J-KOMA Journal of Computer Science and Applications

Universitas Negeri Jakarta
Phone: (021) 4890856
East Jakarta City
Jakarta, 13220

Open Access

E-ISSN: *2620 - 4872*
Vol.*07*, ***No***.*02*
Doi: *https://doi.org/10.21009/j-koma.v7i2.02*

Recivied: *02 Okt 2024*
Accepted: *15 Okt 2024*
Published: *20 Des 2024*

Keywords:
Yogyakarta Batik;
Gray Level Co-occurrence Matrix;
K-Nearest Neighbor;
Streamlit;
Image Classification;

Correspondence Email:
qurania@unpak.ac.id

Classification of Yogyakarta Batik Using the K-Nearest Neighbor (KNN) and Gray Level Co-occurrence Matrix (GLCM) Methods

Arie Qur'ania[1], Ananda Dias Saharani[2], Riri Handini[3]
[1,2,3]Computer Science Program Study,Universitas Pakuan, Indonesia (8pt)

Abstract
The preservation of Yogyakarta batik motifs as part of Indonesia's cultural heritage can be supported through digital image classification technology. This study aims to develop an automatic classification system for Yogyakarta batik motifs using the Gray Level Co-occurrence Matrix (GLCM) method for texture feature extraction and the K-Nearest Neighbor (KNN) algorithm for the classification process. The dataset consists of 1,350 digital images of six different batik motif types, sourced from Kaggle. The system was developed and tested on the Google Colab platform through several stages, including preprocessing, feature extraction, model training, and performance evaluation using accuracy, precision, recall, and F1-score metrics. The evaluation results show that the model achieved an accuracy of 60%, with the best performance on the batik-ceplok motif (F1-score of 77%) and the lowest on the batik-kawung motif (F1-score of 46%). The system was then implemented as a web application using the Streamlit framework, allowing users to upload images and receive classification results in real time. This implementation not only contributes to the field of image processing but also aids in cultural preservation through digitization and easy access to batik motif classification.

INTRODUCTION

Batik is an Indonesian cultural heritage that has been recognized by UNESCO as a world heritage site. The diversity of Indonesian batik motifs reflects its richness, varying from one region to another. Each motif presents a unique design while embodying specific meanings and ideologies. This variety is shaped by the values and cultural characteristics it represents [1]. Yogyakarta is renowned for its batik, featuring distinct patterns primarily in white, black, and brown hues [2]. Some famous motifs from Yogyakarta include Batik Ceplok, Kawung, Nitik, Parang, Sidoluhur, and Truntum [3].

Preserving Yogyakarta batik designs is crucial to safeguarding this important cultural heritage from extinction. One approach is to identify and document batik patterns using digital technology [4]. The use of technology enables automated image classification of batik motifs through the application of classification algorithms and texture feature extraction methods [5]. One such method is the Gray-Level Co-occurrence Matrix (GLCM), which extracts texture features based on the distribution of gray levels between pixels [6]. The K-Nearest Neighbor (KNN) algorithm is chosen for the classification stage due to its capability to assess similarity between training and testing data by calculating feature distances [7].

This classification system is developed using the Streamlit framework, which facilitates the creation of interactive web applications, allowing users to perform batik motif classification directly. A key advantage of Streamlit is its ability to present data in an interactive and appealing manner [8]. Streamlit offers a variety of features to build application interfaces, with several components accompanied by pre-written code. It provides a solution for users who may have limited technological knowledge, simplifying the addition of code to web applications without requiring specialized skills or front-end development experience [9].

Universitas Negeri Jakarta
Phone: (021) 4890856
East Jakarta City
Jakarta, 13220

Previous research has demonstrated significant advancements in image classification. Prayoga et al. (2023) tested seven different CNN architectures to recognize Yogyakarta batik motifs such as Ceplok, Kawung, and Parang, achieving their best model accuracy of 87.83% [3]. Meanwhile, Nurhalimah et al. (2020) developed a classification system for Lombok songket fabric by combining GLCM texture feature extraction with Moment Invariant shape features, classified using the LDA method, reaching an accuracy of up to 98.33% under optimal conditions [10]. Furthermore, recent research by Suciani et al. (2024) classified batik types from Lasem, Sekar Jagad, Tambal, and Truntum using the KNN algorithm, achieving a highest accuracy of 85% with a 90% training and 10% testing data split [11].

Based on this background, the present study aims to implement the GLCM method for texture feature extraction and the KNN algorithm for classifying Yogyakarta batik motifs, as well as to develop an interactive web application using Streamlit to facilitate easier access to classification. Through this approach, the study is expected to contribute not only to the fields of information technology and computer vision but also to the preservation of local culture through digitalization.

METHODS

The research method provides an overview of the steps taken to address the research problem. Each step is designed with the objective of accomplishing its specific task. The steps involved in conducting this research are as follows.

Data Collection

The initial stage of this research is data collection, where the dataset used is secondary data obtained entirely from the Kaggle platform. The data consists of 1,350 batik motif images categorized into six types of Yogyakarta batik, namely Batik Ceplok, Batik Kawung, Batik Truntum, Batik Parang, Batik Sidoluhur, and Batik Nitik.

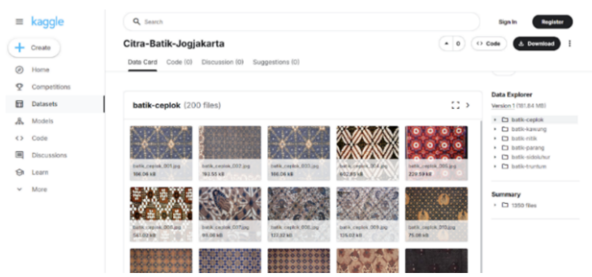
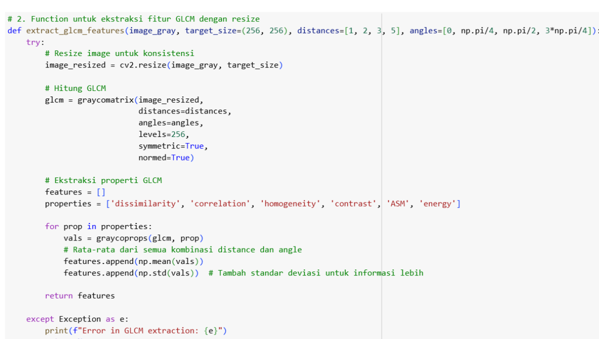


Figure 1. Yogyakarta Batik Image Data on the Kaggle Site

System Design



```python
# 2. Function untuk ekstraksi fitur GLCM dengan resize
def extract_glcm_features(image_gray, target_size=(256, 256), distances=[1, 2, 3, 5], angles=[0, np.pi/4, np.pi/2, 3*np.pi/4]):
    try:
        # Resize image untuk konsistensi
        image_resized = cv2.resize(image_gray, target_size)

        # Hitung GLCM
        glcm = graycomatrix(image_resized,
                            distances=distances,
                            angles=angles,
                            levels=256,
                            symmetric=True,
                            normed=True)

        # Ekstraksi properti GLCM
        features = []
        properties = ['dissimilarity', 'correlation', 'homogeneity', 'contrast', 'ASM', 'energy']

        for prop in properties:
            vals = graycoprops(glcm, prop)
            # Rata-rata dari semua kombinasi distance dan angle
            features.append(np.mean(vals))
            features.append(np.std(vals))  # Tambah standar deviasi untuk informasi lebih

        return features

    except Exception as e:
        print(f"Error in GLCM extraction: {e}")
        return None
```

Figure 2. GLCM Feature Extraction Function in Google Colab

System design begins with modeling features using the Gray Level Co-occurrence Matrix (GLCM) on Google Colab. The function extract_glcm_features analyzes image texture using four main parameters: the grayscale image, target image size (default 256×256), pixel distance, and analysis angles (0°, 45°, 90°, 135°). The process starts by resizing the image for consistency, followed by calculating the GLCM with 256 intensity levels, a symmetric matrix, and probability normalization. From the GLCM, six texture properties are extracted: dissimilarity, correlation, homogeneity, contrast, ASM (Angular Second Moment), and energy.

Universitas Negeri Jakarta
Phone: (021) 4890856
East Jakarta City
Jakarta, 13220

For each property, both the mean and standard deviation are computed, resulting in a total of 12 features that enrich the image texture characterization and support classification based on texture patterns.

```
[ ] # 6. Split data
    print("\n=== Splitting Data ===")
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y
    )
    print(f"Training samples: {len(X_train)}")
    print(f"Testing samples: {len(X_test)}")
```

Figure 3. Separating Data in Google Colab

After the GLCM modeling is completed, the next step involves modeling using the K-Nearest Neighbors (KNN) algorithm. In this modeling process, the data is first split using the train_test_split function with a test_size parameter of 0.2, where 80% of the data is allocated for training and the remaining 20% for testing. The use of the random_state parameter set to 42 ensures consistent data splitting results across multiple runs of the algorithm, while the stratify parameter set to the target variable y maintains the class distribution balance in both training and testing datasets.

```
# 7. Feature Normalization
print("\n=== Feature Normalization ===")
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
print("Features normalized using StandardScaler")
```

```
=== Feature Normalization ===
Features normalized using StandardScaler
```

Figure 4. Data Normalization in Google Colab

At this stage, StandardScaler is applied to standardize the feature data so that each variable has a mean of zero and a standard deviation of one. This process is performed using the fit_transform() method on the training data and the transform() method on the testing data, thereby preventing potential data leakage. The purpose of this step is to avoid features with large values dominating the algorithm’s process, enabling the model to produce more accurate predictions.

```
# 8. Parameter Tuning dengan Grid Search
print("\n=== Parameter Tuning ===")
param_grid = {
    'n_neighbors': [3, 5],
    'metric': ['euclidean', 'manhattan', 'minkowski'],
    'weights': ['uniform', 'distance']
}

knn = KNeighborsClassifier()
grid_search = GridSearchCV(
    knn, param_grid,
    cv=5,
    scoring='accuracy',
    n_jobs=-1,
    verbose=1
)

print("Running Grid Search...")
grid_search.fit(X_train_scaled, y_train)

print(f"Best parameters: {grid_search.best_params_}")
print(f"Best cross-validation score: {grid_search.best_score_:.4f}")
```

Figure 5. Parameter Setup in Google Colab

At this stage, the hyperparameters of the KNN algorithm are optimized using Grid Search Cross-Validation. Various parameter combinations, such as n_neighbors (3, 5), metrics (Euclidean, Manhattan, Minkowski), and weights (uniform, distance), are systematically evaluated. GridSearchCV assesses all these combinations using 5-fold cross-validation to identify the parameters that yield the highest accuracy. Through this process, the most optimal hyperparameter configuration is obtained, resulting in a KNN model with the best performance.

```
# 9. Training dengan best model
print("\n=== Training Best Model ===")
best_knn = grid_search.best_estimator_
print(f"Training KNN with parameters: {best_knn.get_params()}")

# Cross-validation score
cv_scores = cross_val_score(best_knn, X_train_scaled, y_train, cv=5)
print(f"Cross-validation scores: {cv_scores}")
print(f"Mean CV score: {cv_scores.mean():.4f} (+/- {cv_scores.std() * 2:.4f})")
```

Figure 6. KNN Modeling in Google Colab

At this stage, the best KNN model obtained through hyperparameter tuning is trained and validated. The optimal model resulting from the Grid Search is stored in the variable best_knn, and its best parameters are displayed using best_knn.get_params(). Subsequently, validation is performed using cross-validation on the normalized training data (X_train_scaled, y_train). This method partitions the training data into multiple folds to assess the consistency of the model's performance. The validation results are presented as cv_scores, which contain the accuracy scores for each fold.

```
# 10. Prediction dan Evaluasi
print("\n=== Model Evaluation ===")
y_pred = best_knn.predict(X_test_scaled)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Test Accuracy: {accuracy:.4f}")

# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

Figure 7. KNN Model Evaluation on Google Colab

After the modeling process is completed, the next step is to evaluate the performance of the best KNN model using the test data, which was not utilized during training. Predictions are generated using best_knn.predict(X_test_scaled), where X_test_scaled is the normalized test data to ensure consistency with the training data. The prediction results are stored in the variable y_pred. Initial evaluation is conducted using accuracy_score(y_test, y_pred) to calculate the accuracy rate, defined as the ratio of correct predictions to the total number of predictions. For a more comprehensive analysis, classification_report(y_test, y_pred), is employed, which presents precision, recall, and F1-score metrics for each class. Precision measures the accuracy of positive predictions, recall assesses the model's ability to identify actual positive instances, and the F1-score is the harmonic mean of these two metrics. This report is valuable for detecting biases or performance gaps between different classes.

System Implementation

After applying the GLCM and KNN techniques in Google Colab, the system is then integrated into a web application built with Streamlit. The implementation stages include model creation, image processing, feature extraction, and interactive presentation of prediction results. The core phases of this implementation involve preparing the model in Google Colab and developing the application using Streamlit.

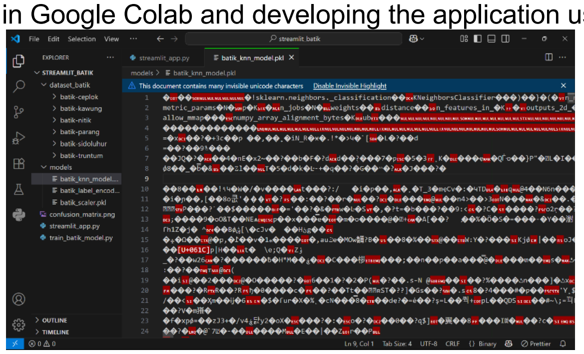


Figure 8. GLCM and KNN Modeling Display with .pkl File Format

The first stage of system implementation begins with model development using Google Colab. The batik classification model, developed with GLCM and KNN techniques in Google Colab, is exported to the .pkl format via Joblib and subsequently stored in the Streamlit project directory for use in the subsequent classification process.

J-KOMA Journal of Computer Science and Applications
Universitas Negeri Jakarta
Phone: (021) 4890856
East Jakarta City
Jakarta, 13220

After the model has been successfully prepared, this study utilizes Streamlit, an open-source framework, to create an interactive web-based user interface (UI) using Python. Prior to the development phase, system requirements were established, as outlined in Table 1.

Table 1. Requirements Used

No	Type of Requirements	Version
1.	Streamlit	1.34.0
2.	Opencv-python	4.9.0.80
3.	Numpy	1.26.4
4.	Joblib	1.4.2
5.	Scikit-learn	1.4.2
6.	Scikit-image	0.22.0
7.	Matplotlib	3.8.4
8.	Pillow	10.3.0

Based on the table, various requirements are utilized to manage dependencies within the project. Streamlit plays a role in system development, Numpy is used for numerical computations, Scikit-learn is employed for preprocessing and model selection, while Matplotlib is used for graphical visualization. The next step involves writing the Streamlit code. This process leverages a model saved in the .pkl format to perform label predictions. Additionally, labels such as “batik-parang” or “batik-ceplok” are converted into numerical form through encoding to be processed by the KNN algorithm. The code is written using Visual Studio Code, as shown in Figure 9.

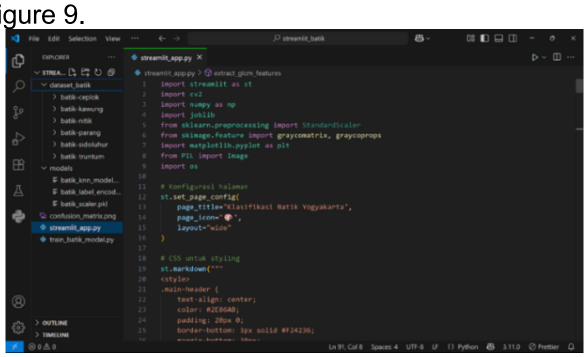


Figure 9. Writing Code in Visual Studio Code

After the code has been successfully written, the final step is to test the Streamlit application by running the command “streamlit run app.py” in the terminal. This command will generate a localhost link that is used to run the Streamlit project. In this study, the generated link is http://localhost:8501/.

RESULTS AND DISCUSSION

The implementation of batik motif classification has been successfully carried out using a dataset containing 1,350 Batik Yogyakarta images across 6 different motif classes. This dataset exhibits a fairly balanced distribution among the classes, as shown in the following table.

Universitas Negeri Jakarta
Phone: (021) 4890856
East Jakarta City
Jakarta, 13220

Table 2. Yogyakarta Batik Dataset Distribution

Type of Batik	Number of Samples	Percentage Level
Batik Parang	250	18.5%
Batik Sidoluhur	200	14.8%
Batik Ceplok	200	14.8%
Batik Kawung	250	18.5%
Batik Nitik	200	14.8%
Batik Truntum	250	18.5%

The data distribution shows that the batik motifs parang, kawung, and truntum each have the highest number of samples, with 250 samples (18.5%) each, indicating a high level of availability or popularity. Meanwhile, the sidoluhur, ceplok, and nitik motifs each consist of 200 samples (14.8%). The differences in sample counts among the motifs are relatively small, with percentages ranging from 14.8% to 18.5%.

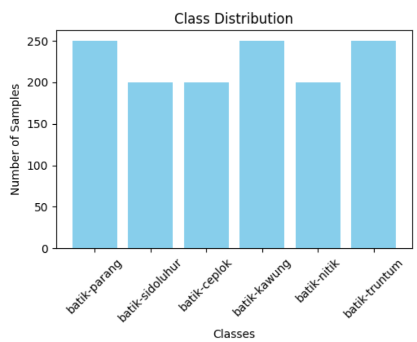


Figure 10. Class Distribution of Yogyakarta Batik Dataset on Google Colab

Subsequently, 12 texture features were successfully extracted from each image using the Gray Level Co-occurrence Matrix (GLCM) method. The following figure illustrates the GLCM analysis results on one of the batik images, specifically batik_parang_028.

```
ANALISIS SATU IMAGE:
==================================================
HASIL EKSTRAKSI FITUR GLCM
==================================================
No  Nama Fitur           Nilai
----------------------------------------
1   dissimilarity_mean   43.371930
2   dissimilarity_std    11.684348
3   correlation_mean     0.367952
4   correlation_std      0.304026
5   homogeneity_mean     0.024819
6   homogeneity_std      0.007975
7   contrast_mean        3126.497213
8   contrast_std         1504.096941
9   ASM_mean             0.000043
10  ASM_std              0.000007
11  energy_mean          0.006570
12  energy_std           0.000541

Jumlah total fitur: 12
Rentang nilai: 0.000007 - 3126.497213
```

Figure 11. GLCM Analysis Results (example: batik_parang_028 image) on Google Colab

The GLCM feature extraction results on the batik_parang_028 image reveal complex and high-contrast texture characteristics. The high values of contrast_mean (3126.50) and dissimilarity_mean (43.37) indicate significant variations in pixel intensity. Conversely, the low values of homogeneity_mean (0.0248), ASM_mean (0.000043), and energy_mean (0.00657) depict a non-uniform texture with a low

J-KOMA Journal of Computer Science and Applications

Universitas Negeri Jakarta
Phone: (021) 4890856
East Jakarta City
Jakarta, 13220

degree of pattern repetition. These values reflect the sharp, intricate, and heterogeneous motif of the parang batik.

```
=== Splitting Data ===
Training samples: 1080
Testing samples: 270
```

Figure 12. Data Splitting Results on Google Colab

From a total of 1,350 samples, the data were split using an 80:20 ratio with stratified sampling, resulting in 1,080 samples for training and 270 samples for testing. The stratified sampling method was employed to maintain the class proportion balance in both subsets, ensuring a more objective model evaluation and better reflecting the model's generalization capability on new data.

```
=== Parameter Tuning ===
Running Grid Search...
Fitting 5 folds for each of 12 candidates, totalling 60 fits
Best parameters: {'metric': 'euclidean', 'n_neighbors': 5, 'weights': 'distance'}
Best cross-validation score: 0.5944
```

Figure 13. Parameter Tuning Results on Google Colab

The parameter optimization process using Grid Search combined with 5-fold cross-validation successfully identified the best combination for the KNN algorithm, namely n_neighbors = 5, weights = 'distance', and metric = 'euclidean'. This model operates optimally by considering the 5 nearest neighbors, assigning greater weight to closer neighbors, and using the Euclidean distance as the metric. Among the 12 parameter combinations tested over 60 fitting iterations, the model achieved a highest cross-validation score of 59.44%, indicating a moderate classification performance

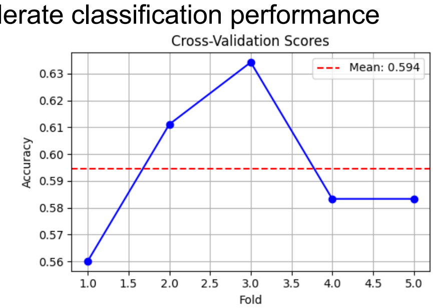


Figure 14. Visualizing Cross Validation Scores in Google Colab

The analysis of the cross-validation results demonstrates a reasonably good consistency, with accuracy across folds ranging from 56.02% to 63.43%. The third fold recorded the highest performance at 63.43%, while the first fold had the lowest value of 56.02%. This indicates a reasonable variation alongside an adequate level of model stability.

```
Mean CV score: 0.5944 (+/- 0.0513)
```

Figure 15. Standard Deviation Results in Google Colab

A standard deviation of ±0.0513 indicates relatively low variability across folds, suggesting that the model demonstrates good consistency and is not significantly affected by variations in the training data subsets. This implies that the model is capable of stable generalization and does not exhibit substantial overfitting.

```
=== Model Evaluation ===
Test Accuracy: 0.6000
```

Figure 16. KNN Model Accuracy Results in Google Colab

Based on the results from testing 1,350 samples of batik motifs, the model was trained on 1,080 samples and evaluated using 270 samples. The accuracy achieved was 60%, meaning approximately 162 samples were classified correctly, while the remaining 108 samples were misclassified. With an error rate of 40%, it is evident that, despite the comparatively large amount of training data, the model still struggles to distinguish several batik motifs. This difficulty may be attributed to similarities in patterns, the complexity of visual features, varying image quality, or the need for a more advanced model.

Table 3. KNN Classification Report Results

Classification Report			
Classification	**Precision**	**Recall**	**F1-Score**
batik-ceplok	67%	90%	77%
batik-kawung	54%	40%	46%
batik-nitik	51%	53%	52%
batik-parang	63%	54%	58%
batik-sidoluhur	58%	65%	61%
batik-truntum	64%	64%	64%

Referring to the results of the Classification Report containing the Precision, Recall, and F1-Score values for each class of batik motifs, the model's performance analysis is explained as follows:

1. Batik Ceplok exhibits the best performance, with Precision at 67%, Recall at 90%, and an F1-score of 77%. This indicates a high success rate of the model in accurately identifying this motif.
2. Batik Kawung is the class with the lowest results, with Precision of 54%, Recall of 40%, and F1-score of 46%, meaning the model faces difficulty in accurately recognizing this motif.
3. Batik Truntum demonstrates stable performance across all metrics (64%), while Batik Sidoluhur shows a Recall of 65% but relatively lower Precision (58%). Batik Nitik and Batik Parang have F1-scores of 52% and 58%, respectively.

Overall, the performance score differences highlight various challenges in the visual classification process for each motif. Batik Ceplok is the easiest motif to identify, while Batik Kawung is the most challenging class for the model to distinguish from other batik motifs.

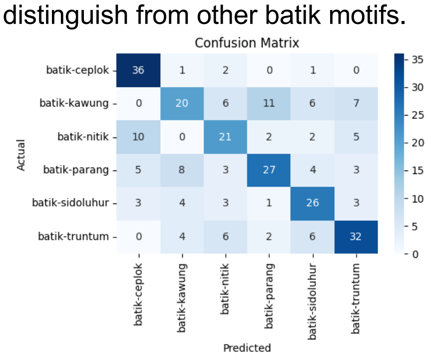


Figure 17. KNN Confusion Matrix Results

The results of the Confusion Matrix provide an in-depth overview of error patterns in the system's classification process. The following is a detailed analysis of classification outcomes for Yogyakarta batik motifs based on the obtained Confusion Matrix:

1. Batik Ceplok
   The batik-ceplok motif demonstrates strong performance, with 36 out of 40 samples accurately classified. Misclassifications are minimal and dispersed across the kawung, nitik, and sidoluhur motifs. This indicates that the distinct texture of batik-ceplok makes it easier for the model to correctly identify this motif.
2. Batik Truntum
   The batik-truntum motif exhibits fairly good performance, with 32 out of 50 samples accurately classified. The highest number of classification errors occurred with batik-nitik and batik-sidoluhur (6 samples each), followed by batik-kawung (4 samples) and batik-parang (2 samples). This pattern

suggests that the textural similarities between batik-truntum and several other motifs cause confusion for the model.

3. Batik Parang
   A total of 27 out of 50 batik-parang samples were correctly predicted. The largest number of errors occurred with batik-kawung (8 samples), reflecting notable textural similarity between the two motifs. Additional errors were distributed among batik-ceplok (5 samples), batik-nitik (3 samples), batik-truntum (3 samples), and batik-sidoluhur (1 sample). This indicates that the batik-parang motif shares textural features with other motifs, making it susceptible to misclassification.
4. Batik Sidoluhur
   The batik-sidoluhur motif was correctly classified in 26 out of 40 samples, while errors were observed in several other motifs, particularly batik-kawung (4 samples) and batik-ceplok (3 samples). This demonstrates that the texture of batik-sidoluhur bears resemblance to other motifs, which can result in classification confusion.
5. Batik Nitik
   The model succeeded in accurately classifying only 21 out of 40 batik-nitik samples. The majority of errors occurred with batik-ceplok samples (10 samples), while the rest were scattered across the truntum, parang, and sidoluhur motifs. This suggests that the batik-nitik motif is difficult to distinguish from other motifs due to its textural features.
6. Batik Kawung
   Batik-kawung exhibited the lowest performance, with only 20 out of 50 samples correctly classified. The most frequent errors were observed with batik-parang (11 samples) and batik-truntum (7 samples), with others distributed among batik-nitik and sidoluhur. This indicates that the texture of batik-kawung is highly similar to other motifs, especially batik-parang, making it challenging for the model to differentiate them.

These patterns of errors confirm the presence of textural similarities between several batik motifs based on GLCM features, especially between the kawung and parang motifs. The Confusion Matrix analysis highlights the need for special handling of motifs that are difficult to distinguish, such as by incorporating additional features or employing more accurate classification methods.

Figure 18. Yogyakarta Batik Classification Website

Figure 18 shows the user interface of the batik classification web application based on Streamlit. This application utilizes the GLCM method for feature extraction and the KNN algorithm for classification. There is a green notification displaying “Model berhasil dimuat”, indicating that the system is ready for use. Users can upload batik images via the “Browse files” button, then press the “Klasifikasi Batik” button to obtain prediction results along with the model’s confidence level.

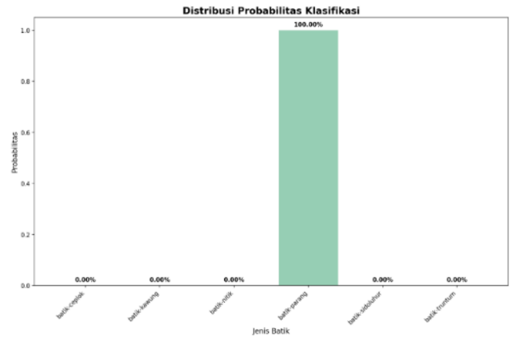


Figure 19. Classification of Probability Distributions Visualization

The system also displays probability visualizations for each batik class in the form of bar charts. In the given example, the batik-parang class has a probability of 100%, while the other classes show values of 0%, indicating that the image is fully classified as batik-parang.

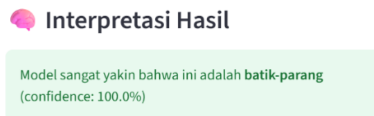


Figure 20. Interpretation of System Results

The system also provides an "Interpretasi Hasil" feature that offers explanations regarding the model's confidence level in the obtained classification outcome. For example, the model may show high confidence that the tested image belongs to the batik-parang type, with a confidence level of 100.0%.

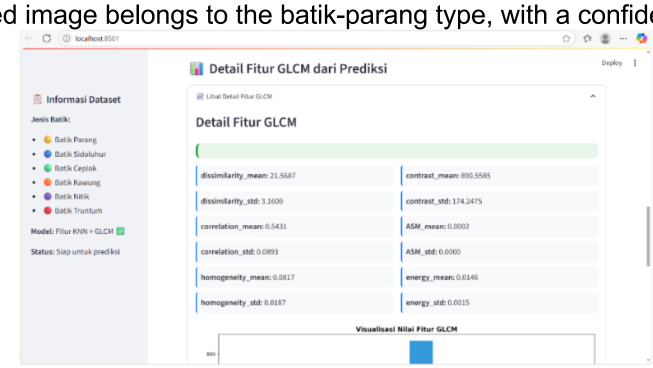


Figure 21. Detailed View of GLCM Features on the System

The system also displays detailed GLCM texture features in the "Detail Fitur GLCM dari Prediksi" section. It presents 12 feature values, namely the mean and standard deviation of six main GLCM properties, such as Dissimilarity, Correlation, Homogeneity, Contrast, ASM, and Energy. These features serve as the foundation for the classification process.

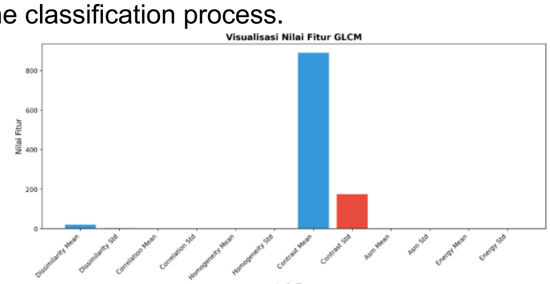


Figure 22. Visualization of GLCM Feature Values in the System

The GLCM feature values were then visualized in the form of bar charts to provide a visual overview of the contribution of each feature in the classification process. This visualization helps users understand how the texture characteristics of the tested batik images are utilized by the system in making classification decisions.

CONCLUSION

This research was successfully conducted on a dataset consisting of 1,350 images with 6 different classes of Yogyakarta batik motifs. The dataset distribution shows a good balance, with the percentage of each class ranging from 14.8% to 18.5%. Based on the research results, several conclusions can be drawn as follows:

1. The model achieved an accuracy of 60% on the test data with optimal parameters (n_neighbors=5, weights='distance', metric='euclidean') obtained through Grid Search and 5-fold cross-validation. The cross-validation results demonstrated stability with a standard deviation of ±0.0513, indicating model consistency without significant overfitting.
2. The batik-ceplok motif showed the best performance with an F1-score of 77%, while the batik-kawung motif had the lowest performance with an F1-score of 46%. The other motifs had F1-scores ranging between 52% and 64%. The confusion matrix revealed that the main classification errors occurred among motifs with similar textures, such as batik-kawung and batik-parang.
3. The system was successfully implemented as a web application using Streamlit, providing an interface for batik classification complete with probability visualization and result interpretation.

Overall, although the accuracy of 60% is an improvement compared to the random classification rate for 6 classes (16.7%), there is still significant room for improvement. The main challenge lies in the similarity of texture characteristics among certain motifs, necessitating the development of additional features or more advanced methodological approaches to enhance the model's discriminative ability in classifying Yogyakarta batik motifs.

REFERENCES

[1] A. A. Trixie, "FILOSOFI MOTIF BATIK SEBAGAI IDENTITAS BANGSA INDONESIA," *Folio*, vol. 1, no. 1, pp. 1–9, Feb. 2020.
[2] M. U. Januarko, "KEMITRAAN MASYARAKAT DAN STRATEGI PEMASARAN BATIK KELOMPOK PEMBATIK PALBATU," *Jurnal Abdimas*, vol. 6, no. 2, pp. 145–149, Jan. 2020.
[3] A. Prayoga, Maimunah, P. Sukmasetya, M. R. A. Yudianto, and R. A. Hasani, "Arsitektur Convolutional Neural Network untuk Model Klasifikasi Citra Batik Yogyakarta," *Journal of Applied Computer Science and Technology*, vol. 4, no. 2, pp. 82–89, 2023, doi: 10.52158/jacost.v4i2.486.
[4] A. Tjahyanto and F. J. Atletiko, "Peningkatan Kinerja Pengklasifikasi Objek Bawah Laut dengan Deep Learning," *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 21, no. 3, pp. 753–760, Jul. 2022, doi: 10.30812/matrik.v21i3.1466.
[5] M. J. D. Deke, T. A. Anastasya, A. D. P. Saka, and E. Y. Puspaningrum, "ANALISIS PENGARUH METODE EKSTRAKSI FITUR CITRA BATIK TERHADAP KINERJA KLASIFIKASI SVM," *Jurnal Mahasiswa Teknik Informatika*, vol. 9, no. 5, pp. 7516–7523, Oct. 2025, [Online]. Available: https://www.kaggle.com/datasets/geryxg/cora
[6] F. Haikal, M. Razak, and N. Umar, "Citra Digital Untuk Klasifikasi Kualitas Udang Windu Menggunakan Algoritma GLCM dan K-Nearest Neighbor," *JURNAL INFORMATIKA*, vol. 9, no. 2, pp. 93–102, Oct. 2022, [Online]. Available: http://ejournal.bsi.ac.id/ejurnal/index.php/ji
[7] F. Putra, H. F. Tahiyat, R. M. Ihsan, Rahmaddeni, and L. Efrizoni, "Penerapan Algoritma K-Nearest Neighbor Menggunakan Wrapper Sebagai Preprocessing untuk Penentuan Keterangan Berat Badan Manusia," *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 4, no. 1, pp. 273–281, Jan. 2024, doi: 10.57152/malcom.v4i1.1085.
[8] K. N. Oktaviarini, E. D. Wahyuni, and R. P. Sari, "Transformasi Data Statistik Menjadi Visual Interaktif Menggunakan Streamlit: Studi Kasus BPS Kota Mojokerto," *Jurnal Kridatama Sains dan Teknologi*, vol. 6, no. 2, pp. 804–822, Dec. 2024.
[9] Suharsono and B. M. Suparwanto, "Visualisasi Data Dari Data Ketidakhadiran Mahasiswa Menggunakan Pemrograman Python," *Jurnal Teknologi Informasi*, vol. 02, no. 02, pp. 103–113, Oct. 2023, [Online]. Available: http://jurnal.utu.ac.id/JTI
[10] Nurhalimah, I. G. P. S. Wijaya, and F. Bimantoro, "KLASIFIKASI KAIN SONGKET LOMBOK BERDASARKAN FITUR GLCM DAN MOMENT INVARIANT DENGAN TEKNIK

PENGKLASIFIKASIAN LINEAR DISCRIMINANT ANALYSIS (LDA)," *JTIKA*, vol. 2, no. 2, pp. 173–183, Sep. 2020, [Online]. Available: http://jtika.if.unram.ac.id/index.php/JTIKA/

[11] A. B. Suciani, Gunawansyah, and K. M. Adhinugraha, "KLASIFIKASI JENIS BATIK MENGGUNAKAN ALGORITMA K-NEAREST NEIGHBORS (KNN)," *Infotronik : Jurnal Teknologi Informasi dan Elektronika*, vol. 9, no. 2, pp. 83–89, Dec. 2024, doi: 10.32897/infotronik.2024.9.2.3846.