Identification of Batik Motif Based Deep Learning-Convolutional Neural Network Approach

Ade Oktarino [a], Yanti Desnita Tasri [b,*] and Akmar Efendi [c]

[a] *Information Technology Department,Universitas Adiwangsa Jambi, Indonesia*
[b] *Health Informatics Department, STIKES Dharma Landbouw, Padang, Indonesia*
[c] *Informatics Technology Department,Universitas Islam Riau, Indonesia*

*Corresponding author: zyri_bkt@yahoo.com

Paper History

Received: 14-October-2024
Received in revised form: 24-November-2024
Accepted: 30-November-2024

ABSTRACT

Batik, a rich Indonesian cultural heritage, boasts a diverse array of motifs, each reflecting the unique philosophy of different regions. However, this diversity can make it challenging to distinguish between various batik patterns. This study aims to identify batik motifs using the Convolutional Neural Network (CNN) method. This research dataset comprises 521 digital batik images, encompassing five distinct motifs: *Betawi, Cendrawasih, Kawung, Megamendung*, and *Parang*. The data underwent a rigorous processing pipeline, including pre-processing, image segmentation, and feature extraction using Gray Level Co-occurrence Matrix (GLCM). Subsequently, a CNN model was employed for classification. The experimental results yielded an impressive average accuracy of 99.2% in identifying batik motifs. This outcome underscores the efficacy of deep learning, particularly CNNs, in recognizing and categorizing intricate batik patterns. This study may expect to serve a foundational step towards the development of advanced, automated batik recognition systems.

KEYWORDS: *Batik, Convolutional Neural Network, Motif Identification, Digital Image, Deep Learning.*

1.0 INTRODUCTION

Batik represents a significant aspect of the cultural heritage that is prevalent in Indonesia. Batik signifies an artistic form that is derived from Indonesia, recognized for its conventional production methodologies. Therefore, it is of utmost importance that it be consistently preserved and maintained [1]. Indonesia showcases an array of distinct batik motifs across its various regions, which often leads to considerable challenges for both local residents and tourists in discerning and recognizing these motifs. The proliferation of batik in Indonesia continues to evolve in contemporary times, encompassing not only traditional written batik but also digital batik (print). According to information disseminated on the national website kemenparekraf.go.id, there are presently approximately 5,849 batik motifs in Indonesia, spanning from Aceh to Papua. Batik is an integral component of Indonesian cultural heritage, characterized by a diverse array of unique elements that reflect botanical designs, fauna, and the quotidian experiences of individuals. The classifications of batik can be delineated into three categories, specifically batik stamp, print, and written [2-3]. The extensive variety of batik motifs complicates the task of differentiating between the designs of among batik. A significant number of modern Indonesians struggle to accurately identify the type or design of batik [4]. As articulated by [5], batik is a manifestation of Indonesian children's creativity, representing a synthesis of artistic and technological elements.

Given the multitude of batik motifs that currently exist, it is imperative to undertake the identification of batik. The identification of these batik motifs is essential for the purpose of categorizing or classifying the existing batik according to the motifs present. Each motif of batik displays a highly distinctive motif that recurs throughout the design. Such motifs may function as identifying features in determining the provenance of the batik fabric. Images exhibiting varied textures possess different characteristics. Pertaining to the imagery of batik, textural features constitute a critical aspect, as the ornaments on batik can be perceived as compositions of diverse textures. Moreover, batik motifs are also shaped by the characteristics of the forms that constitute each motif. For instance, a slope motif may exhibit a slash pattern, while a cleaver motif may appear oblique with indentations at each extremity, along with several other motifs. In light of the myriad of batik motifs that currently prevail in Indonesia, it is imperative to categorize these motifs. The categorization of batik motifs holds significant importance, as articulated by

[6], who posits that the creation process of batik motifs serves as a reflection of the geographical area. Furthermore, the development of these motifs is fundamentally influenced by the visual stimuli present in the surrounding environment [7]. Classification entails an analytical process that scrutinizes the image to yield a model delineating the categories inherent within that image.

A digital image can be understood as an image that is encoded in a digital format, characterized as a two-dimensional function that incorporates horizontal and vertical coordinates, with 'f' reflecting the intensity value or the level of darkness associated with the coordinates x and y. The image is represented as a two-dimensional function, as dictated by the computational hardware utilized (computer). The ostensibly three-dimensional image is subsequently rendered in a two-dimensional format, represented through discrete data points, owing to the limitations of the computer's capacity to process the authentic three-dimensional image. The data points referenced are also termed pixels [8-9]. Viewed mathematically, the image is deemed a continuous function of light intensity distributed across a two-dimensional surface. For the purpose of processing by a digital computer, an image must be numerically represented through discrete values. The conversion of continuous functions into discrete values is referred to as image digitization. A digital image can be conceptualized as a two-dimensional matrix f (x, y) consisting of M columns and N rows, wherein the intersection of columns and rows is identified as a pixel (pixel = picture element) or the fundamental component of an image [10].

Given the numerous varieties of batik motifs that currently exist in Indonesia, it is imperative to conduct a thorough identification of these motifs. The identification of batik motifs is of paramount importance, as articulated by [6], since the process of creating batik motifs serves as a reflection of a particular region. Moreover, according to [7], the process of motif formation is inherently impacted by the environmental context. Classification entails a systematic analytical process of imagery which yields a model that delineates the categories inherent within the image. One viable technique for identification can be achieved through the utilization of algorithms inherent in Deep Learning methodologies [11]. Deep Learning or deep structured learning or hierarchical learning or deep neural is a learning method that utilizes multiple non-linear transformations. Deep learning can be viewed as a combination of machine learning with AI (artificial neural network) [12-14]. Several algorithms are included in the Deep Learning category, including [15-16]: a. Convolutional Neural Networks, b. Restricted Boltzmann Machine (RBM), Deep Belief Networks (DBN), and d. Stacked Autocoder.

A prominent method for identification may be employed through algorithms utilized in Deep Learning frameworks. One type of deep learning model is Convolutional Neural Networks (CNNs). Convolutional Neural Networks (CNNs) signify a complex deep learning technique capable of recognizing and pinpointing objects in digital imagery. This efficacy is predominantly ascribed to enhancements in computational capabilities, the accessibility of extensive datasets, and the evolution of methodologies for the training of more profound network configurations.

Convolutional Neural Networks (CNNs) have emerged as a cornerstone in the field of computer vision, revolutionizing the way we process and interpret visual information. These powerful deep learning models are inspired by the structure and function of the human visual cortex, enabling them to automatically learn hierarchical representations of visual data. Convolutional Neural Networks (CNNs) are widely used for image classification, a task that involves categorizing images into predefined classes. These models have achieved impressive results in various applications, including facial recognition and scene classification. Classic CNN architectures like AlexNet and VGG have set new standards in image classification competitions, showcasing the power of CNNs in extracting meaningful features from images and making accurate predictions [17]. Image segmentation is another area where CNNs have made significant strides. This technique involves dividing an image into segments, each corresponding to a specific object or region of interest. CNN-based models, such as U-Net, have proven to be highly effective in medical image segmentation, enabling accurate diagnosis and treatment planning [18].

This study addresses two primary challenges in batik image classification. Firstly, it aims to develop a robust method for accurately identifying diverse batik motifs from a variety of image formats, including .JPG, .PNG, .TIFF, and .BMP. Secondly, it seeks to establish a comprehensive classification system for grouping these identified motifs into meaningful categories, thereby facilitating a deeper understanding of batik heritage. In other hand, this study aims to develop an effective methodology and concept for classifying batik images based on motifs, to assess the accuracy of motif classification, and to improve the speed of classifying newly inputted batik images. The primary objective of this research is to develop an effective methodology for classifying batik images based on their unique motifs. By achieving this goal, the study expects to deliver several significant benefits. Firstly, it will enable the precise identification of existing batik motifs, contributing to a better understanding of their origins and cultural significance. Secondly, the proposed classification system will provide more accurate and detailed information about the specific types of batik, aiding in their preservation, authentication, and commercialization.

2.0 METHODS

This research employs a qualitative methodology to identify batik motifs using Convolutional Neural Network (CNN). The study leverages Python programming language and Jupyter Notebook for data processing, enabling an interactive integration of code, equations, visualizations, and narrative text within a single document. A comprehensive model was developed to demonstrate the implementation of CNN for batik image detection. The research delves into the collection and analysis of batik motif patterns, identifying the appropriate data sources and types. The primary objective is to construct a model capable of recognizing and classifying batik motifs through image processing. The proposed model utilizes CNN, involving several stages to extract and analyze the distinctive features of batik patterns.

In the context of batik motif detection using Convolutional Neural Networks (CNN), the following outlines the sequential steps involved, from data collection to model deployment [17],[19-23].

- Dataset is the initial stage where a collection of images is gathered and organized.
- Pipeline and Augmentation are the dataset undergoes pre-processing and augmentation. Pre-processing involves tasks like resizing, normalization, and noise reduction. Augmentation techniques, such as rotation, flipping, and cropping, are applied to artificially increase the dataset size and improve model generalization.
- Data Splitting: The dataset is divided into three subsets: (a) training set is used to train the CNN model, (b) validation set is used to tune hyper-parameters and monitor model performance during training, (c) testing set is used to evaluate the final model's performance on unseen data.
- Data Transformation and Generator: Data transformation involves converting images into a suitable format for the CNN, such as tensors. A data generator is used to efficiently images feed batches to model during training.
- CNN Structure: This represents the architecture of the CNN, which typically consists of convolutional layers, pooling layers, and fully connected layers. Convolutional layers extract features from the input images, while pooling layers reduce dimensionality. Fully connected layers classify the features into different categories.
- Compile Model: In this stage, the CNN model is compiled with an optimizer (e.g., Adam, SGD) and a loss function (e.g., categorical cross-entropy).
- Training Model: A model is trained on training set using the specified optimizer and loss function. The number of epochs (iterations over the entire dataset) is set to 5.
- Model Evaluation: The trained model is evaluated on the validation or testing set to assess its performance using metrics like accuracy, precision, recall, and F1-score.
- Prediction Function**:** The final trained model can be used to predict the class of new, unseen images.

Data Collection

The first step involves gathering a comprehensive dataset of batik images, each annotated with labels that specify the corresponding motif. It is crucial to ensure that the dataset is diverse, encompassing various types of batik, color schemes, and background variations, to enhance the model's ability to generalize across different batik patterns. In this research employed a dataset comprising 521 batik motif images as a testbed for the study. This dataset was categorized into five distinct batik motifs: *Betawi, Cendrawasih, Kawung, Mega Mendung*, and *Parang*. The images were sourced from Google Images and be utilized for both training and testing the model. All images were in the JPEG format.

Data Preprocessing

Data preprocessing is a critical step in preparing the dataset for training a Convolutional Neural Network (CNN). It involves several key procedures to ensure the model's effectiveness and efficiency. First, image resizing standardizes all images to a consistent dimension, which is essential for ensuring uniform input sizes for the network. Second, data augmentation techniques, such as rotation, flipping, and cropping, are applied to artificially expand the dataset and enhance the model's ability to generalize by introducing variation in the training data. Finally, normalization is performed by scaling pixel values to a specific range, typically between 0 and 1, to optimize the performance of the CNN, ensuring faster convergence and more stable training. These preprocessing steps collectively improve the quality and robustness of the model’s learning process.

Model Design

The next phase is model design that plays a crucial role in the success of a Convolutional Neural Network (CNN) for batik motif detection, as selecting an appropriate architecture is essential for achieving optimal performance. The choice of architecture depends on the complexity of the batik motifs and the available computational resources. Commonly used CNN architectures, such as VGG and ResNet, are effective for extracting hierarchical features from images. The design process typically involves defining several key layers, convolutional layers, which are responsible for feature extraction; pooling layers, which reduce the spatial dimensions of the feature maps; and fully connected layers, which are used for classification based on the extracted features. This layered structure allows the model to learn both low- and high-level representations, ultimately enabling accurate motif classification.

Data Training

The model is then trained using the processed dataset. The model is trained using a dataset that is split into training, validation, and testing sets to prevent overfitting. During training, input images are fed into the CNN, and predictions are generated through forward propagation. The loss between the predicted and actual labels is calculated and used to update the network's weights via backpropagation. An optimization algorithm like Adam or SGD is employed to minimize the loss function and enhance the model's accuracy. In this study, three types of data are required to support the research process:

- The training data is used to train the model on batik motifs, consisting of 521 batik motif images. This dataset is crucial for teaching the model to recognize and classify different motifs during the training phase.
- The validation data is employed during the training process to assess the model's accuracy in real-time. It helps evaluate the model's performance and adjust the parameters to improve its predictive capabilities.
- The test data is used to evaluate the system's performance after it has been trained. This dataset is applied to test the model in the Jupyter Notebook environment to assess its ability to make accurate predictions when deployed.

Model Evaluation

Model evaluation is a crucial step in assessing the effectiveness of a trained Convolutional Neural Network (CNN) in detecting batik motifs. Following the training phase, the model’s performance is quantitatively evaluated using several key metrics, including accuracy, precision, recall, and F1-score. These metrics provide a comprehensive assessment of the model’s ability to correctly identify and classify batik motifs, with precision and recall offering insights into the trade-offs between false positives and false negatives. Additionally, visualization techniques, such as confusion matrices, are employed to further analyze the model's strengths and weaknesses, helping to identify specific areas where the model may require improvement or fine-tuning. This multi-faceted approach ensures a thorough evaluation of the model's performance.

Finally, fine-tuning is an essential process when a model's performance does not meet the desired standards. If the initial results are unsatisfactory, several strategies can be employed to improve the model. These strategies may include adjusting hyper parameters, such as learning rate, batch size, or regularization parameters, to optimize the training process. Additionally, experimenting with alternative architectures or more complex network structures can enhance the model's ability to capture intricate features of the batik motifs. Another approach involves expanding the dataset by including more diverse examples, which helps the model generalize better and perform more robustly across varied input patterns. Fine-tuning is an iterative process aimed at refining the model to achieve optimal performance.

3.0 RESULT AND DISCUSSION

3.1 Results

To prepare the images for subsequent analysis, a preprocessing step is applied to enhance image quality. Image segmentation using thresholding is then employed to isolate the batik motifs from the background. Feature extraction based on gray-level co-occurrence matrices is used to capture the spatial relationships between pixels within the motifs. These extracted features are then fed into a Convolutional Neural Network (CNN) to classify the batik patterns.

A. Dataset Preparation

The primary objective of this research is to evaluate the system's accuracy in detecting and classifying various batik motifs. The dataset, primarily composed of JPEG images obtained from Google Images, was organized into 20 distinct folders. A total of 521 batik motif images were included, categorized into five primary types: *Betawi, Cendrawasih, Mega Mendung, Kawung,* and *Parang*. The Batik motif image dataset is depicted in Figure 1.

B. Preparing the dataset pipeline and augmentation for the dataset creation process

The dataset pipeline systematically fetches image data from a specified directory. This data undergoes preprocessing steps, such as resizing and normalization, before being converted into a multidimensional array. This array, in a format suitable for Tensorflow, serves as input to the deep learning model. The function used can be seen in Figure 2.

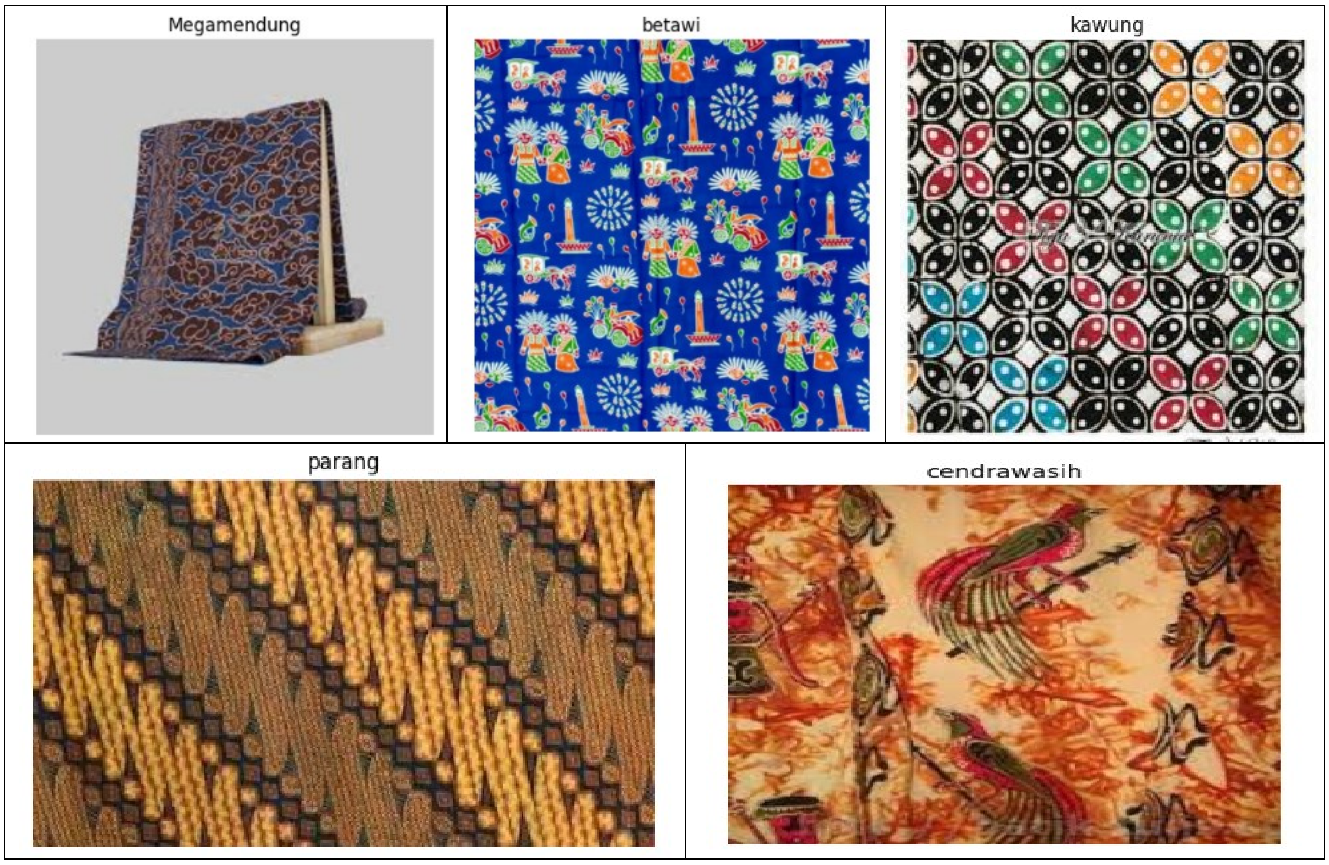
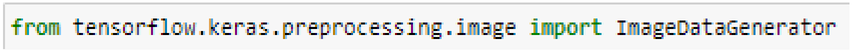
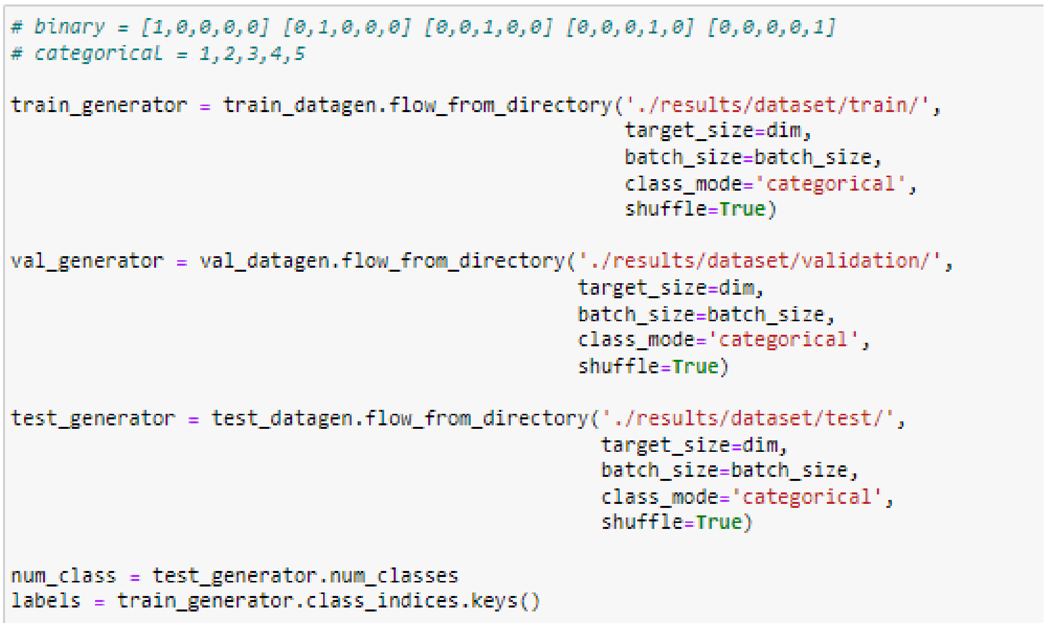


Figure 1: Batik motif image dataset


```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```
Figure 2: Tensorflow function code in python



```
# binary = [1,0,0,0,0] [0,1,0,0,0] [0,0,1,0,0] [0,0,0,1,0] [0,0,0,0,1]
# categorical = 1,2,3,4,5

train_generator = train_datagen.flow_from_directory('./results/dataset/train/',
                                                    target_size=dim,
                                                    batch_size=batch_size,
                                                    class_mode='categorical',
                                                    shuffle=True)

val_generator = val_datagen.flow_from_directory('./results/dataset/validation/',
                                                target_size=dim,
                                                batch_size=batch_size,
                                                class_mode='categorical',
                                                shuffle=True)

test_generator = test_datagen.flow_from_directory('./results/dataset/test/',
                                                  target_size=dim,
                                                  batch_size=batch_size,
                                                  class_mode='categorical',
                                                  shuffle=True)

num_class = test_generator.num_classes
labels = train_generator.class_indices.keys()
```

Figure 3: Code for creating data flow

The function is an Image data generator so that the next process generates image data from a file / folder that has been created previously. In this section the process can determine what kind of generator / augmentation can be done.

C. Create the data flow

After defining the generator, the next step is defining where the data source comes from. For example, in this study, data from a folder (dictionary) is used so that the code used can be seen in Figure 3. In Figure 3, using the flow from "dir" where the function directs to the previously created set folder. Where:
target size = dimensions of the image to be used in the training process
batch size = number of images entered in each training step.
class mode = classification selection method
shuffle = data in the folder is shuffled so that it does not match the existing order such as alphabetical order.

D. Transform generator data into tf.data

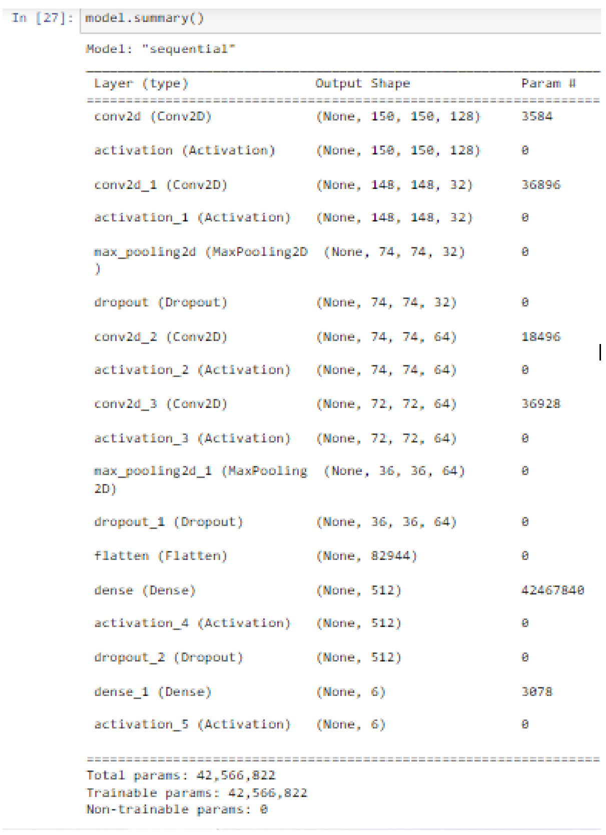
Training deep learning models often involves working with large datasets. Efficient data handling becomes crucial for optimal performance. This section explores the rationale behind transforming data generators into tf.data objects for Tensorflow training benefit of using tf.data lies in its ability to significantly improve data throughput compared to traditional hard generators. This is particularly advantageous when dealing with large datasets. The tf.data facilitates a more optimized data pipeline, allowing the model to process information faster during training. This can lead to quicker training times and overall improved efficiency.

E. Creating a Convolutional Neural Network Structure

During the process of constructing the image structure, several Convolutional layers are applied using MobileNet V2. In the first convolutional layer, a 128-kernel filter was used to identify parameters, resulting in a total of 3,584 parameters. Next, in the second convolutional layer, a 32-kernel filter was employed, allowing the model to capture 36,896 parameters. In the third convolutional layer, a 64-kernel filter was utilized to extract 18,496 parameters. Finally, in the fourth convolutional layer, a 64-kernel filter was again applied to capture 36,928 parameters. These layers progressively refine the feature extraction process, contributing to the model's ability to recognize and classify batik motifs effectively.
Figure 4 presents the output of each layer after training on the dataset. The hidden layer yielded an output of 42,566,822. The model was trained for 5 epochs, and the resulting model was then saved for subsequent testing. During each epoch, the model was exposed to the entire training dataset, allowing it to adjust its parameters and improve its performance. Once the training process was completed, the final model was saved. This saved model can be used for future predictions or further fine-tuning.

F. Compile model

The model was compiled to prepare it for the training process. This step involves defining the optimization algorithm, loss function, and evaluation metrics. An optimization algorithm, the optimizer's goal was to minimize the loss function, which measures the discrepancy between the model's predictions and the true labels. Loss Function, in this case, categorical cross-entropy was employed as the loss function. This is appropriate for classification tasks where the output is a probability distribution over multiple classes. The loss function calculates the difference between the predicted probability distribution and the true distribution, guiding the model to adjust its parameters to minimize this discrepancy. Finally, the evaluation metric to assess the model's performance during training and testing, accuracy was used as the primary evaluation metric. Accuracy measures the proportion of correctly classified samples. By monitoring the accuracy, it can gauge the model's ability to make accurate predictions.



```
In [27]: model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 128)     3584

 activation (Activation)     (None, 150, 150, 128)     0

 conv2d_1 (Conv2D)           (None, 148, 148, 32)      36896

 activation_1 (Activation)   (None, 148, 148, 32)      0

 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0
 )

 dropout (Dropout)           (None, 74, 74, 32)        0

 conv2d_2 (Conv2D)           (None, 74, 74, 64)        18496

 activation_2 (Activation)   (None, 74, 74, 64)        0

 conv2d_3 (Conv2D)           (None, 72, 72, 64)        36928

 activation_3 (Activation)   (None, 72, 72, 64)        0

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 36, 36, 64)        0

 flatten (Flatten)           (None, 82944)             0

 dense (Dense)               (None, 512)               42467840

 activation_4 (Activation)   (None, 512)               0

 dropout_2 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 6)                 3078

 activation_5 (Activation)   (None, 6)                 0

=================================================================
Total params: 42,566,822
Trainable params: 42,566,822
Non-trainable params: 0
```

Figure 4: Summary table/ sequential/ training results for each layer

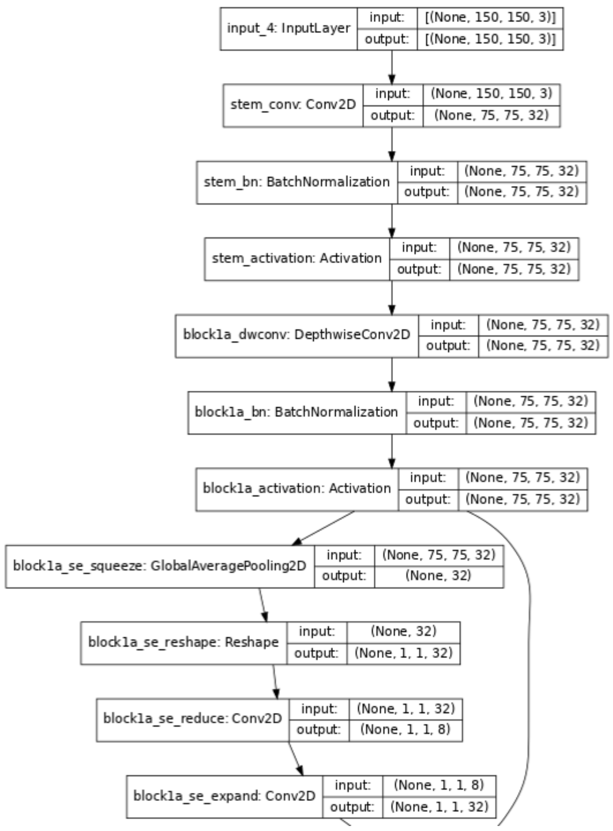
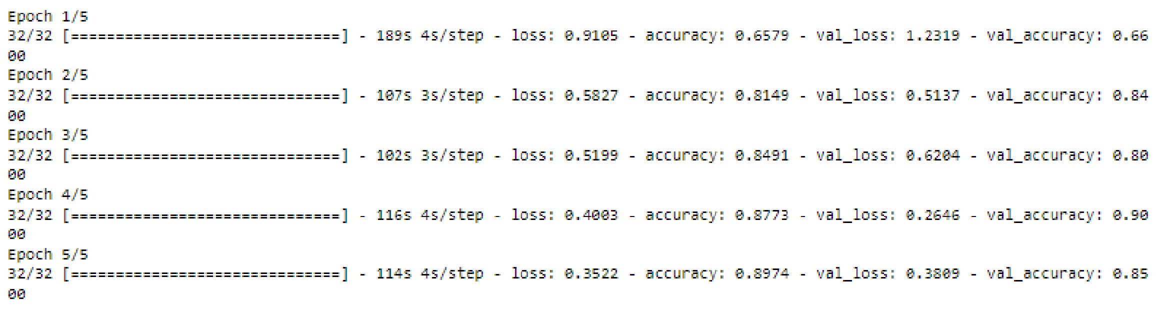

input_4: InputLayer
input: [(None, 150, 150, 3)]
output: [(None, 150, 150, 3)]
stem_conv: Conv2D
input: (None, 150, 150, 3)
output: (None, 75, 75, 32)
stem_bn: BatchNormalization
input: (None, 75, 75, 32)
output: (None, 75, 75, 32)
stem_activation: Activation
input: (None, 75, 75, 32)
output: (None, 75, 75, 32)
block1a_dwconv: DepthwiseConv2D
input: (None, 75, 75, 32)
output: (None, 75, 75, 32)
block1a_bn: BatchNormalization
input: (None, 75, 75, 32)
output: (None, 75, 75, 32)
block1a_activation: Activation
input: (None, 75, 75, 32)
output: (None, 75, 75, 32)
block1a_se_squeeze: GlobalAveragePooling2D
input: (None, 75, 75, 32)
output: (None, 32)
block1a_se_reshape: Reshape
input: (None, 32)
output: (None, 1, 1, 32)
block1a_se_reduce: Conv2D
input: (None, 1, 1, 32)
output: (None, 1, 1, 8)
block1a_se_expand: Conv2D
input: (None, 1, 1, 8)
output: (None, 1, 1, 32)

Figure 5: Model Visualization



```
Epoch 1/5
32/32 [==============================] - 189s 4s/step - loss: 0.9105 - accuracy: 0.6579 - val_loss: 1.2319 - val_accuracy: 0.66
00
Epoch 2/5
32/32 [==============================] - 107s 3s/step - loss: 0.5827 - accuracy: 0.8149 - val_loss: 0.5137 - val_accuracy: 0.84
00
Epoch 3/5
32/32 [==============================] - 102s 3s/step - loss: 0.5199 - accuracy: 0.8491 - val_loss: 0.6204 - val_accuracy: 0.80
00
Epoch 4/5
32/32 [==============================] - 116s 4s/step - loss: 0.4003 - accuracy: 0.8773 - val_loss: 0.2646 - val_accuracy: 0.90
00
Epoch 5/5
32/32 [==============================] - 114s 4s/step - loss: 0.3522 - accuracy: 0.8974 - val_loss: 0.3809 - val_accuracy: 0.85
00
```

Figure 6: Training process with epoch 5

G. Model Visualization

Figure 5 provides a visual representation of the model's internal workings. It allows explores the learned representations and decision-making processes within the neural network. By visualizing the model's outputs, it can gain deeper insights into how it arrives at its predictions. This can help to understand the model's strengths and limitations, potentially leading to improved performance and more reliable decision-making.

H. Process of Training Model

The next process is to carry out the training process, where this training process is a process where deep learning works so that the algorithm that we have defined can remember the patterns of each class in the trained data. The training process uses epoch 5. The results of the training process with epoch 5 are presented in Figure 6.

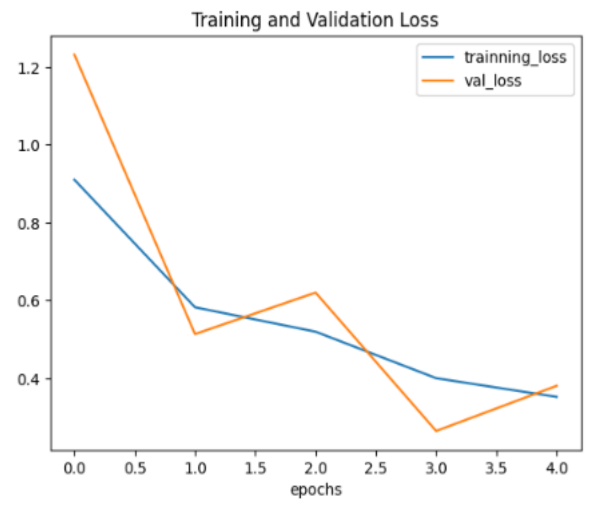
Description:

- X = is the generator data that have processed so that it becomes tf-data, this is training data.
- Steps per epoch is the number of steps / steps to complete 1 epoch, here 1 step is 1 batch_size / 1 batch_size is 16 images, as previously defined.
- epoch = the number of iterations / repetitions in training
- val_data = validation data that is evaluated at the end of each epoch
- val_steps = the same as steps per epoch

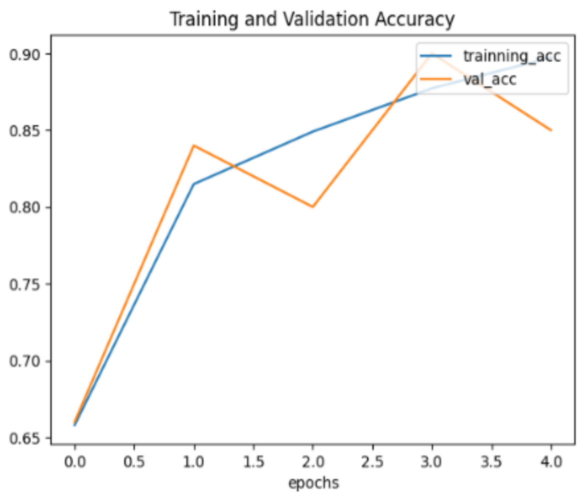
- shuffle = is where each batch in the generator is shuffled so that the data will not be in order.
- Verbose = displays a progress bar where 1, because it can see the process and results of each step as seen in Figure 7.

G. Model evaluation using testing data

At the model evaluation stage, testing was carried out using test data. Where, this test data was not included in the training process. The results of the model evaluation using testing data can be seen in Figure 8. Figure 8 demonstrates the model's stability with training accuracy of 91% and training loss of 0.3. The corresponding test accuracy and loss were 88% and 0.5, respectively. The results visualized in Figure 8 indicate that the model has achieved a reasonable level of generalization. The training accuracy of 91% suggests that the model has effectively learned the underlying patterns in the training data, while the test accuracy of 88% demonstrates its ability to generalize to unseen data.

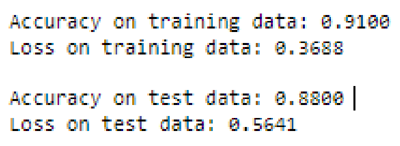


(a)



(b)

Figure 7: Training process with epoch 5, (a) validation loss, (b) validation accuracy



```
Accuracy on training data: 0.9100
Loss on training data: 0.3688

Accuracy on test data: 0.8800
Loss on test data: 0.5641
```

Figure 8: Results of model evaluation using testing data

3.2 Discussion

A Convolutional Neural Network (CNN) model was employed to identify and classify batik motifs. This involved a series of steps, including image training, testing, and processing. The results of the model's performance are presented in Table 1. Table 1 showcases the accuracy of the CNN model in classifying five distinct batik motifs: *Betawi, Megamendung, Kawung, Parang*, and *Cendrawasih*. The model demonstrated exceptional performance, achieving accuracies of 99%, 100%, 99%, 99%, and 99%, respectively.

Inference times were generally fast, ranging from 124ms/step to 383ms/step. Overall, the model achieved an impressive average accuracy of 99.2%. The model's ability to learn complex visual patterns and generalize to unseen data highlights the potential of deep learning techniques for cultural heritage preservation and pattern recognition tasks. The high accuracy and efficiency of the model suggest its potential for practical applications. By accurately distinguishing between different batik motifs, this model could be utilized for image search, pattern recognition, and the preservation of cultural heritage.

The ability to rapidly classify batik motifs can aid in tasks such as authenticating textiles, identifying historical patterns, and categorizing large collections of batik images. However, the validation results revealed some inconsistencies, particularly in terms of loss and accuracy. These findings suggest that the model may benefit from additional training or fine-tuning. While the model achieved a reasonable level of performance on the training set, its generalization to unseen data was somewhat limited. Future work could explore techniques such as regularization, data augmentation, or early stopping to improve the model's robustness and prevent over-fitting.

5.0 CONCLUSION

The study successfully employed a Convolutional Neural Network (CNN) to classify batik motifs. The model achieved an average accuracy of 99.2% across five different batik categories, indicating its strong performance. The implementation results revealed that the proposed CNN model was highly effective in classifying batik motifs. The model achieved a training accuracy of 91% with a corresponding training loss of 0.3, demonstrating its ability to learn the underlying patterns in the training data. When evaluated on a held-out test set, the model exhibited a slightly lower but still commendable accuracy of 88% with a test loss of 0.5. The high accuracy and relatively fast inference times suggest that this approach could be effectively applied to real-world applications. The validation loss exhibited instability, indicating that the model may be over-fitting to the training data. To mitigate this issue, further experimentation with a larger number of epochs could be considered. The model's performance, as evidenced by the fluctuating validation loss and accuracy, suggests that it may not have fully captured the underlying patterns in the data. This instability could be attributed to over-fitting. To address these limitations, further investigation is warranted. This may involve experimenting with different hyper-parameters, such as increasing the number of epochs, or exploring alternative deep learning architectures.

Table 1: The CNN model testing results

No	Image	Motif Types	Time	Accuracy
1		Betawi	0s 132ms /step	99%
2		Mega mendung	0s 141ms /step	100%
3		Kawung	0s 124ms /step	99%
4		Parang	0s 383ms /step	99%
5		Cendra-wasih	0s 142ms /step	99%
	Average accuracy			99.2%

REFERENCES

[1] Handhayani, T., Hendryli, J. & Hiryanto, L. (2017, November). Comparison of shallow and deep learning models for classification of Lasem batik patterns. In *2017 1st International Conference on Informatics and Computational Sciences (ICICoS)* (pp. 11-16). IEEE.

[2] Rosyada, M. & Wigiawati, A. (2020). Strategi survival UMKM Batik Tulis Pekalongan di tengah pandemi COVID-19 (studi kasus pada "Batik Pesisir" Pekalongan). *Jurnal Bisnis dan Kajian Strategi Manajemen*, *4*(2).

[3] Setiawan, J., Atika, V., Pujilestari, T., & Haerudin, A. (2018). Kesesuaian Batik Tulis Ikm Berdasarkan Sni 08-0513-1989. *Jurnal Standardisasi*, *20*(1), 69.

[4] Akbar, M.D., Martanto, M. & Wijaya, Y.A. (2022). Klasifikasi Motif Batik Jawa Menggunakan Algoritma K-Nearest Neighbors (Knn). *Jurnal Sistem Informasi dan Manajemen*, *JURSIMA*, *10*(2), 161-168.

[5] Hakim, L., Kristanto, S.P., Yusuf, D. & Afia, F.N. (2022). Pengenalan Motif Batik Banyuwangi Berdasarkan Fitur Grey Level Co-Occurrence Matrix. *Jurnal Teknoinfo*, *16*(1), 1-7.

[6] Fajar, R.M., Mulyono, H. & Adi, F.P. (2021). Identifikasi Nilai Karakter Motif Batik Ngawi Berbasis Budaya Lokal sebagai Muatan Pendidikan Seni Rupa di Sekolah Dasar. *Jurnal basicedu*, *5*(2), 571-580.

[7] Wijaya, Y.A. (2021). Analisa Klasifikasi menggunakan Algoritma Decision Tree pada Data Log Firewall Jurnal Sistem Informasi dan Manajemen. *Jurnal Sistem Informasi dan Manajemen (JURSIMA)*, *9*(3), 256-264.

[8] Hartono, S., Perwitasari, A. & Sujaini, H. (2020). Komparasi Algoritma Nonparametrik untuk Klasifikasi Citra Wajah Berdasarkan Suku di Indonesia. *JEPIN (Jurnal Edukasi dan Penelitian Informatika)*, *6*(3), 337-343.

[9] Tasri, Y. (2022). Identification of Signature Images with Edge Detection Canny. *Journal Of Ocean, Mechanical And Aerospace -Science And Engineering-*, *66*(3), 89-93. doi:10.36842/jomase.v66i3.316.

[10] Ryansyah, R. (2021). Identifikasi tingkatan warna pada kopi roasting menggunakan metode HSV berbasis mobile. *Jurnal Terapan Informatika Nusantara*, *1*(10), 520–526.

[11] Maulana, F.F. & Rochmawati, N. (2019). Klasifikasi citra buah menggunakan convolutional neural network. *Journal of Informatics and Computer Science (JINACS)*, *1*(02), 104-108.

[12] Juliansyah, S. & Laksito, A.D. (2021). Klasifikasi Citra Buah Pir Menggunakan Convolutional Neural Networks. *InComTech: Jurnal Telekomunikasi dan Komputer*, *11*(1), 65-72.

[13] Diana, R., Warni, H. & Sutabri, T. (2023). penggunaan teknologi machine learning untuk pelayanan monitoring kegiatan belajar mengajar pada smk bina sriwijaya palembang. *Jurnal Teknik Informatika (JUTEKIN)*, *11*(1).

[14] Wijanarko, E.W.S. & Adhisa, R.R. (2023). Media pembelajaran object detection perangkat jaringan komputer menggunakan machine learning berbasis

desktop. *Edumatic: Jurnal Pendidikan Informatika*, *7*(2), 207-216.

[15] Nugroho, P.A., Fenriana, I., & Arijanto, R. (2020). Implementasi deep learning menggunakan convolutional neural network (CNN) pada ekspresi manusia. *Algor*, *2*(1), 12-20.

[16] Pulung Nurtantio Andono, N.P. & Rachmawanto, H.E. (2020). Evaluasi ekstraksi fitur GLCM dan LBP menggunakan multi-kernel SVM untuk klasifikasi batik. *Jurnal Rekayasa Sistem dan Teknologi Informasi*, *5*(1), 1–9.

[17] Jiang, J., Zhang, Y., Xie, H., Yang, J., Gong, J. & Li, Z. (2024). A deep learning based fine-grained classification algorithm for grading of visual impairment in cataract patients. *Optoelectronics Letters*, *20*(1), 48-57.

[18] Younesi, A., Ansari, M., Fazli, M. A., Ejlali, A., Shafique, M. & Henkel, J. (2024). A comprehensive survey of convolutions in deep learning: Applications, challenges, and future trends. *IEEE Access*. https://doi.org/10.1109/access.2024.3376441.

[19] Jiamin, J. (2024). The eye of artificial intelligence - Convolutional neural networks. *Applied and Computational Engineering*. 76,273-279. https://doi.org/10.54254/2755-2721/76/20240613.

[20] Junkun, Y. (2024). Application of CNN in computer vision. *Applied and Computational Engineering*, 30,104-110. https://doi.org/10.54254/2755-2721/30/20230081.

[21] Xia, Z., Zhang, Y., Han, X., Deveci, M. & Parmar, M. (2024). A review of convolutional neural networks in computer vision. *Artificial Intelligence Review*, 57, 99. https://doi.org/10.1007/s10462-024-10721-6.

[22] Krichen, M. (2023). Convolutional neural networks: A survey. *Computers*. 12(8), 151. https://doi.org/10.3390/computers12080151.

[23] Mustapha, M.T., Ozsahin, I. & Ozsahin, D.U. (2024). Convolution neural network and deep learning. In *Artificial Intelligence and Image Processing in Medical Imaging* (pp. 21-50). https://doi.org/10.1016/b978-0-323-95462-4.00002-9.