review and meta-analysis of generative adversarial networks and their applications in remote sensing," *Int. J. Appl. Earth Obs. Geoinf.*, vol. 108, no. March, 2022.

[12] S. Sabnam and S. Rajagopal, "Application of generative adversarial networks in image, face reconstruction and medical imaging: challenges and the current progress," *Comput. Methods Biomech. Biomed. Eng. Imaging Vis.*, vol. 12, no. 1, 2024.

[13] V. Sampath, I. Maurtua, J. J. Aguilar Martín, and A. Gutierrez, *A survey on generative adversarial networks for imbalance problems in computer vision tasks*, vol. 8, no. 1. Springer International Publishing, 2021.

[14] P. Salehi, A. Chalechale, and M. Taghizadeh, "Generative Adversarial Networks (GANs): An Overview of Theoretical Model, Evaluation Metrics, and Recent Developments," 2020.

[15] I. G. R. A. Sugiartha, M. Sudarma, and I. M. O. Widyantara, "Ekstraksi Fitur Warna, Tekstur dan Bentuk untuk Clustered-Based Retrieval of Images (CLUE)," *Maj. Ilm. Teknol. Elektro*, vol. 16, no. 1, p. 85, 2016.

[16] M. Fahmi Wibawa, M. A. Rahman, and A. W. Widodo, "Penerapan Ruang Warna HSV dan Ekstraksi Fitur Tekstur Local Binary Pattern untuk Tingkat Kematangan Sangrai Biji Kopi," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 5, no. 7, pp. 2819–2825, 2021.

[17] S. A. R. Srg, Irhamna, M. F. Aldi, M. Ramadhan, and N. L. Siregar, "Ekstraksi Fitur Citra Berdasarkan Tekstur Dengan Glcm (Gray Level Co-Occurrence)," *JUTISAL (Jurnal Tek. Inform. Komput. Universal)*, vol. 3, no. 1, pp. 44–51, 2023.

[18] W. I. Praseptiyana, A. W. Widodo, and M. A. Rahman, "Pemanfaatan Ciri Gray Level Co-Occurrence Matrix (GLCM) untuk Deteksi Melasma pada Citra Wajah," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 3, no. 11, pp. 10402–10409, 2019.

[19] Normah, B. Rifai, S. Vambudi, and R. Maulana, "Analisa Sentimen Perkembangan Vtuber Dengan Metode Support Vector Machine Berbasis SMOTE," *J. Tek. Komput. AMIK BSI*, vol. 8, no. 2, pp. 174–180, 2022.

[20] T. A. S. Srinivas, K. Khan, K. K. Reddy, and G. S. Chand, "Artistry in Detail : Mastering HOG Feature Extraction Artistry in Detail : Mastering HOG Feature Extraction," no. December, 2023.

[21] M. Wulandari, "Pengukuran Ssim Dan Analisis Kinerja Metode Interpolasi Untuk Peningkatan Kualitas Citra Digital," *J. Muara Sains, Teknol. Kedokt. dan Ilmu Kesehat.*, vol. 1, no. 1, pp. 184–195, 2017.

[22] U. Sara, M. Akter, and M. S. Uddin, "Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study," *J. Comput. Commun.*, vol. 07, no. 03, pp. 8–18, 2019.