International Journal on Advanced Science Engineering Information Technology

Vol.13 (2023) No. 1
ISSN: 2088-5334

Generation of *Batik* Patterns Using Generative Adversarial Network with Content Loss Weighting

Agus Eko Minarno [a], Toton Dwi Antoko [a], Yufis Azhar [a,*]

[a] *Department of Informatics, University of Muhammadiyah Malang, Jl. Raya Tlogomas No.246, Malang, 65145, Indonesia*
*Corresponding author: *yufis@umm.ac.id*

Abstract— **Every craftsman who draws *Batik* can also not necessarily draw various types of *Batik*. On the other hand, it takes a long time ranging from weeks to months, to make *Batik*. Image generation is regarded as an essential part of the field of computer vision. One of the popular methods includes the Generative Adversarial Network, commonly implemented to generate a new data set from an existing one. One model of the Generative Adversarial Network is BatikGAN SL generating *batik* images by inserting the two *Batik* patterns to produce a new *Batik* image. Currently, the generated *Batik* image does not maintain the input of the *Batik* pattern. Therefore, this study proposes a GAN model of BatikGAN SL, with the addition of a content loss function using hyperparameters to weight the content loss function. The content loss function is added from the Neural Transfer Style method. Previously, the style loss function in this method has been implemented in BatikGAN SL, and the dataset consists of *Batik* patches (326 images) and real *Batik* (163 images). This paper compares the BatikGAN SL model from previous studies with the BatikGAN SL model by implementing hyperparameters on the content loss function. The evaluation is conducted with FID, containing FID Local and FID Global. The results obtained in this study include a collection of *Batik* images, test evaluation value of 42 on FID Global and 16 on FID Local. These results are obtained by implementing the content loss function with a weight value of 1.**

Keywords— **Generative adversarial network; batik GAN SL; image generation; *Batik*.**

Manuscript received 1 Sep. 2021; revised 19 Dec. 2021; accepted 12 Jul. 2022. Date of publication 28 Feb. 2023.

I. Introduction

Identity refers to the description of a nation in a broad scope and a human being in a small scope. The identity of a country is manifested in various forms, such as through state symbols, heritage flags, national anthems, and culture. One of the Indonesian people's cultures is typically represented by various types of *Batik* found in various regions in Indonesia [1]. *Batik* becomes the visual evidence for the intellectual property of Indonesian culture, depicting distinctive shape, color, style, and texture of traditional fabrics, influenced by religions and cultures, such as Java, Islam, Hinduism, the Netherlands, Japan, and China. In addition, UNESCO, on October 2, 2009, acknowledged that *batik* cloth is a cultural heritage of Indonesia [2].

Batik is generally created by craftsmen using specific techniques, including dyeing, printing, stamping, and dabbing. Each technique has a different difficulty level, thereby depicting imperfection in drawing *Batik* pattern [3], [4]. Every craftsman who draws *Batik* can also not necessarily draw various types of *Batik*. On the other hand, it takes a long time ranging from weeks to months to make *Batik* [5]. However, if the craftsman already has a motif or example or template of a new type of *Batik*, it will significantly simplify the manufacturing process. If a computer could generate new *batik* patterns by inserting the previous and new *batik* patterns, this would assist the craftsman in designing *batik* [6], preventing the craftsman from sketching a new type of *batik* pattern. However, most craftsmen remain possible to experiment with their designs to create new *batik* pattern [7].

Studies on *Batik* have been previously conducted by numerous researchers, especially in the field of image generation. One of the studies on Batik was conducted by Tian [8], produced *Batik* automatically by simply entering a few parameters using the fractal geometry method. In 2020, research on *Batik* was also conducted by Purba Daru Kusuma to produce Madura-style *Batik*, implemented on the website by utilizing the sine wave model method [9].

In addition, developments in deep learning regarding image generation, especially through Neural Style Transfer, are currently being conducted for research. Neural Style Transfer was initially proposed by Gatys et al. [10]. Similarly, research on Neural Style Transfer was conducted by Zhou et al. [11] using the VGG19 pre-trained model architecture on the ImageNet dataset, requiring 5 hours 32 minutes with an image size of 256 x 256 for training the model. Further, another study conducted by Irawan and Widjaja [12] has generated *Batik* patterns by transferring natural styles through Neural Transfer Style using VGG-16 architecture. The results of this study succeeded in transferring the natural image style to the *batik* image. In 2020, research by Atarsaikhan et al. [13] accomplished the transfer of the image styles to the image-detected shapes by using the VGG pre-trained model architecture. Research by Kwon et al. [14] similarly generated image style transfer to produce captcha images that would resist machine recognition while maintaining their ability to be recognized by humans using the VGG19 pre-trained model architecture.

In terms of image generation, in addition to implementing Neural Style Transfer, a deep learning algorithm is frequently used through the Generative Adversarial Network (GAN), focusing on shaping and generating completely new data[15], [16]. This algorithm was initially proposed by Goodfellow et al. [17]. Thus, numerous studies using GAN have been carried out in recent years, especially in image generation.

One of the studies implementing GAN was conducted by Wang et al. [18] to improve the sharpness quality of blurry images by utilizing the Relativistic average Least Square GAN (RaLSGAN) method with an additional layer of attention. The results indicated that the SSIM and PSNR values were 0.968 and 27.291 in the NYU2 Synthesis dataset. Kupyn et al. [19] similarly researched GAN to improve blurry images' sharpness by implementing DeblurGAN by comparing additional pre-trained models of Inception, MobileNet, and MobileNetDSC. The results reported that the PSNR and SSIM values were 29.55 and 0.934 in the pre-trained Inception model. In 2020, research by Yuan et al. [20] reconstructed MRI images by using Self Attention and Relative Average on the GAN discriminator, generating the PSNR and SSIM values of 45.75 and 0.99 in an average sample of 30%. Another survey by Jin et al. [21] proposed E-WACGAN consisting of WGAN-GP and ACGAN to reproduce image data samples on telecommunications networks.

Another study applying GAN has also been conducted on *batik* images by Abdurrahman et al. [22] using the DCGAN architecture. The proposed model generated a *Batik* pattern that is almost similar to the dataset. Research by Huang et al. [23] proposed a method to increase image resolution and capture details of cracks in the image using BatikDeblurGAN by changing the use of WGAN-GP to RaLSGAN, changing transpose convolution to resize convolution, adding Hybrid Dilatation Convolution (HDC) and adding several loss functions. The study's results reported that the best PSNR was in the L1 loss function with a value of 19.73 and SSIM with a value of 0.79. In 2020, Chu and Ko [24] also researched the generation of *Batik* patterns by entering the two pattern pieces. The proposed method consists of two stages: Patch Generator and Batik Generator. In Batik Generator, the three proposed architectural models included BatikGAN, BatikGAN_S, and BatikGAN_SL. The architecture was tested by qualitative and quantitative methods with survey results of 27%, 28%, and 45%, respectively, and quantitative test results with FID calculations of 168.42, 163.92, and 137.87, respectively.

A previous study by Chu and Ko [24] was conducted on *Batik* generation by comparing the three proposed architectures. The third proposed architecture was considered good quality based on the qualitative and quantitative tests described above. However, the produced batik images are still not controlled, resulting in the pattern of inserted *Batik* pieces not being generated properly due to local style loss and local discriminators generating the original image sample, and the use of weights to be more dominant than the input image.

Referring to the problems above, this study aims to improve model performance by maintaining the pattern or style of the input image. The difference between this research and previous research lies in the addition of a content loss calculation function by performing hyperparameters on the weight value of the loss or error calculation, which are advantageous to maintain the original image or input image [12].

The major contributions of this paper are listed below.

- We built an improved GAN model to generate a new *Batik* pattern using two patches.
- A substantial improvement in performance is observed when handcrafted GAN with content loss weighting features are used.
- The proposed method recorded the best model compared to state-of-the-art methods.

The remaining part of this paper is organized as follows. In section 2, we present an overview of the description of the dataset used in the study, followed by GAN architecture and the complete framework for the proposed algorithm. Section 3 gives details of the experiments performed, presents evaluation results, and provides a discussion of the results. Section 4 provides the conclusion of this paper.

II. Materials and Method

This chapter discusses each part of the stages, including the data used and the work process, starting from data preprocessing to model evaluation. This research design aims to describe the following steps better to be taken. An overview of the stages of the research design is illustrated in Fig. 1.

A. Dataset

The dataset used is not solely collected by the researchers of this study. The data used to refer to the *Batik* dataset as in the research of Yohanes Gultom et al., accessed on Github at the following link https://github.com/yohanesgultom/deep-learning-batik-classification [25]. This dataset was also utilized by previous studies [24], consisting of 645 batik images with various types of *Batik* patterns, including *Ceplok*, *Kawung*, slopes, *Nitif*, machetes, and mixed patterns.

B. Preprocessing Data

The dataset is obtained from the GitHub site, which contains *Batik* images consisting of various types of *Batik* with written and fractal *Batik* patterns. The fractal *Batik* pattern has a regular pattern that regularly repeats, while the written *Batik* pattern has an irregular random pattern according to the maker's will. Therefore, fractal *Batik* pattern is implemented in this study. The selected images include the 163 images from various types of *Batik* as illustrated in Fig. 2.

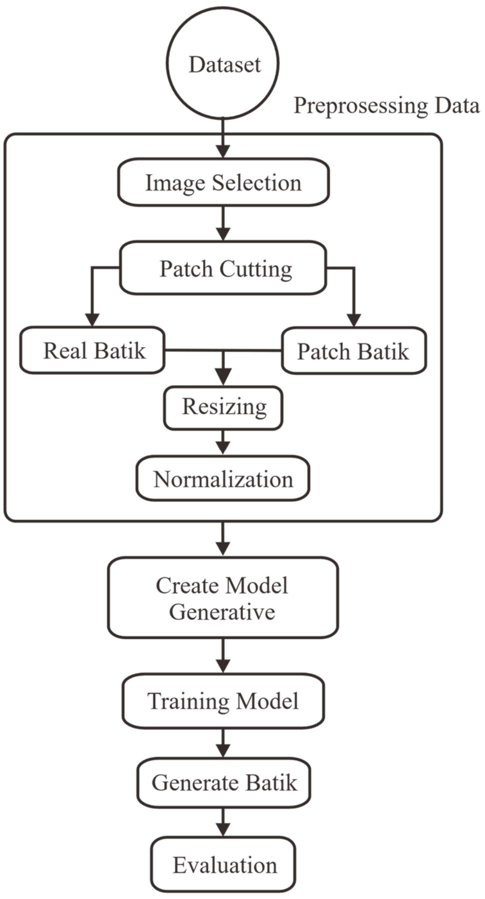

Fig. 1 The proposed research series

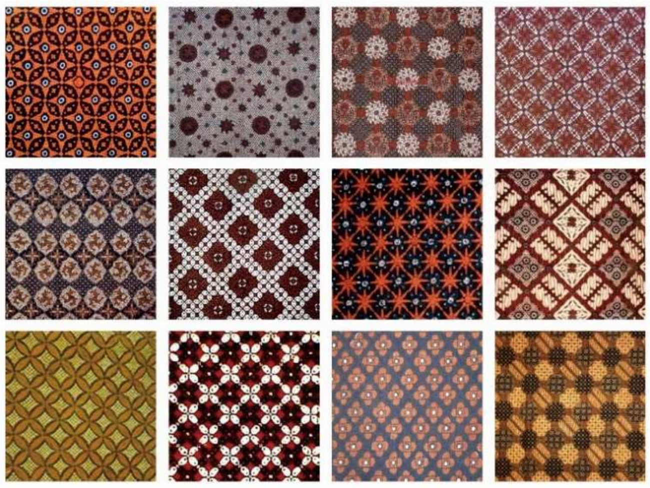

Fig. 2 Samples of selected batik images

The selection results of the *Batik* image include fractal *Batik* patterns, which are sequential and repeated. Therefore, each *Batik* image will be cropped to obtain the patch or pattern. The patch cutting is manually performed by cropping each *Batik* image generating the two patterns. A Batik image with more than two patterns will be taken randomly, as each batik pattern or patch will be in pairs. The pair of *Batik* patches would further be used as input into a model to perform the training process. Thus, if the pattern used is more or less than two, then the model will not be able to perform the training process. In addition, during the model training process, each *Batik* pattern should not be paired with patterns from other *Batik* images as the *Batik* pattern that is not a partner is only made when evaluating. The cropping results of *Batik* pattern or patch generate the 326 *Batik* patches. Some samples of patch cutting results are illustrated in Fig. 3.

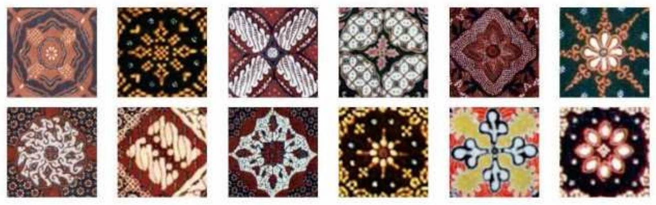

Fig. 3 Batik Patch Sample

The set of images from the dataset illustrated in Fig.2 has an unequal number of pattern repetitions. Therefore, another *Batik* image will be formed by cutting the patch previously performed, but it will be initially resized into 32x32. The new *Batik* image will have 16 patterns if the original size of the new *Batik* image is 128x128. Fig. 4 indicates a new and real *Batik*-making scenario with a similar number of pattern repetitions.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Fig. 4 A scenario of making new real Batik

The normalization process is implemented to modify the value in data, such as in an image, to have a range of values that are not much different. An image is formed from a collection of numbers with values ranging from 0 to 255. Hence, a closer value range of 0 generates a dark image, whereas a closer value range of 250 generates a brighter image. Therefore, these values will be carried out in the data normalization process. The numbers or values that make up the image will be reduced by 127.5 and further divided by 127.5. These calculations are generally implemented to normalize the data used in generative models employing the tanh activation function [26]. The results of the obtained data normalization process will have a range of values between -1 and 1.

The implemented data normalization method has similarities to the MinMax method, depicting a maximum and minimum limit for the results of the data normalization process [27]. However, the difference lies in their values which can determine these limits according to their respective needs (commonly used limit values are [0, 1] or [-1, 1]).

C. *Architecture Model*

At this stage, the section explains the series of implemented architectural models. The model applied in this study is BatikGAN SL with the addition of a content loss function based on the concept of neural transfer style employing the content loss and style loss functions [12] to navigate the style loss in the BatikGAN SL model without using the content loss function.

The first step in the model is performed by entering a collection of *batik* patches on the generator. Each patchA and patchB *Batik* will be entered into generator B1 and generator B2. Both generators will produce a series of matrices called feature maps. The result of feature maps from the two generators will be combined into a 128x128 image form. In the image formation, there is a local loss of the l2 function, leading to the image having a *Batik* pattern. The resulting image from the generator will be compared with the original *Batik* image by applying several loss functions, including style loss, adversarial loss or global discriminator loss, and local loss (local adversarial loss + local style loss). In addition, the image generated by the generator will be compared with the *Batik* patch input using the content loss function, as illustrated in Fig. 5.

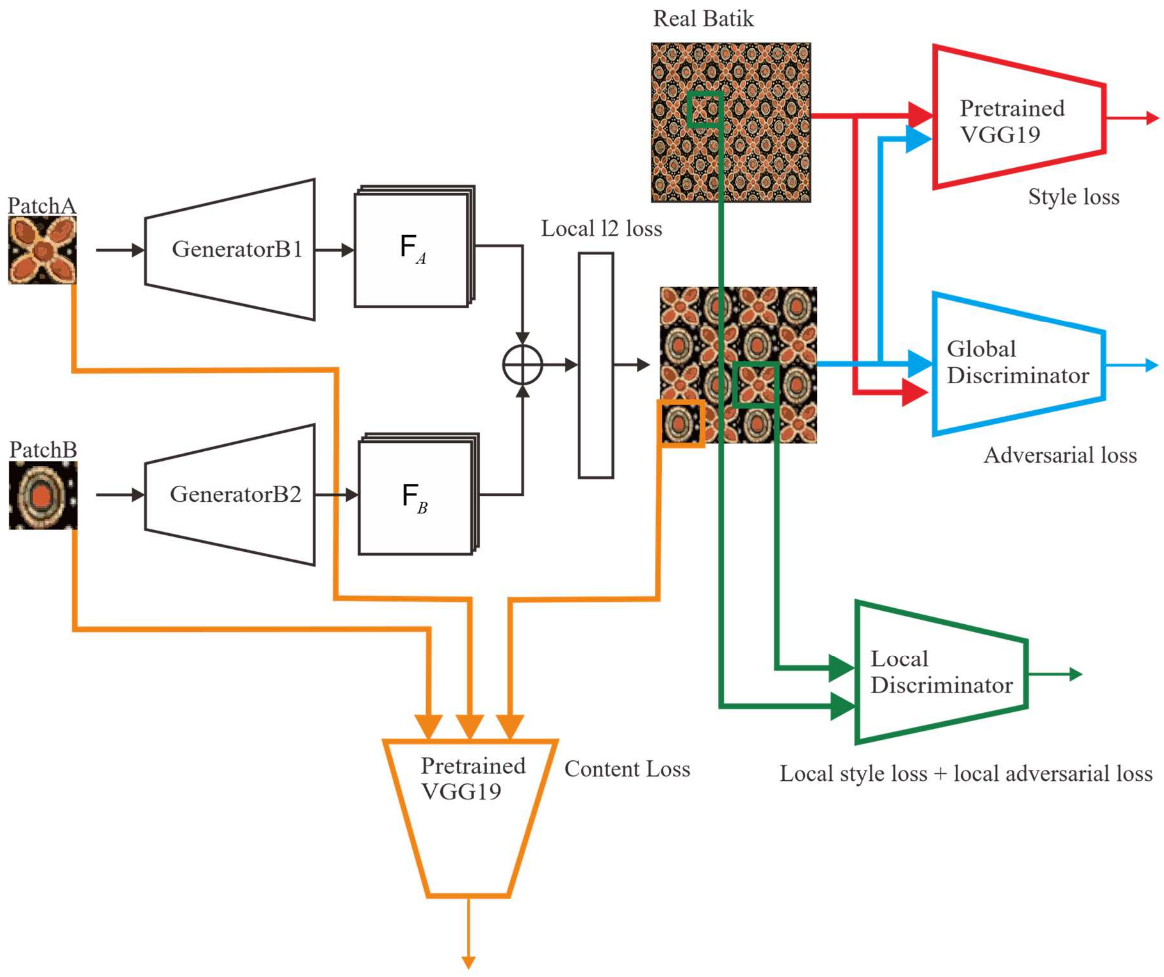

Fig. 5 Batik GAN SL with content loss

D. Generator

The generator is utilized to generate a new image based on the available dataset. Generators, in general, will continue to produce an image until achieving the original image. However, in this study, the generator will combine the user's patch input with the original data set to generate a new *batik* image.

Furthermore, a local_l2 loss function is required to produce *batik* images that have to repeat and intersect patterns. The loss function will force the generator to form repeated *batik* patterns. Therefore, the *batik* results from the generator will be cropped into 16 images divided into two blocks, including block_y with index {1, 3, 6, 8, 9, 11, 14, 16} and blok_z with index {2, 4, 5, 7, 10, 12, 13, 15} [24]. The loss calculation is also acknowledged as a reconstruction error [28], illustrated in the following equation (1). In addition to using the loss function, an optimal generator model architecture design is required, as illustrated in Fig. 7.

$$\mathcal{L}_{local\_l2} = \sum_{y \in Y} \sqrt{\left(Patch_A - Block_y\right)^2} + \sum_{z \in Z} \sqrt{(Patch_B - Block_z)^2} \quad (1)$$

where:

$Patch_A$	: *batik* cut A
$Patch_b$	: *batik* cut B
$Block_y$	: arrangement of *batik* block y
$Block_z$	: arrangement of *batik* block z

E. Discriminator

Basically, the image generated by the generator is useless, thereby requiring a partner (the discriminator) to accompany the generator. The discriminator in this step will check the image result from the generator whether it is genuine or fake. The inspection result is in the form of an error or loss value that will be given to the generator as a reference to produce an image. The architecture of the discriminator is illustrated in Fig. 6.

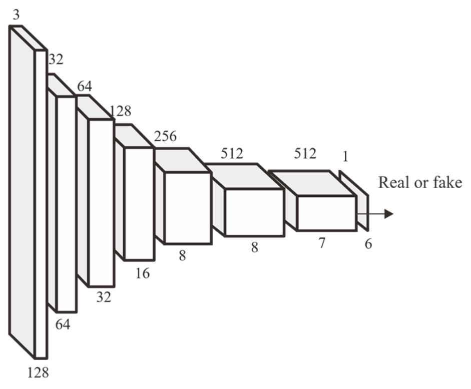

Fig. 6 Discriminator model architecture

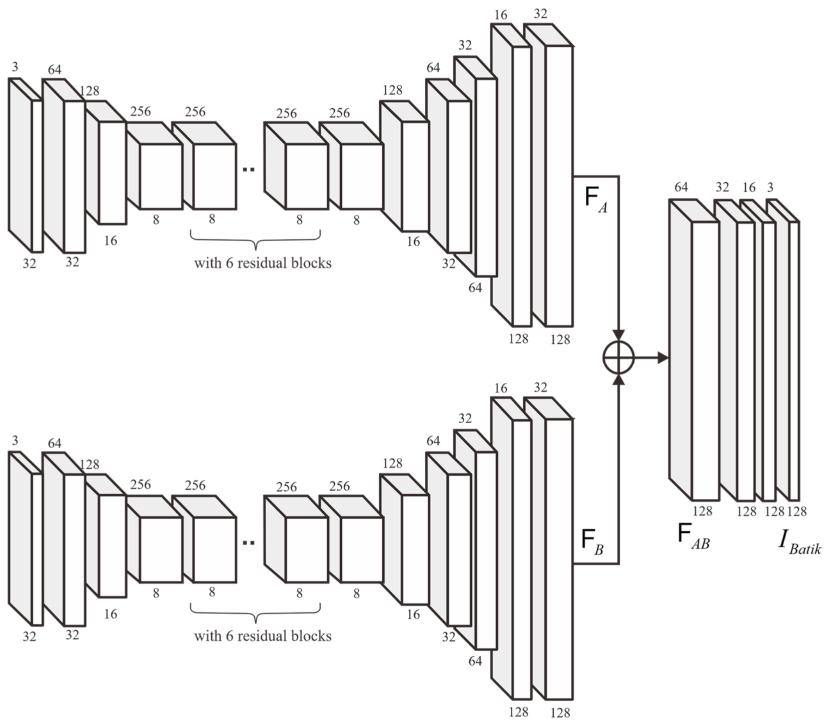

Fig. 7 Generator model architecture

F. Loss Function

In addition to implementing the right model architecture, a calculation method or function is required. The discriminator contains several functions to check the results of the image by the generator. The loss functions include adversarial loss, style loss, and local loss.

1) Adversarial Loss: Adversarial loss is employed to calculate the loss value between the *Batik*-generated results and the original *batik* image, as illustrated in equation (2).

$$\mathcal{L}_{adv}(I_{batik}, D_{global}) = \mathbb{E}_x[\log D_{global}(x)] + \mathbb{E}[\log(1 - D_{global}(I_{batik}))] \quad (2)$$

where:

D_{global}	: overall image discriminator value
I_{batik}	: generated image
x	: original image

2) Style Loss: The pre-trained VGG-19 model is employed to calculate the style loss, further used to calculate the loss value between the generated *Batik* and the original *Batik* image. In this VGG-19 architecture, block1_conv1, block2_conv1, block3_conv1, block4_conv1, block5_conv1 will be calculated, further, to calculate the gram matrix. The result of the gram matrix from the generated image with the original image will be added up to calculate the distance.

3) Local Loss: Local loss combines several loss functions, including local l2 loss, local adversarial loss, local style loss. This loss function is used to calculate the loss value between the *Batik*-generated results and the *Batik* image in the form of a patch. Therefore, the batik images which were previously were divided into 16 parts to calculate its loss value. However, it is only used on local style and local adversarial loss due to the ability of the local l2 loss function in calculating the loss value between the *Batik* image produced and the *Batik* patch input. The local adversarial loss and local l2 loss function equations are illustrated in equations (3) and (4), as for how the local style loss works similarly to the style loss.

$$\mathcal{L}_{local\_l2} = \sum_{i,j,y \in Y} \sqrt{(Patch_A - Block_y)^2} + \sum_{z \in Z} \sqrt{(Patch_B - Block_z)^2} \quad (3)$$

where,

$Patch_A$	: *Batik* cut A
$Patch_b$	: *Batik* cut B
$Block_y$	: arrangement of the generated *batik* images y
$Block_z$	: arrangement of the generated *batik* images z

$$\mathcal{L}'_{adv} = \sum_m \left(D_{local}\left(C_m(x), C_m(I_{Batik})\right) - 1\right)^2 \quad (4)$$

where,

D_{local}	: local discriminator value
x	: original image
I_{batik}	: generate image
C_m	: *batik* patch

4) Content Loss: Content loss is usually performed with style loss. Therefore, the VGG-19 pre-training model will be used for this content loss calculation. The layer used is block5_conv1 as the image of the content or user input is in the form of a patch and is smaller than the resulting image, further cropped into 16 images. The image has been cut into two blocks of block_y with index {1, 3, 6, 8, 9, 11, 14, 16} and block_z with index {2, 4, 5, 7, 10, 12, 13, 15}. Basically, this loss calculation is similar to the local calculation of the previous 12 losses. However, the difference lies in the images which are initially passed into the VGG layer feature. The formula for calculating content loss is illustrated in equation (5).

$$\mathcal{L}_{content} = \sum_{i,j,y \in Y} \sqrt{\left(F_{ij}(Patch_A) - F_{ij}(Block_y)\right)^2} + \sum_{z \in Z} \sqrt{\left(F_{ij}(Patch_B) - F_{ij}(Block_z)\right)^2} \quad (5)$$

where,

$\boldsymbol{Patch_A}$	: *Batik* cut A
$\boldsymbol{Patch_b}$	: *Batik* cut B
$\boldsymbol{Block_y}$	: arrangement of the generated *batik* images y
$\boldsymbol{Block_z}$	: arrangement of the generated *batik* images z
F	: function layer feature

G. Neural Transfer Style

Neural style transfer refers to a method for transferring style from an image to the desired image [13] by utilizing a pre-trained CNN model to extract features from both images. The commonly used pre-trained model is VGG, performed by inserting the extracted features into the original image to preserve certain content by enhancing the content information of the original image.

In order to obtain the desired image, the neural transfer style applies the two loss functions: content loss and style loss. The content loss will ensure that the resulting image is not much different from the original image by utilizing a pre-trained model layer to extract features. The style loss attempts to embed the target image style on the resulting image by using five pre-trained model layers to extract features, further calculating the gram matrix for style loss. This step is performed to capture the distribution of features from a set of layers.

VGG19 refers to a CNN model with a depth of 19 layers consisting of six main structures, including several connected convolution layers and a full-connected layer. The convolutional kernel size is 3 * 3, and the input size is 224 * 224 * 3. The structure of the VGG-19 model is illustrated in Fig. 8.

Compared to other CNN models, this model has been improved in depth by employing an alternating structure of multiple convolutions and non-linear activation layers. This layer structure is better as it can extract image features by employing Maxpooling for down sampling and modifying the linear unit (ULT) as the activation function by selecting the largest value in the image area as the collected area value. The down-sampling layer is utilized to increase the anti-distortion capability of the model to the image, maintain the sample's main features and reduce the number of parameters [29].

H. Hyperparameter Loss

Hyperparameter is employed in selecting attribute values that are usually applied to a model. This study uses hyperparameters to weigh the content loss function by selecting the best weight loss value. Content loss is weighted as a loss function that will be added to this study. Thus, the appropriate weight value for the content loss function remains uncertain. There are three weight values 1, 10, and 100. The choice of the weight value refers to using the weight value of another loss function that already exists. The loss weight value used in other existing loss functions is illustrated in Table I. The weight value refers to that in previous studies [24].

TABLE I
ORIGINAL LOSS FUNCTION WEIGHT VALUES FROM PREVIOUS STUDIES

Function	Loss Value
Adversarial Loss	1
Local L2 Loss	1
Stye loss	10
Local Adversarial Loss	100
Local Style Loss	1

I. Training Model

The training model is implemented to train the model by updating the existing weights by alternating every epoch in training the model. This training model will be conducted for 1000 epochs. The loss function as mentioned in the previous sub-chapter plays a role in determining the weight value. The employed optimizer is Adam, with a learning rate of 0.0001 and a momentum of 0.5. This configuration is implemented in both generator and discriminator models.

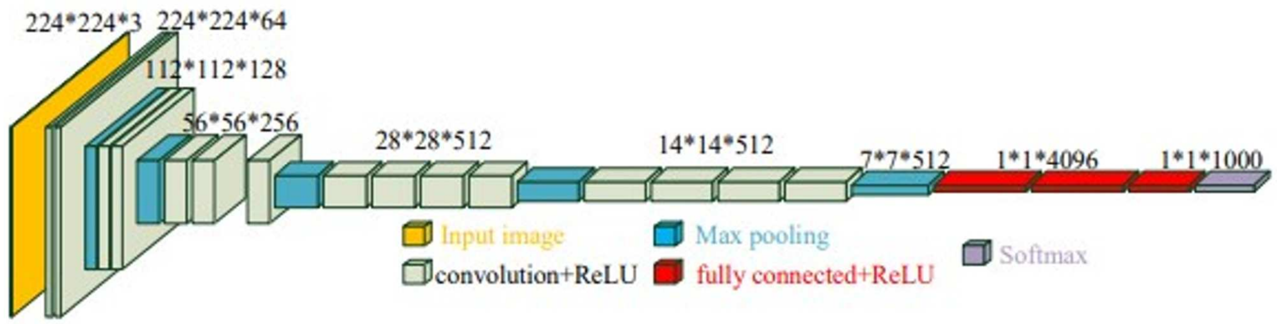

Fig. 8 VGG-19 model architecture

J. Evaluation

Evaluation is conducted to measure the performance results of the proposed method. The evaluation method used is the Frechet Inception Distance or FID, calculating the distance between feature vectors by assessing the level of image similarity [30]. The value of this metric is utilized to evaluate the image quality produced by the model, and lower scores are considered to correlate well with the original image. The FID calculation formula is illustrated in equation (6).

$$FID(r,g) = \left|\left|\mu_r - \mu_g\right|\right|_2^2 + Tr\left(\sum_r + \sum_g - 2\left(\sum_r \sum_g\right)^{\frac{1}{2}}\right) \quad (6)$$

where:

r	: original image
g	: generated image
T_r	: the sum of each element in the main matrix
μ_r	: average original image feature
μ_g	: average generated image feature
$\sum_r \sum_g$	: covariance matrix of real and fake images

There are two types of FID applied in this research, which are Global FID and Local FID. FID Global generates the distance value between the generated *Batik* image and the original *Batik* image. However, FID Local produces a distance value between the *Batik* image produced and the patch input from the user. Therefore, the Batik image must first be broken down according to the patch size, which is 32x32, to obtain the Local FID value.

III. Results and Discussion

The final result of this research is to evaluate the tests conducted on the proposed method by adding a similar pair of *Batik* patches to the model in order to produce a collection of *Batik* images. The results of the generated *Batik* images will be compared by calculating the FID value of each image.

Fig. 9 indicates the results of *Batik* images from each model, provided with similar pair of *Batik* patches. The replication model from the primary reference with the replication model without augmentation is assumed to generate similar *Batik* image results from all the patch pairs entered. Hence, there is no effect on *Batik* images between augmentation and without augmentation. Therefore, augmentation is not used to add the content loss function.

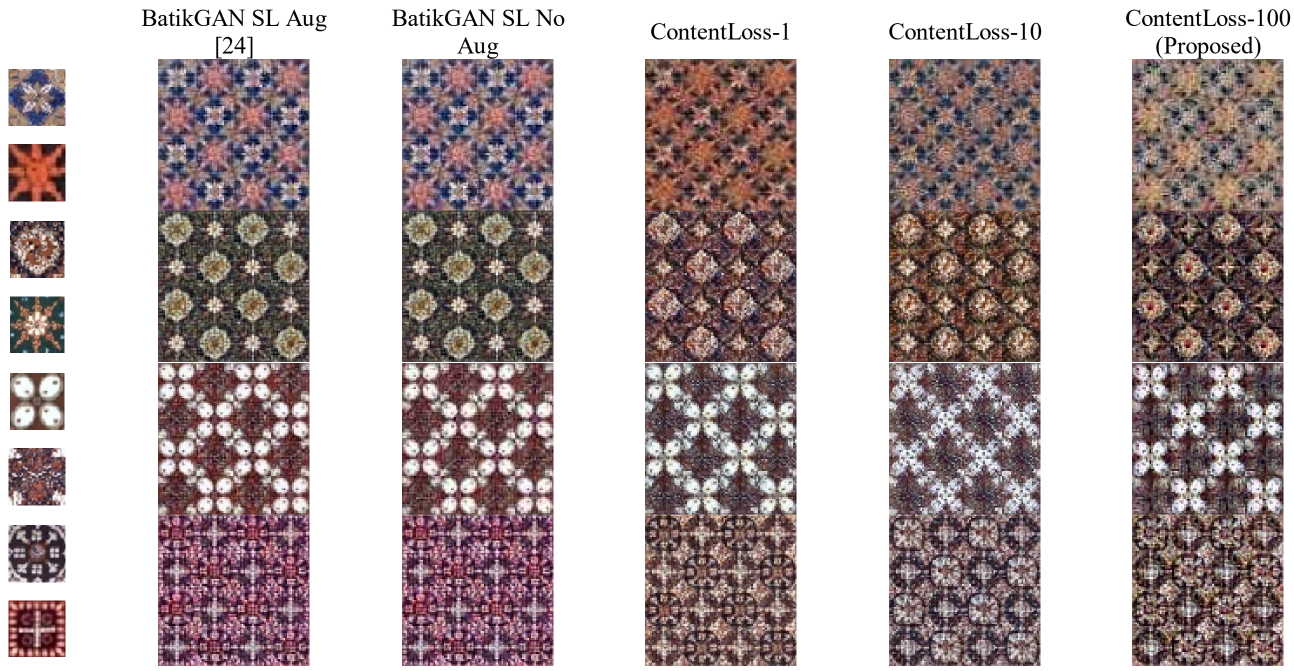

Fig. 9 Comparison of the results of the proposed model of batik drawing with the replication model of previous research

The content loss function is considered a new additional function that is not yet recognized as the exact loss weight to use. Therefore, hyperparameters are employed to select the right weight loss. Three loss weights will be selected, which are 1, 10, and 100. The loss weights are selected based on those used in other existing loss functions. The Batik results obtained from the ContentLoss-1, ContentLoss-10, and ContentLoss-100 models show differences in each patch pair. From the results of the generated *Batik* images, the proposed model changes the color and shape of the pattern due to the application of the neural transfer style.

TABLE II
COMPARISON OF THE PERFORMANCE OF THE PROPOSED MODEL WITH THE PREVIOUS RESEARCH REPLICATION MODEL

Model	FID Global	FID Local
BatikGAN SL Aug[24]	41	28
BatikGAN SL No	43	29
ContentLoss-1	42	16
ContentLoss-10	57	18
ContentLoss-100 (proposed)	41	20

Table II indicates the performance evaluation results on each model test by using FID. This study separates the use of FID values into two: Global FID and Local FID. FID Global calculates the distance between the original image and the model-generated image. Meanwhile, FID Local calculates the distance between the patch input entered into the model and the generated image, in which a smaller value indicates a better result as the image produced by the model resembles the original image.

The FID calculation results obtained from the primary reference replication without augmentation indicate a slight difference. This finding is also similar to the results of the similar image from the two models. For the proposed content loss model with one lose weight, the FID Global value indicates similar results as in the paper replication. However, FID Local indicates a much better value from the reference paper replication. In addition, the results of the generated images have changed from the replication of the primary reference paper, influenced by the content loss function, which complements the concept of neural transfer style in previous studies, and merely applied the style loss function.

The content loss model with a loss weight of 10 obtains a Global FID value that is much worse than the three models previously described. However, for the Local FID results, the value indicates similar results to the content loss model applying a weight of 1, and the generated image produces a slightly different image from the content loss of 1. The weight changes that occur appear to generate slightly different image changes. The last model of content loss with a loss weight of 100 obtains a better Global FID value than the previous models. However, there is a performance drop in the value results obtained by FID Local, and the decrease occurs in every step of the weight of the content loss function.

The results are depicted in Fig. 9; the image of the proposed model is compared with the results of replication in previous studies [19], thereby indicating a less striking difference, with all images depicting elements of patterns and colors from each *Batik* patch input. However, when viewed from the results of the FID calculation in TABLE , the proposed model differs from the replication results, especially in the results of the Local FID, where FID Local calculates the distance between the generated *Batik* image and the *Batik* patch input.

IV. CONCLUSION

Based on the performed tests, this study concludes that the model proposed in this study is eligible to maintain the style element from the entered *Batik* patch input. The tests on the proposed model with the addition of the content loss function obtain a fairly good Local FID value from the replication results in previous studies. The application of content loss with a weight of 1 receives a better Local FID value than content loss with a weight of 10 and 100. However, in FID Global, content loss with a weight of 100 is better than content loss with a weight of 1 and 10. On the other hand, the application of augmentation does not affect changes in the generated image. Hence, it is possible to add or modify the model by adding or subtracting several layers used in the generator or discriminator for further research. In addition, further research is encouraged to perform weighting on several other loss functions, such as the style loss function, adversarial loss, and local loss.

REFERENCES

[1] Y. Azhar, Moch. C. Mustaqim, and A. E. Minarno, "Ensemble convolutional neural network for robust batik classification," *IOP Conf Ser Mater Sci Eng*, vol. 1077, no. 1, p. 012053, Feb. 2021, doi: 10.1088/1757-899X/1077/1/012053.

[2] A. E. Minarno, F. D. S. Sumadi, H. Wibowo, and Y. Munarko, "Classification of batik patterns using K-Nearest neighbor and support vector machine," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 3, pp. 1260–1267, Jun. 2020, doi: 10.11591/EEI.V9I3.1971.

[3] A. N. Amalia, A. F. Huda, D. R. Ramdania, and M. Irfan, "Making a Batik Dataset for Text to Image Synthesis Using Generative Adversarial Networks," *Proceeding of 2019 5th International Conference on Wireless and Telematics, ICWT 2019*, Jul. 2019, doi: 10.1109/ICWT47785.2019.8978233.

[4] T. Tirtawan, E. K. Susanto, P. C. S. W. Lukman Zaman, and Y. Kristian, "Batik Clothes Auto-Fashion using Conditional Generative Adversarial Network and U-Net," *3rd 2021 East Indonesia Conference on Computer and Information Technology, EIConCIT 2021*, pp. 145–150, Apr. 2021, doi: 10.1109/EICONCIT50028.2021.9431867.

[5] A. E. Minarno, F. D. S. Sumadi, Y. Munarko, W. Y. Alviansyah, and Y. Azhar, "Image Retrieval using Multi Texton Co-occurrence Descriptor and Discrete Wavelet Transform," *2020 8th International Conference on Information and Communication Technology, ICoICT 2020*, Jun. 2020, doi: 10.1109/ICOICT49345.2020.9166361.

[6] A. E. Minarno, Moch. C. Mustaqim, Y. Azhar, W. A. Kusuma, and Y. Munarko, "Deep Convolutional Generative Adversarial Network Application in Batik Pattern Generator," *2021 9th International Conference on Information and Communication Technology (ICoICT)*, pp. 54–59, Aug. 2021, doi: 10.1109/ICOICT52021.2021.9527514.

[7] M. Abdurrahman, N. H. Shabrina, and D. K. Halim, "Generative adversarial network implementation for batik motif synthesis," *Proceedings of 2019 5th International Conference on New Media Studies, CONMEDIA 2019*, pp. 63–67, Oct. 2019, doi: 10.1109/CONMEDIA46929.2019.8981834.

[8] G. Tian, Q. Yuan, T. Hu, and Y. Shi, "Auto-generation system based on fractal geometry for batik pattern design," *Applied Sciences (Switzerland)*, vol. 9, no. 11, 2019, doi: 10.3390/app9112383.

[9] P. D. Kusuma, "Modified sine wave based model in madurese batik pattern generation," *J Theor Appl Inf Technol*, vol. 97, no. 23, pp. 3557–3569, 2019.

[10] L. A. Gatys, A. S. Ecker, M. Bethge, and C. V Sep, "A Neural Algorithm of Artistic Style," pp. 3–7.

[11] C. Zhou, Z. Gu, Y. Gao, and J. Wang, "An improved style transfer algorithm using feedforward neural network for real-time image conversion," *Sustainability (Switzerland)*, vol. 11, no. 20, pp. 1–15, 2019, doi: 10.3390/su11205673.

[12] Y. A. Irawan and A. Widjaja, “Pembangkitan Pola Batik dengan Menggunakan Neural Transfer Style dengan Penggunaan Cost Warna,” *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 6, no. 2, pp. 324–341, 2020, doi: 10.28932/jutisi.v6i2.2698.

[13] G. Atarsaikhan, B. K. Iwana, and S. Uchida, “Guided neural style transfer for shape stylization,” *PLoS One*, vol. 15, no. 6, 2020, doi: 10.1371/journal.pone.0233489.

[14] H. Kwon, H. Yoon, and K. W. Park, “CAPTCHA image generation: Two-step style-transfer learning in deep neural networks,” *Sensors (Switzerland)*, vol. 20, no. 5, pp. 1–14, 2020, doi: 10.3390/s20051495.

[15] A. Firman Ihsan, “A Study of Batik Style Transfer using Neural Network,” *2021 9th International Conference on Information and Communication Technology, ICoICT 2021*, pp. 313–319, Aug. 2021, doi: 10.1109/ICOICT52021.2021.9527490.

[16] M. Joseph, J. Richard, C. S. Halim, R. Faadhilah, and N. N. Qomariyah, “Recreating Traditional Indonesian Batik with Neural Style Transfer in AI Artistry,” *8th International Conference on ICT for Smart Society: Digital Twin for Smart Society, ICISS 2021 - Proceeding*, Aug. 2021, doi: 10.1109/ICISS53185.2021.9533197.

[17] I. J. Goodfellow *et al.*, “Generative adversarial nets,” *Adv Neural Inf Process Syst*, vol. 3, no. January, pp. 2672–2680, 2014.

[18] W. Wang, A. Wang, Q. Ai, C. Liu, and J. Liu, “AAGAN: Enhanced Single Image Dehazing with Attention-to-Attention Generative Adversarial Network,” *IEEE Access*, vol. 7, pp. 173485–173498, 2019, doi: 10.1109/ACCESS.2019.2957057.

[19] O. Kupyn, T. Martyniuk, J. Wu, and Z. Wang, “DeblurGAN-v2: Deblurring (orders-of-magnitude) faster and better,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-Octob, pp. 8877–8886, 2019, doi: 10.1109/ICCV.2019.00897.

[20] Z. Yuan *et al.*, “SARA-GAN: Self-Attention and Relative Average Discriminator Based Generative Adversarial Networks for Fast Compressed Sensing MRI Reconstruction,” *Front Neuroinform*, vol. 14, no. November, 2020, doi: 10.3389/fninf.2020.611666.

[21] Q. Jin, R. Lin, and F. Yang, “E-WACGAN: Enhanced Generative Model of Signaling Data Based on WGAN-GP and ACGAN,” *IEEE Syst J*, vol. 14, no. 3, pp. 3289–3300, 2020, doi: 10.1109/JSYST.2019.2935457.

[22] M. Abdurrahman, N. H. Shabrina, and D. K. Halim, “Generative adversarial network implementation for batik motif synthesis,” in *Proceedings of 2019 5th International Conference on New Media Studies, CONMEDIA 2019*, 2019, pp. 63–67. doi: 10.1109/CONMEDIA46929.2019.8981834.

[23] Y. Huang, J. Su, J. Wang, and S. Ji, “BatIK-DG: Improved deblurgan for batik crack pattern generation,” in *IOP Conference Series: Materials Science and Engineering*, 2020, vol. 790, no. 1. doi: 10.1088/1757-899X/790/1/012034.

[24] W. T. Chu and L. Y. Ko, “BatikGAN: A Generative Adversarial Network for Batik Creation,” in MMArt-ACM 2020 - Proceedings of the 2020 Joint Workshop on Multimedia Artworks Analysis and Attractiveness Computing in Multimedia, 2020, no. 1, pp. 13–18. doi: 10.1145/3379173.3393710.

[25] Y. Gultom, R. J. Masikome, and A. M. Arymurthy, “Batik Classification Using Deep Convolutional Network Transfer,” *Jurnal Ilmu Komputer dan Informasi (Journal of a Science and Information)*, vol. 2, pp. 59–66, 2018.

[26] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pp. 1–16, 2016.

[27] S. G. K. Patro and K. K. sahu, “Normalization: A Preprocessing Stage,” *Iarjset*, pp. 20–22, 2015, doi: 10.17148/iarjset.2015.2305.

[28] S. Ganguli, P. Garzon, and N. Glaser, “GeoGAN : A Conditional GAN with Reconstruction and Style Loss to Generate Standard Layer of Maps from Satellite Images,” *ArXiv*, 2019.

[29] J. Xiao, J. Wang, S. Cao, and B. Li, “Application of a Novel and Improved VGG-19 Network in the Detection of Workers Wearing Masks,” *J Phys Conf Ser*, vol. 1518, no. 1, pp. 0–6, 2020, doi: 10.1088/1742-6596/1518/1/012041.

[30] A. Borji, “Pros and cons of GAN evaluation measures,” *Computer Vision and Image Understanding*, vol. 179, pp. 41–65, 2019, doi: 10.1016/j.cviu.2018.10.009.