[redacted]

Research Area : Computer Vision

Paper Title,Writer, Name of Journal/Conference and Publised Year :

1. Generative Adversarial Network Implementation for Batik Motif Synthesis, (Miqdad Abdurrahman , Nabila Husna Shabrina, Dareen K.Halim), 2019 5th International Conference on New Media Studies (CONMEDIA) , 2020

Research Problem :
How to generate batik motif using Generative Adversarial Network ?

Proposed Method :
The proposed methods are using Deep Convolutional GAN and Gradient Descent Optimization

a. Deep Convolutional GAN is an improvement of GAN proposed by Alec Radford and Luke Metz. There are a few guideline which is listed as follows : :
   - Use batchnorm in both generator and discriminator
     - Batchnorm basically is a method to normalize the input from each layer and it is applied for each batch in neural network !
       - Improve the computation speed
   - Remove Fully connected hidden layers for deeper architectures
   - Use ReLU activation in generator for all layers except for the output layer, Which is Use Tanh
   - Use LeakyReLU in the discriminator for all layers

   Explanation why generator using ReLU while discriminator using LeakyReLU.
   - Generator is generating new images that can fool the discriminator therefore it need fast computation and efficient.
   - Discriminator task is to distinguishing between real and fake images generated by the generator. Therefore it need an activation function that able for always learning. ReLU has the Dying ReLU problem. Which is when the neuron using this activation func output ZERO, in which causing the gradient to become zero during backpropagation. It caused the discriminator cannot learn again !

Explanation why using Tanh

- In DCGAN, the generator network is designed to output images that have pixel value in range [-1,1].

Explanation why removes fully connected layers for deeper architectures

- Efficiency : Fully connected layers have a large number of parameters which can make training slower and more difficult. Therefore removing the fully connected layers make the training much more efficient
- Avoiding Overfitting : Because many parameters turns too complex model which can make the model try to memory the training data instead of learning generalizable pattern from the inputs. By removing it, the number of parameters reduced which improve the model's ability to generalize the future inputs
- Better Feature Learning : Because all fully connected layers are removed therefore it makes the feature learning capability of the model much more better. By Using Convolutional layers, the model can capture spatial relationship between pixels, while fully connected layers treat the input pixel independently, which 's make to extract spatial relationship
    - Spatial relationship refers to cues/facts to recognize something
        - For example, spatial relationship between the eyes ,nose, mouth in a facial image is an important cues for recognizing the face

b. Gradient Descent Optimization

In this research there are 2 optimization algorithm used. Adam and RMSProp. And there are 3 types of gradient descent

- Batch gradient descent
    - Basic form of gradient descent. Where the weights are updated when after processing the entire dataset. Slow and memory extensive
    - Advantages
        - Better convergence to GLOBAL minimum / maximum
        - More Stable convergence : Less noisy, by using the whole dataset. It means the estimated variance of gradient is reduced.
    - Disadvantages
        - Computationally Expensive : Batch Gradient Descent requires computing gradient for the entire dataset at every iteration !
        - Memory Expensive : Since it need to load the entire dataset into memory at once !

        - Slow convergence : Batch Gradient Descent updates the gradient is very slow since it need the entire dataset to be computed first
        - Very Sensitive to learning rate : If it set to a big learning rate, it will increase the possiblity gradient exploding problem and vice versa. Too low will make the model could not learn at all.
- Stochastic gradient descent
    - This is the subset of batch gradient descent. The weights are updated after **processing EACH training example**. Of course it is more efficient & reliable
    - Advantages
        - Better Generalization : SGD use the randomness into the training process, which can help the model to avoid overfitting and generalize better to unseen data.
        - Low memory requirement : SGD only use one training example at a time.
        - Can handle non-convex (has multiple local minima) problem : SGD more likely to escape the local minimal because its randomness feature at each of the gradient computation.
    - **Overall SGD can be an effective solution**, especially for large and complex deep learning models. As it has a lot of local minima and and a big parameters .
- Mini-batch gradient descent
    - The combination between batch gradient descent and stochastic gradient descent. By creating a small batch to compute the gradient and then update the weights in each iteration
    - Advantages
        - Faster Convergence : Mini batch (128 / 512 / at least not entire dataset) will updates the model parameters **more frequently** , leads to faster **convergence**
        - Efficient Memory usage : By using mini batches, the algorithms can process the **training** in small chunks, therefore reduce the **memory usage.**
    - Disadvantages
        - Requires Tuning : **Batch size,** the batch size needs to be tuned well based on **the data characteristic**.
        - Potential Noisy Updates : As it is mini-batch, it could be some cases all data inside the batch are noisy which can make

learning / the gradient is not representative. I.e totally differs from the general result !

- Requires **careful learning rate selection :** Continued the above statement, becareful we did not know wheter the mini batch data is representative and not noisy, choosing learning rate with the coressponding batch can be challenging. Since if it wrong, it could make the learning messy.

Table I Model Architecture information

Generator	Discriminator
Strided Convolution layers for upsampling	Convolution layers for downsampling
Initial weight std 0.02	Initial weight std 0.02
Batch norm for all layers except output	Batch norm for all layers except output
ReLU activation function for all layers but output	LeakyReLU activation function for all layers but output
Tanh output activation layer	Sigmoid output activation layer

Table II Model training parameters information

Model	Optimizer	Learning rate	Batch size	Momentum
Discriminator#1	Adam	0.002	128	0.5
Generator #1	Adam	0.002	128	0.5
Discriminator#2	RMS-Prop	0.002	128	0
Generator#2	RMS-Prop	0.002	128	0

Explanation of the parameters :

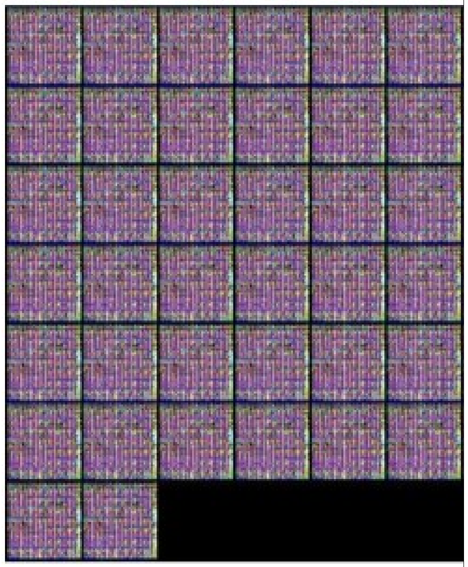
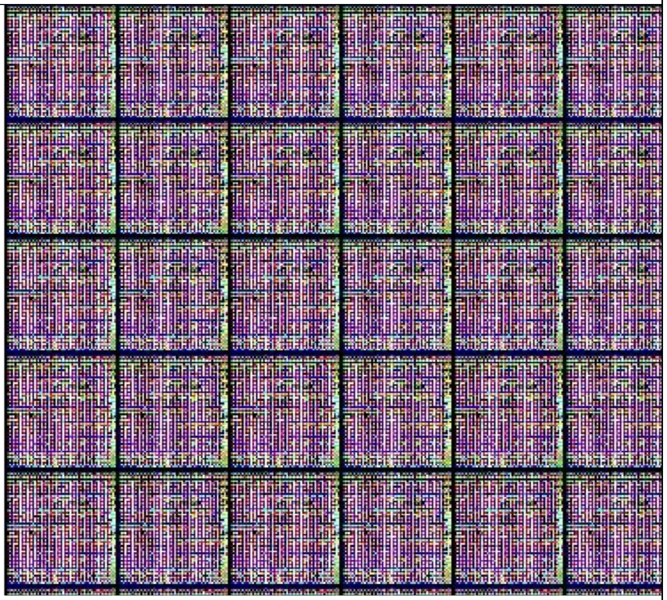
- Optimizers
  - Adam (Adaptive Moment Estimation)
    - Is an optimization algorithm that combines RMSProp with momentum
  - RMS-Prop
    - Is an optimization algorithm that adjust the learning rate adaptively for each weight
- Learning rate
  - A constant value for learning
- Batch Size

  - Value for implement the mini-batch gradient descent. Every 128 observed data in dataset will then used to calculate the gradient, and then update the weight.
- Momentum
  - Is a techniques used to speed up the convergence of the Adam optimization algorithm.

Experiment Result :

Table III shows the result for AdamOptimizer and table IV shows the result for RMSProp

Table III Adam Optimizer results

Epoc hs	Adam optimizer result	Details
4550		

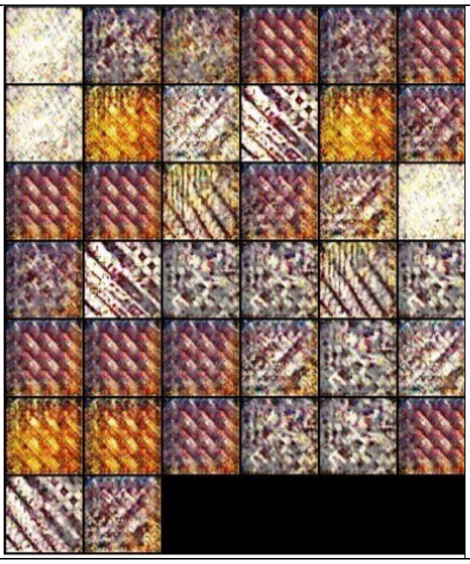
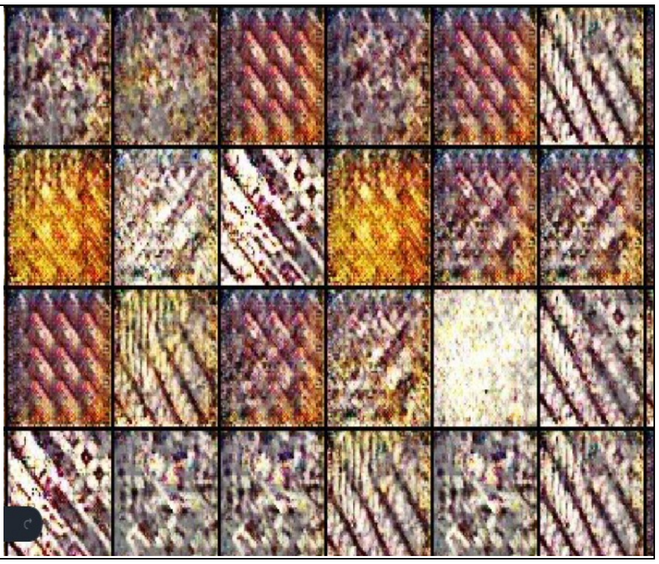
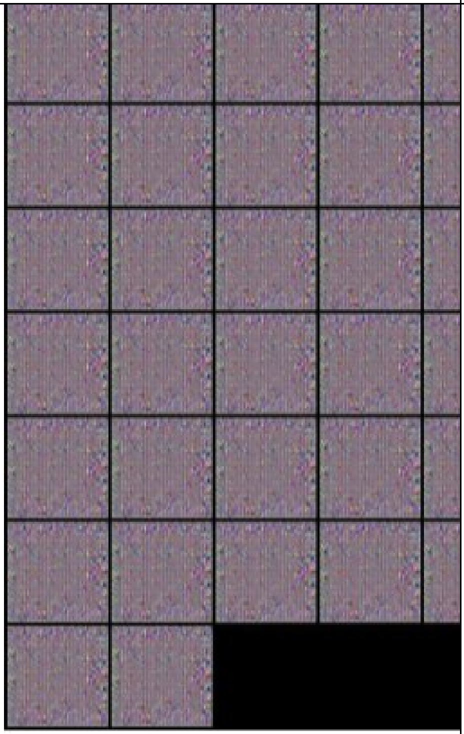
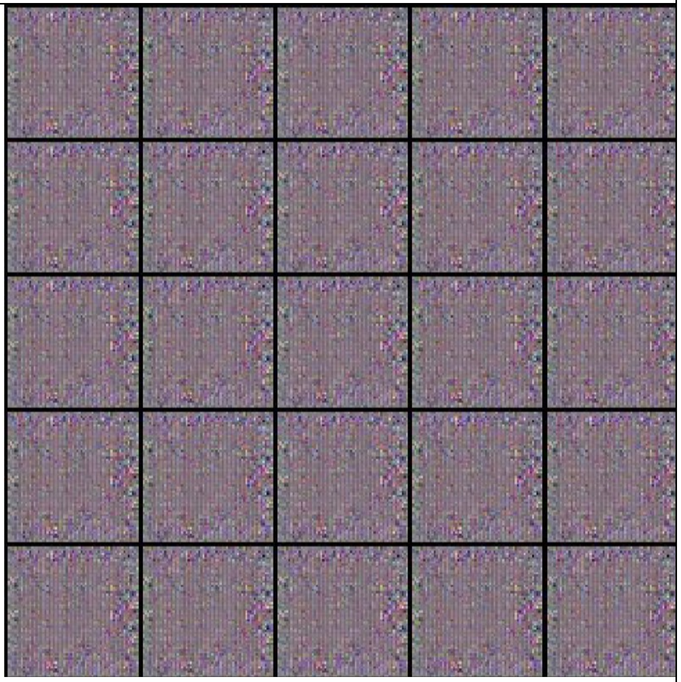
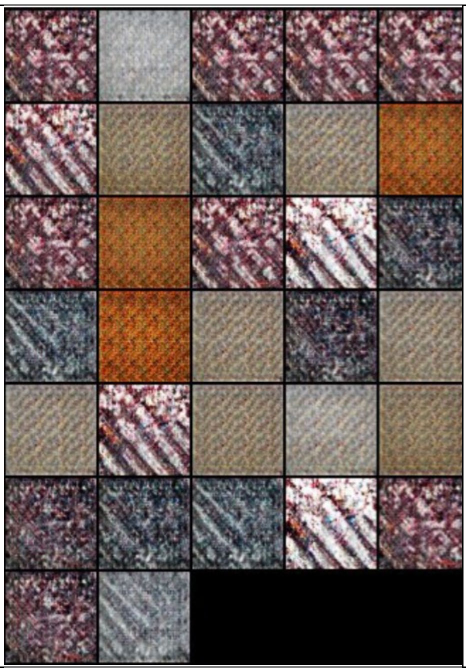
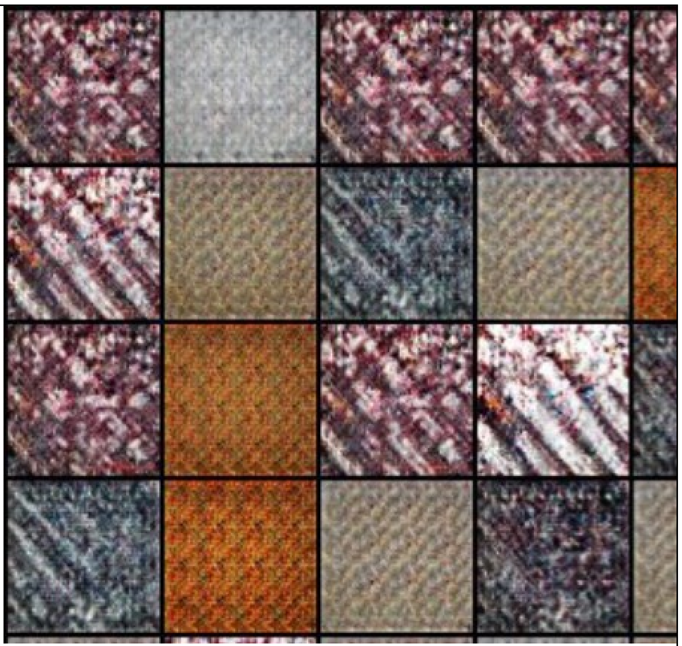
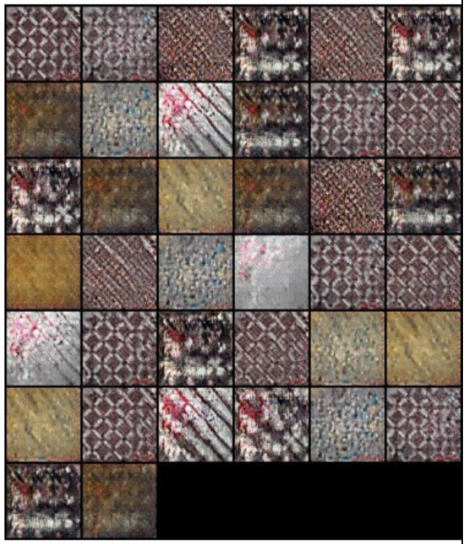
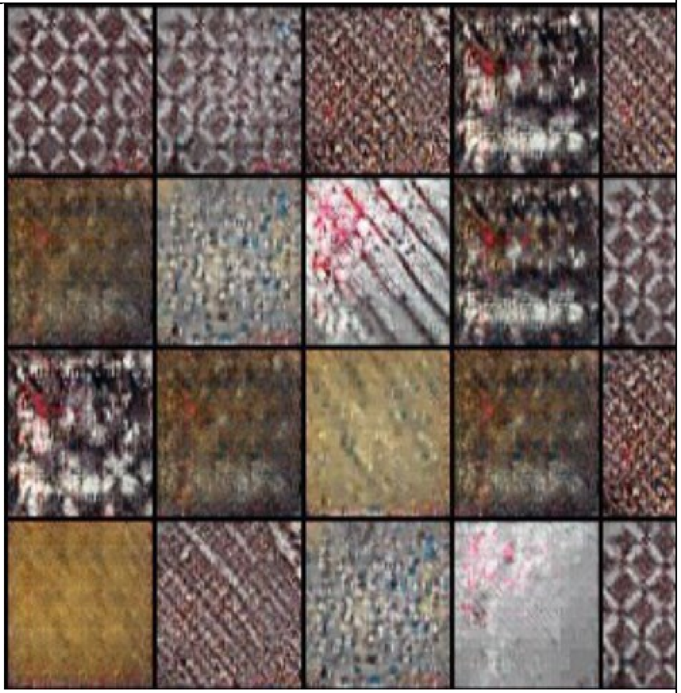
8500		

Table IV RMSProp Results

Epochs	RMSProp optimizer result	Details
4550		

4999		
9400		

From their experiment, we can say. The training takes big amount of time and resources, while the result is not necessary good for the most part. At epochs 8500

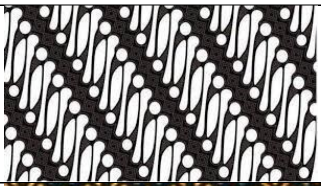
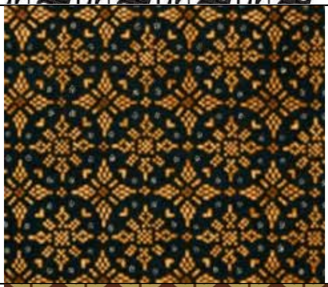
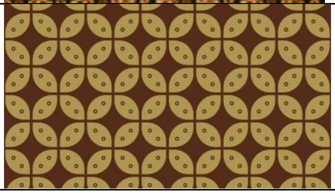
with adamOptimizer, model able to generate batik parang. But its not necessarly completed batik parang. And it takes nearly 24 hours to generate the results.
From their comparasion, a better result can be achieved by using the adam optimizer.

Paper Contribution :

- Preserving Indonesian culture
- It can be used as a baseline to generate more reliable batik shape in the future
- It can be used as a baseline to create an application for introducing batik / generating batik for various use case.

Paper Drawback/Limitation :

- It shown DCGAN able to generate batik shape, but unfortunately the explanation regarding the output is not clear
- The training step takes a long time to able generating shape, 9400 epochs
- Batik patterns used are shown below

Batik Name	Batik Pattern
Parang	
Nitik	
Kawung	

- It does not provide explanation regarding the layer output information

2. **Paper Title,Writer, Name of Journal/Conference and Publised Year :**

A Study of Batik Style Transfer using Neural Network, Aditya Firman Ihsan, 2021 9th International Conference on Information and Communication Technology (ICoICT), 2021

Research Problem :

1. How the neural style transfer can reconstruct the batik style using pretrained CNN architecture ?
2. How to merge the reconstructed batik style to arbitrary image ?

Proposed Method :

- Style Reconstruction and Transfer
  - o Style reconsturction is an image generative task based on the style of a target image. The style of an image is defined by the correlations of deep feature on each layer of the neural network (Setiap layer memiliki korelasi/hubungan terhadap fitur). The covariance matrix between these features known as **Gram matrix** is used as numerical representation of the "style". Here's the mathematical form

$$G_l(I_s) = F_l(I_s)\, F_l'(I_s)$$

  - o
  - o So the Gram Matrix for layer$_s$ is equal to dot product of **FeatureTensor** of style image layer$_s$ and Inverse **FeatureTensor** for style image layer$_s$
  - o Therefore the Style loss can be defined as **squared error** between **gram matrix of** the target image **I**$_s$ and **gram matrix** of the generated image **I.**

$$L_S = \sum_l \left[\alpha_l \left(G_l(I_s) - G_l(I)\right)^2\right]$$

  - o
    - ▪ α_l is a style weight. Managing the style contribution of layer **l**. The bigger it is, the more style will be implemented on the layer
  - o In this research **a uniformly black image** is used as the initial **Generated image** to ensure the uniformity of the style
    - ▪ Black image has no style information.
  - o **Style reconstruction** can be modified (dapat dirubah) to **Style transfer** by including a **Content image** as **a BASE CANVAS** in the **reconstruction process**. **To do so**, an additional loss term is added to

Preserve high level information (Agar informasi penting content image tidak hilang) of the **content image as** the style is **reconstructed. Content loss is** defined directly to the respective layer **feature representation**

$$L_c = \sum_l [\beta_l (F_l(I_s) - F_l(I))^2]$$

- o
  - βl is the content weight of layer l.
- o Therefore the **Total Loss** can be calculated as
  - $$L = a^c L_c + a^s L_s$$
    - Tuning the ratio $a^s : a^c$ wil determine how much style will be painted to the content image. For example
      - o 3 : 1 , for every 3 pixels of the style draw 1 pixel of the content
- o Diagram of style transfer process

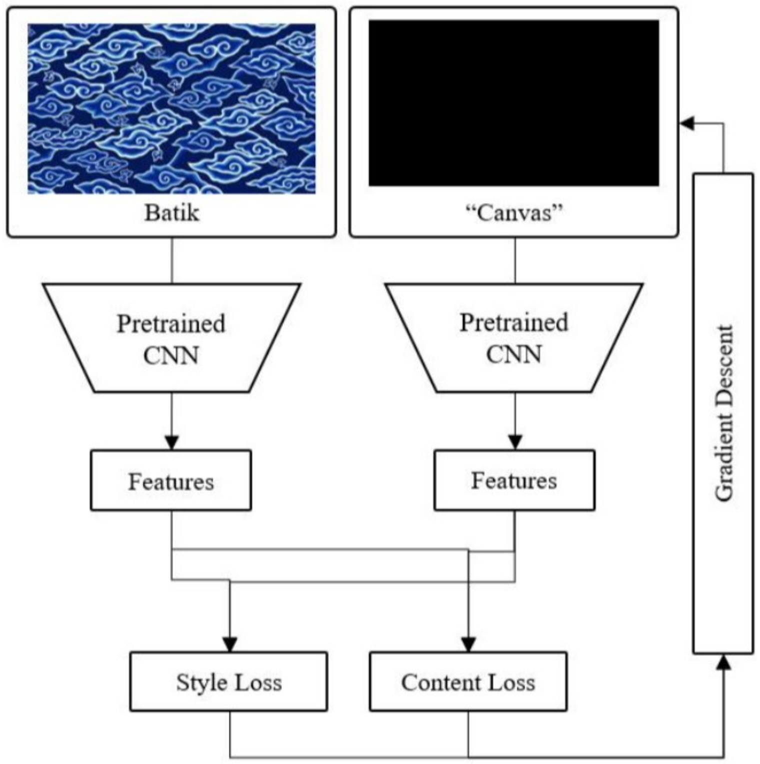
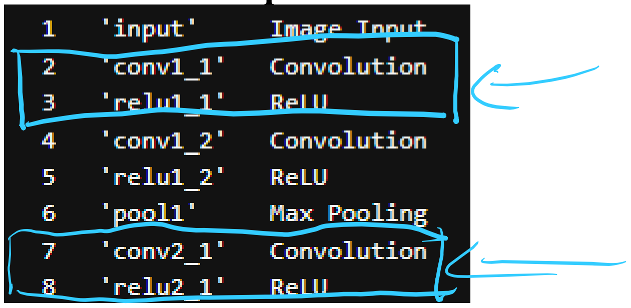


Figure 1 Diagram of style transfer process

- **Stylization** from differenct network layers

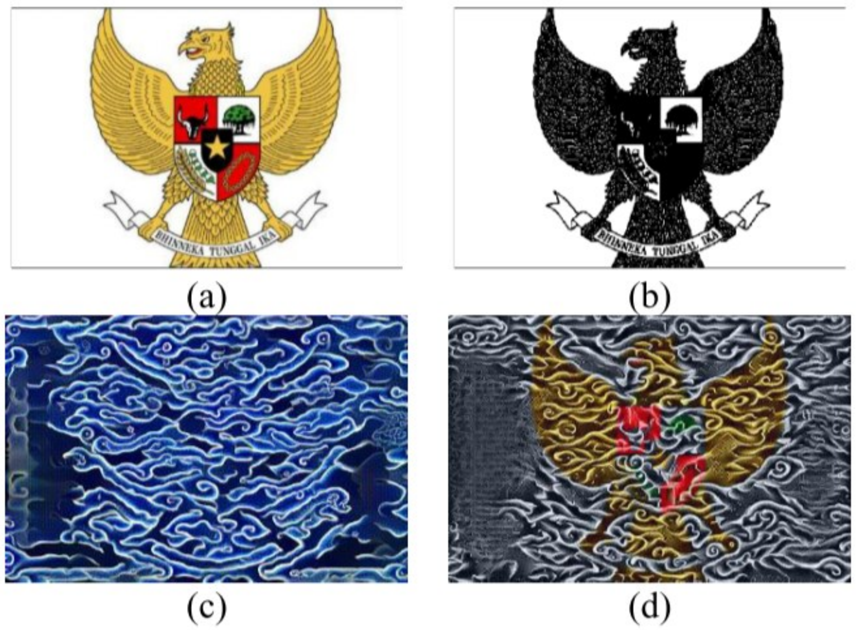
- **Each layer** in **CNN** maps different **style feature** of the image. Therefore choosing the right combination of the style layers is imporant to obtain **a great stylized image**.
    - **Pemilihan layer convolution biasanya dilakukan sebelum activation layer** biasanya di pilih karena memberikan **filtered image representation.** Pemilihan dapat dilakukan kembali saat telah melakukan **Pooling** or
        - **Here's the example of VGG19 architecture**
        - 

```
1   'input'     Image Input
2   'conv1_1'   Convolution
3   'relu1_1'   ReLU
4   'conv1_2'   Convolution
5   'relu1_2'   ReLU
6   'pool1'     Max Pooling
7   'conv2_1'   Convolution
8   'relu2_1'   ReLU
```

          That mean, layer **conv1_1 or conv2_1 can are the best candidate.**
        - Batch Normalized → the layers value is normalized for faster computation
        - **Downsampled** → Removing some of **feature maps(it just a bunch of pixels**) from the input, while retaining the most important information **(Pooling layer)**
        - The deeper layer **Usually give more** Abstract texture representation of the image. Because **batik motifs** memliliki struktur yang **kaku dan solid** dan juga banyaknya **local patterns, therefore** layer analysis is needed to find a suitable layer to be used
    - In this research, there are 4 models used
        - VGG-19 (**Very Deep Convolutional Networks for large scale image recognition)**
        - ResNet-50 (**Residual Network)**
        - Inception-V3
        - DenseNet-121

- **Batik-Textured Image**
    - Use style transfer to generate batik textured image. I.e recreate artwork of **Batik pattern** that contains a **Shape** of an **object** abstractly !
        - Use VGG-19 Architecture
            - Layer block3_conv1

        - A high value of **Ratio : content** is also chosen
    - Preprocessing step
        - Enhancing the image contrast
        - Converting image to 2-bit (black white) not grayscale
            - Does not work well for some cases it can be seen when restoring the image
    - **Reconstruction step**
        - **Do** the figure1 and change the content image !
    - Restoration Step
        - Merging channels from the original **Content image** and the **Stylized image**
        - Using **YUV** color Space
            - Extract **Y** from **stylized image** and merge it with **U & V Channels** from the **original content image.**
                - **The technique does not** always work , if the **content image** have a lot **White areas** !

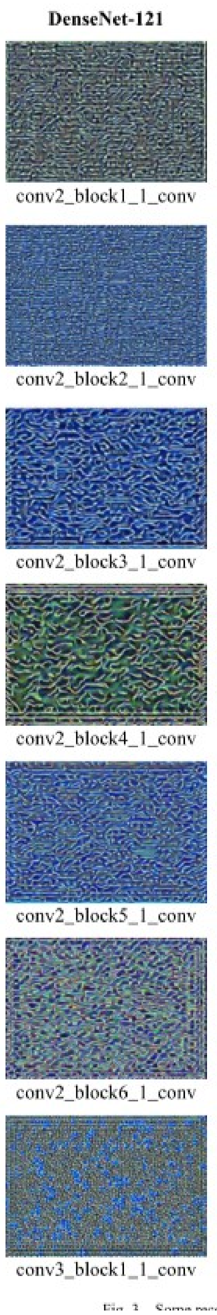
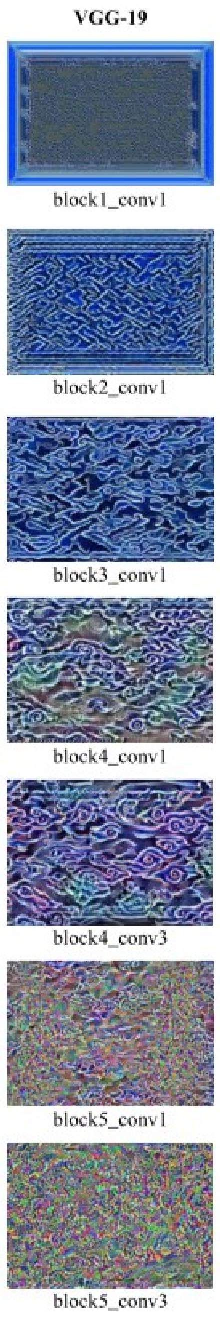
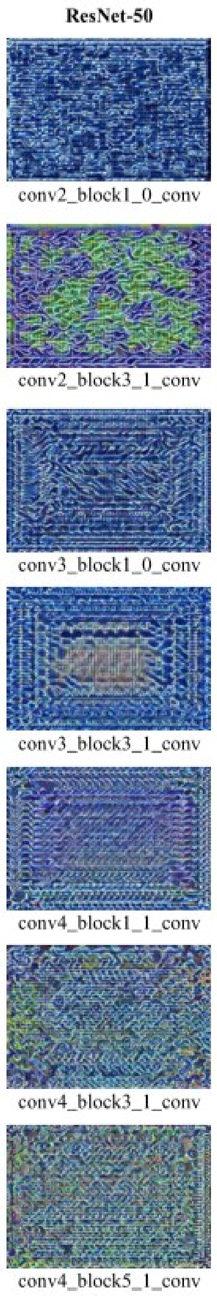
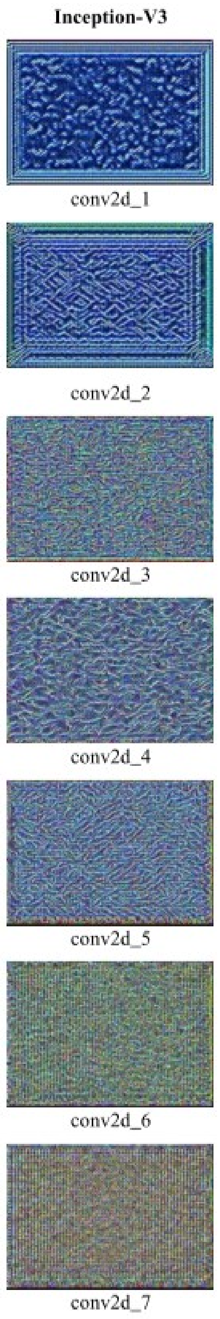


o

Figure 2 (a) Original Image; (b); Preprocessed image; (c) Stylized Image; (d) Stylized image with restored original color

Experiment Result :

- Style Transfer batik experiment result

DenseNet-121	VGG-19	ResNet-50	Inception-V3
 DenseNet-121 conv2_block1_1_conv conv2_block2_1_conv conv2_block3_1_conv conv2_block4_1_conv conv2_block5_1_conv conv2_block6_1_conv conv3_block1_1_conv	 VGG-19 block1_conv1 block2_conv1 block3_conv1 block4_conv1 block4_conv3 block5_conv1 block5_conv3	 ResNet-50 conv2_block1_0_conv conv2_block3_1_conv conv3_block1_0_conv conv3_block3_1_conv conv4_block1_1_conv conv4_block3_1_conv conv4_block5_1_conv	 Inception-V3 conv2d_1 conv2d_2 conv2d_3 conv2d_4 conv2d_5 conv2d_6 conv2d_7

From the table above we can see that as the network getting deeper, the convolutional layer maps more **abstract style features** from the style image. Losing the pattern that need to be extracted from the batik pattern **Rigid** and **solid. Because** batik need to have a **detailed structure** thus using the **deeper layer** feature maps representation can't be used. Of all reconstruction **Vgg-19** gives the closest generated style to the **target image.** The preserved result caused by the **Vgg-19** has shallower network and

small **filter-size (3x3) . Face like pattern such as batik bali or batik sidomukti are failed to be reconstructed**. But, a repeating pattern/ small pattern such as **Batik parang or kawung** can be recreated.
T**he reconstructed algorithm** is best suited for **small localized** pattern **Batik**.

Figure 3 shows the results for the given inputs

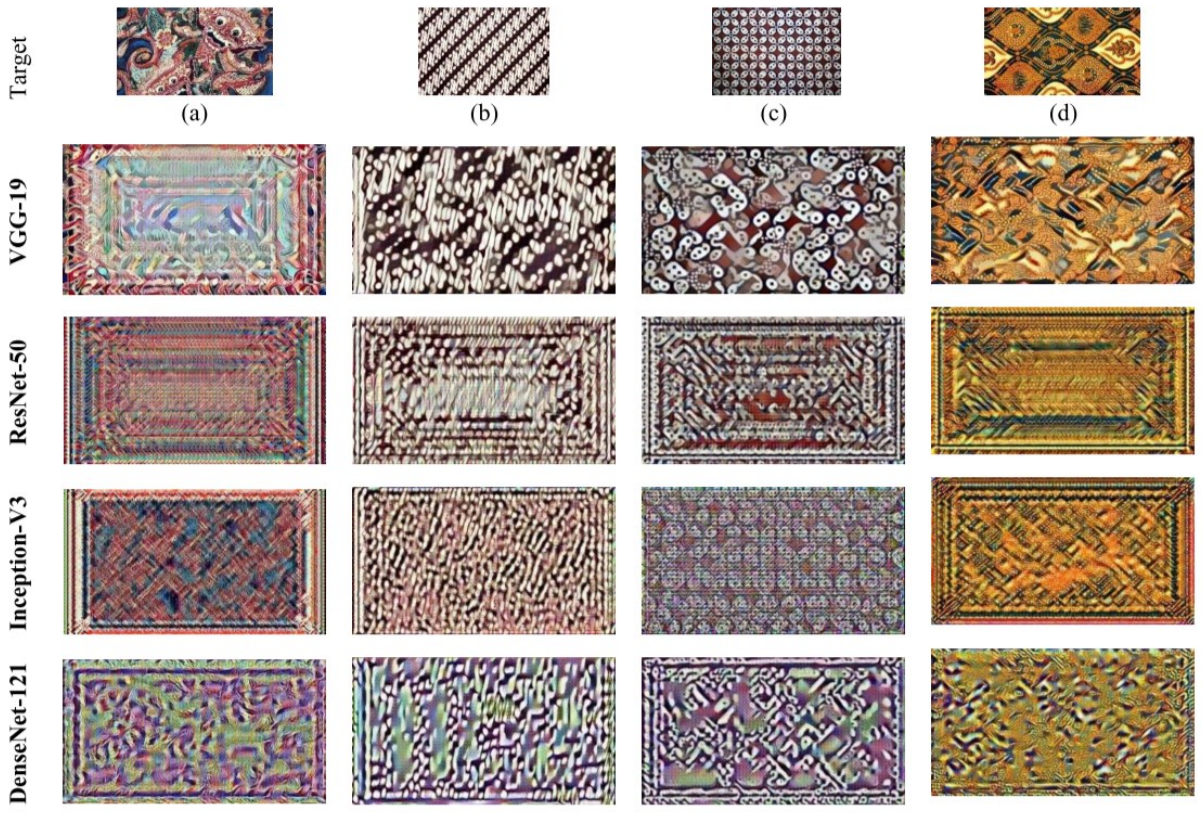


Figure 3 (a) Batik Bali; (b) Batik Parang; (c) batik Kawung; (d) Batik sidomukti

- Batik Textured image Experiment result

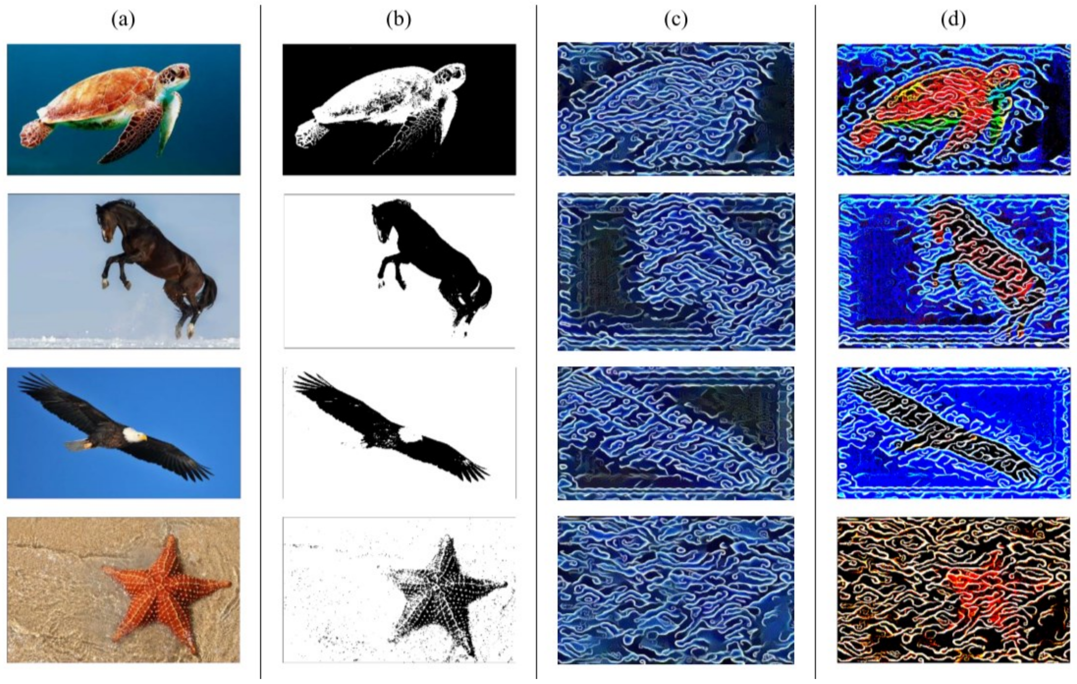


Figure 4 Images stylization

Based on figure 4 above, it can be seen that the batik style can be applied towards the various content images. But in this research the content image is **Carefully selected**. As the writer said before the noise **impact** the restoration and stylizing output image. By using the megamendung as a baseline for styling. We can see from **one to the third** row, the content image is preserved well, while the fourth is not.

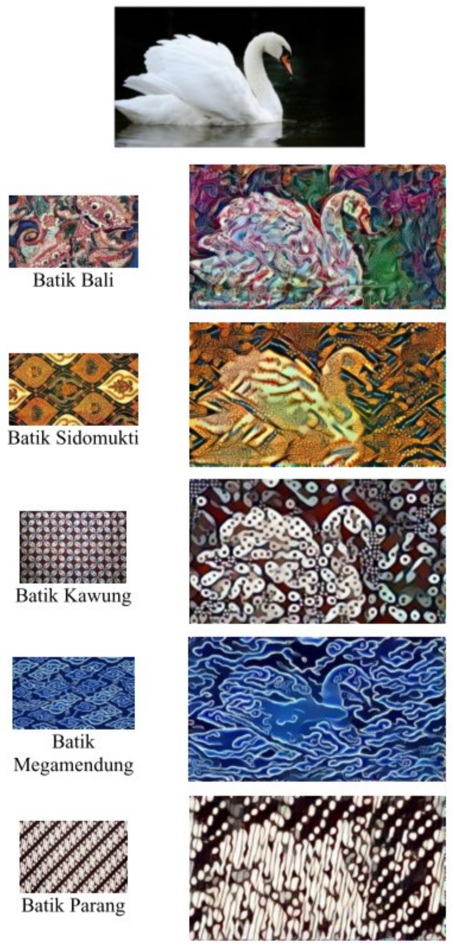


Figure 5 Stylization of a swan image

In this study case, we can see that **batik megamendung** has a better result compare with others. We also see the **inconsistency each results have**! Its mean that the algorithm is not yet able to know how **batik is truly generated**. The algorithm is also messy the **swan's shape and losing its structure.**

Paper Contributions :

- It gave a greate overview about how complex to generate a batik representation in computer vision field
- It can be used as a stepping stone to future research, such as
  - Shape generated batik
  - Batik shape consistency
- Preserving Indonesian culture

Paper Drawback/Limitation :

- Cannot generate **complex batik pattern**
    - Batik Sidomukti
    - Batik Bali
- The Result is nearly a batik but **the pattern is not consistent,** as the most batik should be
- The generated batik is still abstract **inside of the shape when style transfer take place**
- The Restoration Is not always consistent, it is based on the content image representation
    - If there are a lot of noises, such as the star image on figure … it make the restoration result not maximum.
- The Generated batik could not stylized only the content shape. I.e the black part only