Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15th February 2025. Vol. 4. No. 2; e-ISSN: 2808-4519

Comparative Analysis of Model Architectures using Transfer Learning Approach in Convolutional Neural Networks for Traditional Ulos Fabric Classification

Taufik Abdullah[1]*, Kana Saputra S[2], Hermawan Syahputra[3], Zulfahmi Indra[4], Dinda Kartika[5]

[1,2,3,4]*Ilmu Komputer, Universitas Negeri Medan*
[5]*Matematika, Universitas Negeri Medan*
taufikabdullah700@gmail.com[1*]

Abstract

Ulos cloth is a traditional woven fabric of the Batak tribe in North Sumatra, valued for its aesthetic and symbolic significance in various ceremonies. The diversity of ulos motifs presents challenges in preservation due to their unique patterns and functions. This study aims to develop an accurate method for classifying ulos motifs using Transfer Learning on Convolutional Neural Network (CNN) architectures. Five popular models—VGG16, VGG19, MobileNetV3, Inception-V3, and EfficientNetV2—were evaluated on a dataset of 962 ulos images across six motif categories.The results show that Inception-V3 outperformed other models with an average validation accuracy of 98.13% and the lowest loss of 5.67%. Inception-V3 also demonstrated superior generalization, achieving the highest K-fold validation accuracy, while VGG16 and VGG19 exhibited overfitting at higher learning rates. Two-way ANOVA analysis confirmed significant performance differences among the models and highlighted the interaction between model type and training methods. This research recommends Inception-V3 as the optimal model for ulos motif classification, offering an efficient and reliable tool to support cultural preservation through advanced image recognition technology.

Keywords: *Accuracy; Classification; Inception-V3; Transfer Learning; Ulos Motifs*

1. Introduction

Indonesia is renowned for its extraordinary cultural richness, one of which is the variety of traditional textiles that serve as a legacy of art and ancestral heritage. Traditional fabrics such as Ulos, Batik, Ikat, and Songket are not only valued for their aesthetic appeal but also hold philosophical and symbolic meanings that reflect the identity and local wisdom of the communities that create them [1]. Ulos, a traditional woven fabric of the Batak people from North Sumatra, is one such example. It is used not only as clothing but also as a symbol of blessing, affection, and unity in traditional ceremonies [2]. In 2025, Ulos is being proposed by the North Sumatra Provincial Government to UNESCO as an intangible cultural heritage of the world [3].

Ulos carries sacred and mystical values that play a crucial role in the lives of the Batak community. The diversity of Ulos types enriches its cultural heritage, with each type serving unique functions in various traditional rituals and everyday use. However, certain types of Ulos, such as Ulos Raja, Ulos Ragi Botik, Ulos Gobar, Ulos Saput, and Ulos Sibolang, are now rarely produced and are at risk of extinction [4]. This situation underscores the urgency of identifying and preserving Ulos to ensure that knowledge about the remaining types is documented and appreciated before it is completely lost. A major challenge in recognizing Ulos types lies in the differences in motifs, which are often difficult to distinguish visually. Identifying Ulos types typically requires assistance from weavers or community elders. While culturally valuable, this process is time-consuming and prone to errors [5]. To address these challenges, Artificial Intelligence (AI) technology, particularly machine learning, can be applied to automate the process of Ulos motif classification [6], [7]. Algorithms such as Convolutional Neural Networks (CNNs) have shown potential in enhancing accuracy in classification tasks [8], [9], making them a relevant tool for recognizing traditional textile motifs.

Previous studies on fabric motif classification using CNNs demonstrated potential in identifying traditional textiles; however, model accuracy varies widely. CNN models trained from scratch tend to produce varied accuracy levels, ranging from 64.45% at the lowest to 87.27% at the highest [10]. Transfer Learning, which leverages pre-trained models, has demonstrated higher accuracy in several previous studies, such as in the classification of woven fabric patterns and X-ray images [11], [12]. Models like VGG, Inception-V3, MobileNet, and ResNet have proven effective in improving image recognition accuracy due to Transfer Learning [13], [14].

This study focuses on six types of Ulos: Ulos Bintang Maratur, Ulos Mangiring, Ulos Ragi Hotang, Ulos Ragi Hidup, Ulos Sadum, and Ulos Sibolang, chosen for their diverse motifs and cultural significance. By testing the capabilities of CNN and Transfer Learning methods on a dataset of images from these six types of Ulos, the study aims to compare various model architectures to determine the most effective method for classifying Ulos motifs. This research falls under the quantitative category, utilizing image data processed into numerical matrices derived from Ulos image extraction. It is also comparative, aiming to evaluate the effectiveness of five Transfer Learning architectures within Convolutional Neural Networks (CNNs), namely VGG16, VGG19, MobileNetV3, Inception-V3, and EfficientNetV2.

2. Research Method

This study comprises seven processes (as illustrated in Figure 1):

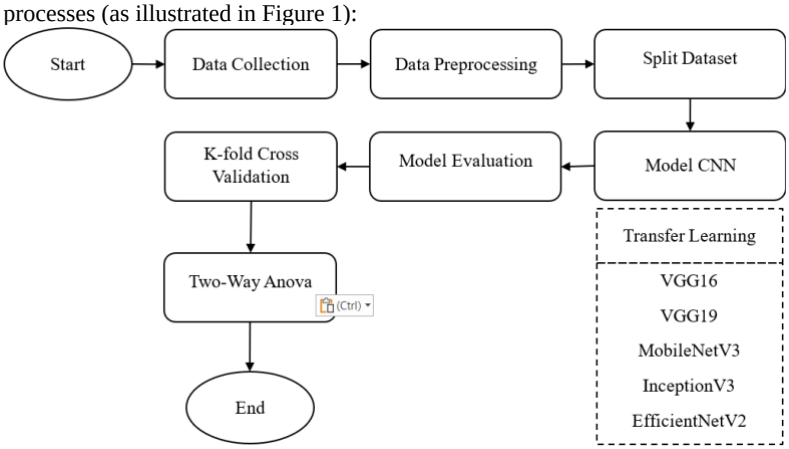

Fig. 1: Research workflow

2.1. Data Collection

The initial stage of this research focused on collecting data consisting of various images of Ulos fabric motifs. This data served as training data for the model. The data was collected through the Roboflow website, with a total of 962 Ulos images divided into six motif categories: Bintang Maratur, Mangiring, Ragihotang, Ragihidup, Sadum, and Sibolang. The data was accessed on January 24, 2024, from a public source on Roboflow (https://universe.roboflow.com/usu-ukmth/ulos/dataset/4). During the data collection process, image quality was examined to ensure that only complete and representative images were used. A total of 56 cropped or low-quality images were removed, leaving 906 images in the dataset. This removal ensured a balanced number of images in each Ulos motif category, with 151 images per category, as shown in Table 1.

Table 1: Comparison of total data classes

Ulos	Before Deletion	After Deletion
Bintangmaratur	175	151
Mangiring	170	151
Ragi Hidup	159	151
Ragi Hotang	155	151
Sadum	152	151
Sibolang	151	151

2.2. Data Preprocessing

After preparing the dataset, the data preprocessing stage began to minimize potential errors in the images. This stage involved verifying the correctness of the images with their designated folders and labeling each piece of data, which is a crucial part of the process. These labels were used to identify classes in the classification task. Additionally, at this stage, the collected images were resized for consistency, ensuring uniform dimensions for each data point. Data augmentation was also performed on the training images to enhance model performance and prevent overfitting.

Fig. 2: Results of image augmentation

2.3. Split Dataset

This stage involved splitting the dataset into two parts with an 80:20 ratio, where 80% of the total data was used for training and 20% for testing. This division means that out of the total 906 samples, 756 samples were used as training data, and 150 samples were used as testing data. Previous studies have shown that an 80:20 split ratio yields better accuracy results compared to other ratios such as 70:30, 60:40, and 50:50 [15]. This division is expected to improve the model's effectiveness in learning data patterns and measuring performance more accurately.

2.4. Model CNN

In this stage, a CNN model was constructed by adding a fully connected layer to the pre-defined Transfer Learning models. The hyperparameters used to build the CNN model are outlined in Table 2.

Table 2: Hyperparameters of the CNN Model

Hyperparameter	Nilai
Optimization	Adam
Learning Rate	1×10^{-3} | 1×10^{-4}
Epoch	50
Batch	32

The optimization algorithm used was Adaptive Momentum Optimization (Adam), selected based on previous studies showing that Adam achieved the highest validation accuracy and the lowest loss compared to other optimizers such as Rprop, Adagrad, Adamax, and Nadam, with a learning rate of 1×10^{-3} [16]. To examine the effect of learning rates, experiments were also conducted using a learning rate of 1×10^{-4}. The CNN model was trained on the preprocessed data using a batch size of 32 and 50 epochs, in line with previous studies indicating that this batch size yielded the highest accuracy compared to other batch sizes [17]. The Transfer Learning models utilized included architectures such as VGG16, VGG19, MobileNetV3, Inception-V3, and EfficientNetV2. These architectures comprised two components: feature learning and classification. The feature learning stage leveraged convolutional layers to recognize patterns, while the classification stage used these patterns to predict classes.

2.5. Model Evaluation

The model evaluation stage involved calculating metrics such as confusion matrix, recall, precision, F1 score, and accuracy to comprehensively assess performance. By monitoring the number of true positives, true negatives, false positives, and false negatives, these metrics were computed using Equations 1 through 4 [18].

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1\ Score = 2 \cdot \frac{1}{\frac{1}{Precision}+\frac{1}{Recall}} \quad (4)$$

2.6. K-fold Cross Validation

K-fold Cross Validation is a model validation technique used to measure model performance more accurately. In this method, the dataset is divided into k parts (folds), and the model is trained k times, each time using one fold as validation data while the rest are used for training. The results of each fold are then averaged to obtain the final accuracy and loss values. Figure 3 illustrates the data partitioning process, providing a detailed explanation of the folds used for training and validation.

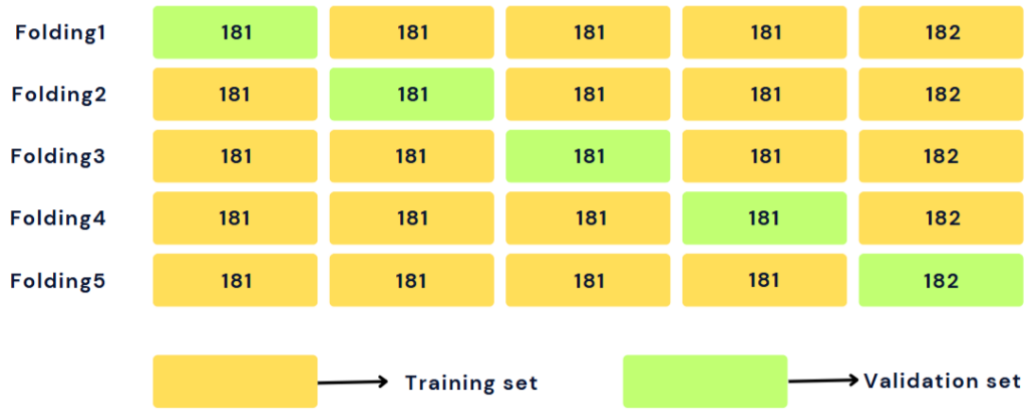

Fig. 3: Dataset split using 5-fold cross-validation

2.7. Two-Way Anova

ANOVA (Analysis of Variance) is a statistical method used to determine whether there are significant differences among the means of two or more groups. In this case, the researchers employed two-way ANOVA to evaluate the effects of two independent factors (Model and Method) as well as their interaction on the dependent variable, accuracy. Two-way ANOVA enables the analysis not only of the main effects of each factor but also of how these factors interact with one another.

3. Experiments and Analysis

3.1. Experimental setup

This study was conducted using Google Colab, a cloud-based platform that provides a Python programming environment with access to hardware like GPUs and TPUs. The experiments were carried out using Google Colab's TPUv2 (Tensor Processing Unit), a specialized hardware designed to accelerate machine learning computations, particularly in deep learning. TPUv2 offers high computational power, enabling more efficient model training compared to conventional CPUs or GPUs. The use of TPUv2 allowed the researchers to perform computations with greater resources without depending on local hardware specifications.

3.2. Experimental results and analysis

To evaluate the performance of each model, precision, recall, F1-score, and accuracy metrics were calculated, as presented in Table 3.

Table 3: Validation results of each model

Model	Learning Rate	Data Class	Precision	Recall	F1 score	Accuracy (%)
VGG16	$1x10^{-3}$	Bintangmaratur	0.96	1	0.98	99
		Mangiring	1	0.96	0.98	
		Ragihidup	1	1	1	
		Ragihotang	1	1	1	
		Sedum	1	1	1	
		Sibolang	1	1	1	
	$1x10^{-4}$	Bintangmaratur	0.66	1	0.79	84
		Mangiring	0.89	0.96	0.92	
		Ragihidup	0.90	0.72	0.80	
		Ragihotang	0.85	0.68	0.76	
		Sedum	0.95	0.72	0.82	
		Sibolang	0.92	0.96	0.94	
VGG19	$1x10^{-3}$	Bintangmaratur	1	1	1	99
		Mangiring	1	1	1	
		Ragihidup	0.96	1	0.98	
		Ragihotang	1	0.96	0.98	
		Sedum	1	1	1	
		Sibolang	1	1	1	
	$1x10^{-4}$	Bintangmaratur	0.86	1	0.93	91
		Mangiring	1	0.96	0.98	
		Ragihidup	0.80	0.80	0.80	
		Ragihotang	0.89	0.96	0.92	
		Sedum	1	0.84	0.91	
		Sibolang	0.92	0.88	0.90	
MobileNet V3 Large	$1x10^{-3}$	Bintangmaratur	0.70	0.92	0.79	61
		Mangiring	0.55	0.72	0.62	
		Ragihidup	1	0.44	0.61	
		Ragihotang	0.58	0.44	0.50	
		Sedum	0.38	0.60	0.46	
		Sibolang	1	0.56	0.72	
	$1x10^{-4}$	Bintangmaratur	0.58	0.44	0.50	49
		Mangiring	0.32	0.80	0.46	
		Ragihidup	1	0.12	0.21	
		Ragihotang	0.41	0.48	0.44	
		Sedum	0.73	0.44	0.55	
		Sibolang	0.73	0.64	0.68	
Inception-V3	$1x10^{-3}$	Bintangmaratur	1	1	1	99
		Mangiring	0.93	1	0.96	
		Ragihidup	1	1	1	
		Ragihotang	1	0.92	0.96	
		Sedum	1	1	1	
		Sibolang	1	1	1	
	$1x10^{-4}$	Bintangmaratur	1	1	1	99
		Mangiring	0.93	1	0.96	
		Ragihidup	1	1	1	
		Ragihotang	1	0.92	0.96	
		Sedum	1	1	1	
		Sibolang	1	1	1	
Efficient NetV2B1	$1x10^{-3}$	Bintangmaratur	0	0	0	17
		Mangiring	0	0	0	
		Ragihidup	0	0	0	

	Ragihotang	0	0	0	
	Sedum	0	0	0	
	Sibolang	0.17	1	0.29	
	Bintangmaratur	0.23	0.76	0.36	
	Mangiring	0	0	0	
$1x10^{-4}$	Ragihidup	0	0	0	29
	Ragihotang	0	0	0	
	Sedum	0	0	0	
	Sibolang	0.37	1	0.54	

Table 3 presents the performance metrics of various Transfer Learning (TL) models in identifying different types of Ulos patterns in the testing dataset. The testing accuracy demonstrates how effectively the deep learning models perform classification. Based on Table 3, VGG16 achieved the highest accuracy of 99% with a learning rate of 1×10^{-3}, along with almost perfect precision, recall, and F1-score values for all data classes. However, with a learning rate of 1×10^{-4}, VGG16's performance declined, showing an accuracy of 84% with varied metrics across different classes. VGG19 exhibited excellent performance with an accuracy of 99% at a learning rate of 1×10^{-4}, maintaining high precision, recall, and F1-score values for all classes. At a learning rate of 1×10^{-4}, VGG19 still performed well with an accuracy of 91%, though some classes experienced a drop in precision and recall. MobileNetV3Large showed lower performance with a 61% accuracy at a learning rate of 1×10^{-3}, and its precision and recall varied significantly across classes. Its performance further declined with a 49% accuracy at a learning rate of 1×10^{-4}, and lower metrics for most classes. InceptionV3 maintained consistent performance, achieving 99% accuracy at both learning rates, with high precision, recall, and F1-score values for all classes. EfficientNetV2B1 demonstrated the lowest results, with accuracies of 17% at a learning rate of 1×10^{-3} and 29% at 1×10^{-4}, alongside very low precision and recall values except for slight improvements in the Sibolang class.

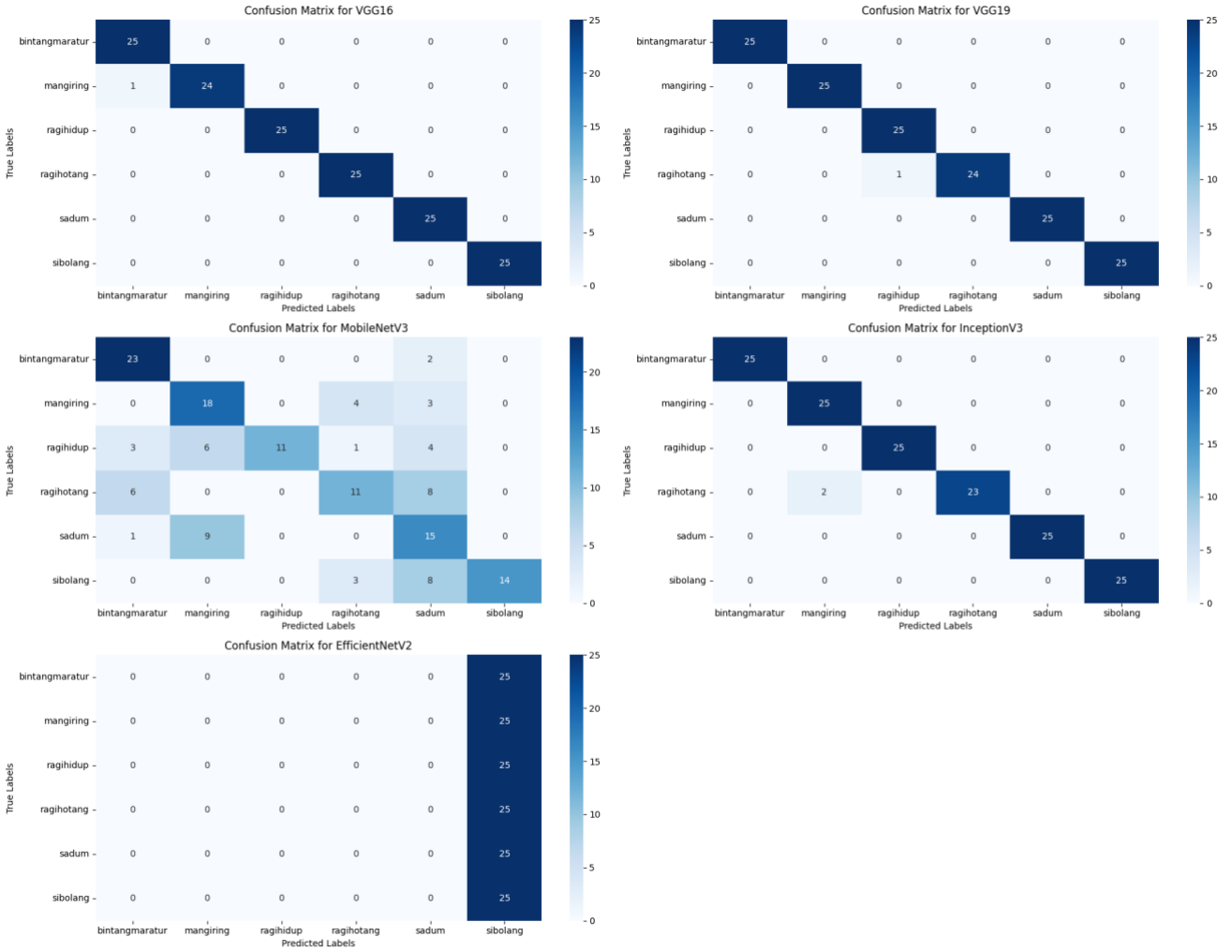

Fig. 4: Confusion matrix learning rate 1×10^{-3}

Figure 4 illustrates the confusion matrix for the testing dataset, offering an intuitive representation of the classification performance of each TL model. The analysis of results for a learning rate of (1×10^{-3}) reveals significant differences in the tested models' performance. VGG16 and VGG19 showed excellent results, achieving near-perfect accuracy in classifying each class. VGG16 had minimal errors, except for the "Mangiring" class, where one instance was misclassified as "Bintangmaratur." VGG19 also showed high accuracy, with a single error in the "Ragihotang" class, which was incorrectly classified as "Ragihidup." Conversely, MobileNetV3 demonstrated varied performance; while it classified the "Bintangmaratur" and "Sibolang" classes well, it struggled to accurately classify other classes. The "Mangiring" and "Sadum" classes had numerous misclassifications, while the "Ragihotang" class had predictions scattered across other

classes, including "Ragihidup," "Sadum," and "Sibolang." InceptionV3, similar to VGG16 and VGG19, exhibited excellent performance with only one error in the "Ragihotang" class, which was classified as "Ragihidup." On the other hand, EfficientNetV2 showed the poorest performance, failing to effectively distinguish between other classes and assigning most instances to a single class.

Fig. 5: Confusion matrix learning rate 1×10^{-4}

Figure 5 presents the confusion matrix for the testing dataset with a learning rate of (1×10^{-4}), highlighting performance variations among the models. VGG16 delivered excellent results with only a few classification errors, notably in the "Ragihidup" class, where seven instances were classified as "Ragihotang" and one as "Sadum," while other classes were classified correctly or with minimal errors. VGG19 also performed well, although there were some misclassifications in the "Ragihidup" and "Sadum" classes, with three instances of "Ragihidup" being classified as "Ragihotang" and one as "Mangiring." InceptionV3 achieved near-perfect classification, with only two errors in the "Ragihotang" class being classified as "Ragihidup," making it one of the most accurate models tested. Conversely, MobileNetV3 showed more varied results, with classification errors spread across different classes, particularly in the "Bintangmaratur" and "Mangiring" classes, where instances were misclassified into other classes such as "Sibolang" and "Ragihotang," leading to less consistent performance than other models. EfficientNetV2, despite showing slight improvement compared to its earlier results, continued to perform poorly, with significant misclassifications in the "Ragihotang" and "Sadum" classes, where several instances were classified as "Sibolang," and it failed to match the performance of models like VGG16, VGG19, and InceptionV3.

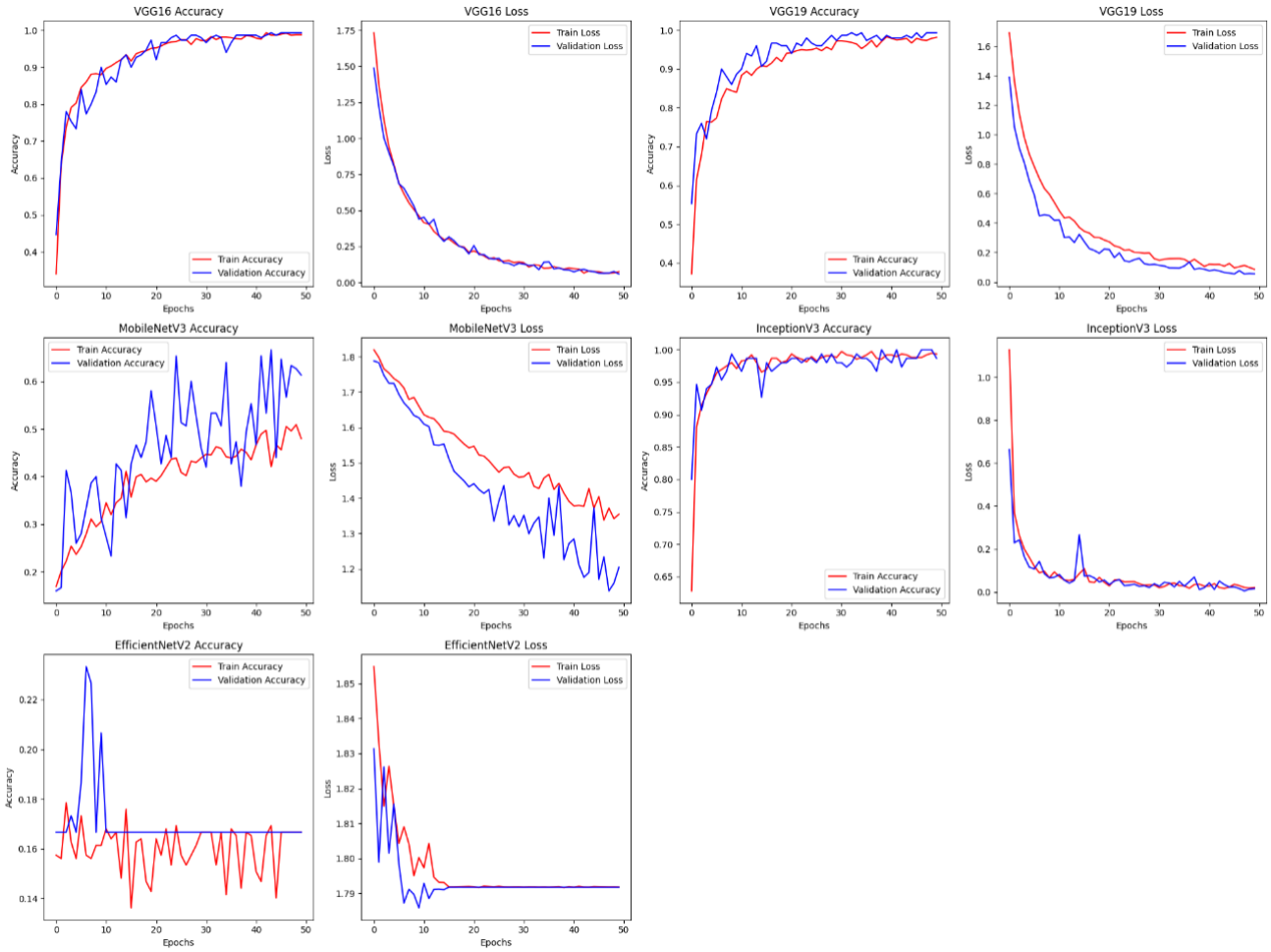

Fig. 6: Accuracy and Loss Graph with a learning rate of 1×10^{-3}

Figure 6 presents the accuracy and loss curves per epoch for various deep learning models (VGG16, VGG19, MobileNetV3, InceptionV3, and EfficientNetV2) with a learning rate of 1×10^{-3} over 50 epochs, providing an overview of each model's performance. The curve for VGG16 shows an increase in accuracy approaching near-optimal levels, with training and validation accuracy reaching approximately 99%. The loss decreases effectively, but the validation accuracy curve displays fluctuations, with accuracy rising and falling throughout the epochs. This indicates that while the model approaches overfitting, the fluctuations in validation accuracy suggest potential instability in performance on unseen data. VGG19 also exhibits an increase in accuracy approaching near-optimal levels, slightly outperforming VGG16, though there is a small gap between training and validation accuracy, where validation accuracy is slightly higher. Validation accuracy reaches approximately 99%, with minor fluctuations indicating mild overfitting, albeit still controlled. In contrast, MobileNetV3 displays greater fluctuations in accuracy; while training accuracy remains stable but lower, validation accuracy is highly volatile. The loss curve for MobileNetV3 shows a general decline but also fluctuates, indicating challenges in consistent learning. InceptionV3 demonstrates excellent performance, with training and validation accuracy rapidly achieving high values near 1.0 (100%) and a steady, rapid decline in loss. However, the validation accuracy curve for InceptionV3 also shows some fluctuations, similar to VGG16, indicating potential instability in generalization, even though the model achieves very high accuracy. EfficientNetV2, on the other hand, exhibits very low accuracy and shows no significant improvement over the 50 epochs. The loss curve indicates a rapid initial decrease but remains high afterward, reflecting the model's difficulty in achieving good generalization at this learning rate.

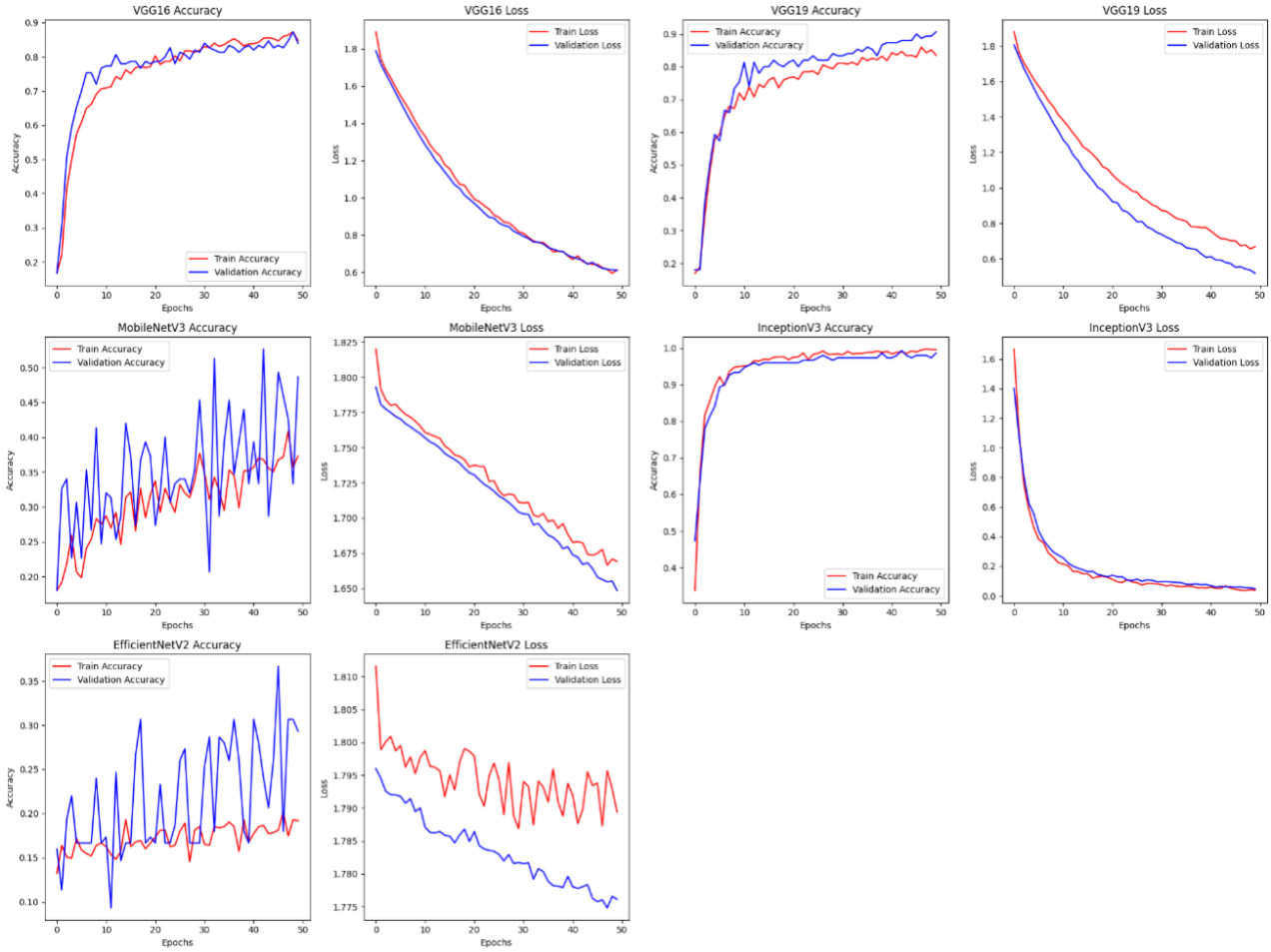

Fig. 7: Accuracy and Loss Graph with a learning rate of 1×10^{-4}

Figure 7 depicts the accuracy and loss curves per epoch for several deep learning models (VGG16, VGG19, MobileNetV3, InceptionV3, and EfficientNetV2) using a learning rate of 1×10^{-4} over 50 epochs, providing insights into the models' performance across epochs. VGG16 shows a steady increase in accuracy, with validation accuracy reaching approximately 84%, accompanied by a stable decline in loss without signs of overfitting, indicating good generalization performance. VGG19 also demonstrates a solid increase in accuracy, nearing 90%, with validation accuracy slightly lower than training accuracy and a stable loss decline, though exhibiting minor overfitting. MobileNetV3 experiences significant fluctuations in accuracy, especially in validation, with its loss curve also displaying volatility, indicating challenges in convergence at this learning rate. InceptionV3 demonstrates excellent performance, with accuracy rapidly reaching high values between 90% and 98% and a steady, rapid decline in loss, indicating good generalization. EfficientNetV2, while showing improvement compared to a learning rate of 1×10^{-3}, still exhibits low and fluctuating accuracy and limited loss reduction, reflecting difficulties in achieving optimal learning at this rate.

To determine whether 1×10^{-3} or 1×10^{-4} is a better learning rate, an analysis of the accuracy and loss graphs reveals significant differences. For VGG16 and VGG19, 1×10^{-3} results in a rapid accuracy increase but with some fluctuations, where accuracy rises and falls throughout the epochs. This indicates that, while the models approach overfitting, the fluctuations in validation accuracy suggest potential instability in performance on unseen data. Conversely, 1×10^{-4} provides steady accuracy improvements, albeit slightly slower, with greater stability. For VGG19, validation loss is slightly higher than training loss, suggesting mild overfitting but within acceptable limits. MobileNetV3 with 1×10^{-3} shows highly fluctuating accuracy in both training and validation, indicating that the learning rate might be too high. With 1×10^{-4}, while fluctuations persist, the model exhibits slightly better stability, although accuracy remains low, indicating the need for further optimization. InceptionV3 with 1×10^{-3} achieves high accuracy quickly but also shows significant fluctuations, suggesting that 1×10^{-3} might be too high. In contrast, 1×10^{-4} delivers more stable performance with consistent accuracy improvements, indicating smoother and more reliable learning. EfficientNetV2 performs poorly with 1×10^{-3}, showing very low accuracy and unstable loss. While 1×10^{-4} results in slight improvements, performance remains low, suggesting the need for additional optimization. Overall, 1×10^{-4} appears to be the better learning rate for most models, offering greater stability and more consistent performance, except for VGG19, where 1×10^{-3} provides more stable and effective results.

3.3.1. Computational time

In this experiment, Google Colaboratory with TPUv2 was used, and the average training time per epoch and the total training time for various models and learning rates showed significant variation. MobileNetV3Large had the shortest training time per epoch, approximately 13.72 seconds for 1×10^{-3} and 13.62 seconds for 1×10^{-4}, with a total training time of 11 minutes and 24 seconds for both learning rates. InceptionV3 and EfficientNetV2B1 demonstrated relatively consistent training times, approximately 14.6 seconds per epoch for 1×10^{-4} and 15.22 seconds per epoch, respectively, with total training times of about 12 minutes and 13 seconds and 12 minutes and 44

seconds. VGG16 and VGG19 had longer training times per epoch, around 25 seconds (VGG16) and 24 seconds (VGG19) for the learning rate 1×10^{-4}, with total training times of approximately 21 minutes and 46 seconds and 20 minutes and 39 seconds, respectively.

Table 4: Training time results for each model

Model	Learning rate	Training time per Epoch (AVG)	Training time (Total)
VGG16	$1x10^{-3}$	24.58 seconds	20 minutes 40 seconds
	$1x10^{-4}$	25.76 seconds	21 minutes 46 seconds
VGG19	$1x10^{-3}$	23.8 seconds	19 minutes 43 seconds
	$1x10^{-4}$	24.9 seconds	20 minutes 39 seconds
MobileNetV3Large	$1x10^{-3}$	13.72 seconds	11 minutes 24 seconds
	$1x10^{-4}$	13.62 seconds	11 minutes 24 seconds
InceptionV3	$1x10^{-3}$	14.82 seconds	12 minutes 15 seconds
	$1x10^{-4}$	14.6 seconds	12 minutes 13 seconds
EfficientNetV2B1	$1x10^{-3}$	15.22 seconds	12 minutes 45 seconds
	$1x10^{-4}$	15.22 seconds	12 minutes 44 seconds

3.3.2. Testing Models

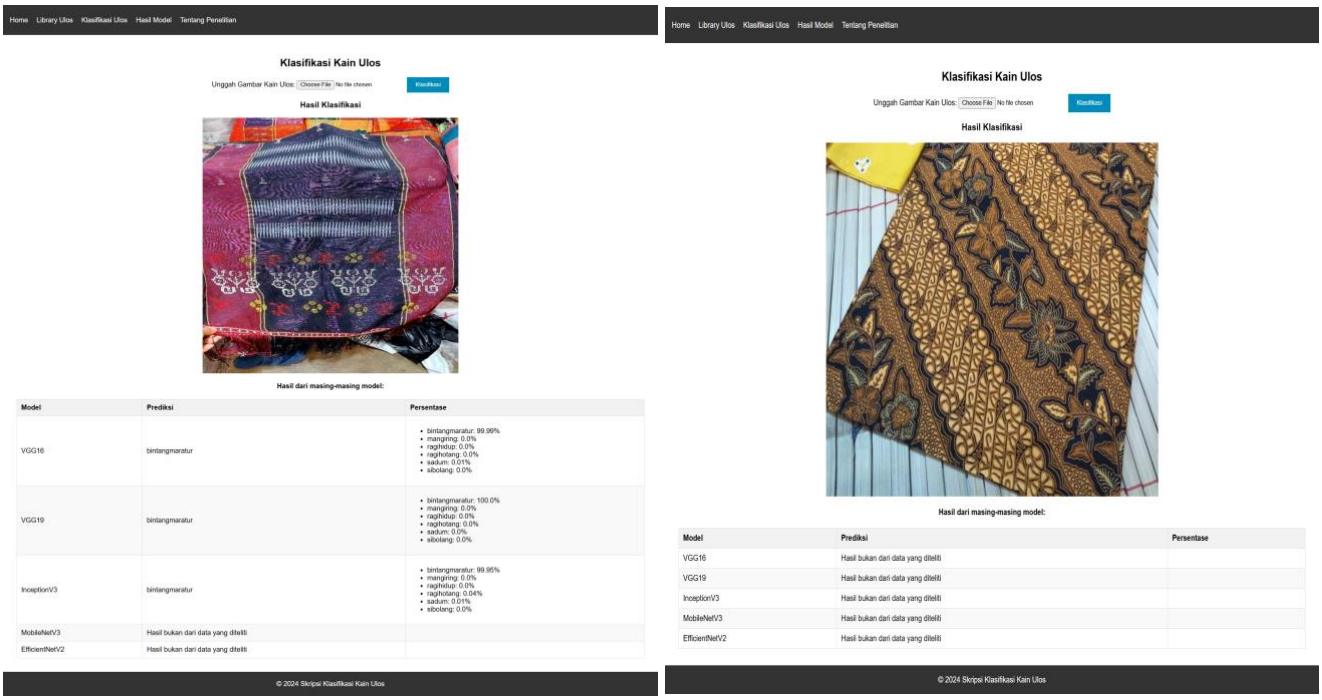

Fig. 8: Testing for each model

After completing the training process, the researcher conducted testing to evaluate the models' performance in image classification. The testing was carried out using two types of data: Bintang Maratur data and new data consisting of batik images that were never involved in the training process. In the tests using Bintang Maratur data, the VGG16, VGG19, and Inception-V3 models successfully predicted the images correctly, demonstrating strong capabilities in recognizing the motif patterns previously trained. However, the MobileNetV3 and EfficientNetV2B1 models failed to predict the Bintang Maratur data accurately, indicating limitations in recognizing more complex patterns. Meanwhile, tests with new data consisting of batik images revealed that all models, including VGG16, VGG19, MobileNetV3, Inception-V3, and EfficientNetV2B1, correctly predicted that the data was different from the ulos motif categories they were trained on. These results indicate that while some models showed weaknesses in recognizing training data, all models demonstrated good generalization capabilities in distinguishing significantly different visual patterns.

3.4. K-fold cross validation

The K-fold Cross Validation test, using 5-fold, was conducted to evaluate the performance of several Transfer Learning models, including VGG16, VGG19, MobileNetV3 Large, InceptionV3, and EfficientNetV2B1. This evaluation included accuracy and loss for each fold, as well as the average values across all folds. The researcher used a learning rate that was deemed more effective overall, providing greater stability and consistent performance for most models, except for VGG19, where the learning rate yielded more stable and effective results.

Table 5: Results of 5-fold cross-validation for each model

Model		**K-fold Cross Validation**					**Mean** (%)
		1	**2**	**3**	**4**	**5**	
VGG16	Accuracy	0.826	0.833	0.839	0.806	0.860	83.33
	Loss	0.655	0.658	0.609	0.665	0.601	63.80
VGG19	Accuracy	0.893	0.866	0.833	0.899	0.853	86.93
	Loss	0.530	0.596	0.602	0.530	0.578	57.53
MobileNet V3Large	Accuracy	0.5	0.466	0.373	0.386	0.433	43.20

InceptionV3	Loss	1.663	1.651	1.643	1.659	1.643	165.23
	Accuracy	0.980	0.980	1	0.973	0.973	98.13
	Loss	0.059	0.055	0.049	0.056	0.061	5.67
Efficient NetV2B1	Accuracy	0.293	0.366	0.273	0.266	0.193	27.87
	Loss	1.775	1.771	1.775	1.774	1.774	177.42

VGG19 demonstrated slightly better performance than VGG16, with an average accuracy of 86.93%. The highest accuracy of 89.9% was recorded in fold 4, while the lowest accuracy of 83.3% occurred in fold 3. The average loss was 57.53%, with the lowest loss of 53.0% recorded in folds 1 and 4, indicating good generalization capability on this dataset. MobileNetV3 Large showed unsatisfactory performance with an average accuracy of 43.20%. The highest accuracy was 50.0% in fold 1, while the lowest accuracy was only 37.3% in fold 3. The high average loss of 165.23% indicates a significant prediction error rate. InceptionV3 achieved the best performance among all models, with an average accuracy of 98.13%. Perfect accuracy was achieved in fold 3 at 100%, and the lowest accuracy remained high at 97.3% in folds 4 and 5. It's very low average loss of 5.67% underscores its excellent predictive ability with minimal error. EfficientNetV2B1 demonstrated the poorest performance, with an average accuracy of only 27.87%. The highest accuracy was 36.6% in fold 2, and the lowest accuracy was just 19.3% in fold 5. The average loss of 177.42% reflects extremely high prediction error, consistent with its low accuracy. Overall, the results of the K-fold Cross Validation highlight that InceptionV3 is the best model for ulos classification, followed by VGG19 and VGG16. In contrast, MobileNetV3 Large and EfficientNetV2B1 showed inadequate performance for this task, with low accuracy and high loss values.

3.5. Uji Anova

Table 6: Two-way ANOVA test results

	Sum_sq	df	F	PR(>F) (p-value)
C(Model)	2.703517	4.0	424.942374	4.924638e-19
C(*Method*)	0.022490	2.0	7.070149	4.760219e-03
C(Model):C(*Method*)	0.051678	8.0	4.061405	5.186686e-03
Residual	0.031810	20.0	-	-

Table 6 shows that the F-statistic value for the Model factor is 424.94 with a p-value of 4.924638e-19, which is much smaller than the significance level of 0.05. This indicates that the differences between the models in this study significantly affect accuracy, confirming that different models yield significantly different accuracies. For the Method factor, the F-statistic value is 7.07 with a p-value of 0.0048, which is also smaller than 0.05, indicating that the differences in the methods applied have a significant impact on accuracy and that the method used significantly influences accuracy. Additionally, the F-statistic value for the interaction between Model and Method is 4.06 with a p-value of 0.0052, indicating a significant interaction between these two factors in influencing accuracy, with the model's effect varying depending on the method applied, thus rejecting the null hypothesis regarding the interaction. The residual variation in this ANOVA model is 0.031810 with 20 degrees of freedom, reflecting the variability that cannot be explained by the tested factors, and it has no p-value as it represents unexplained variation by those factors. Based on the results of the two-way ANOVA, the factors of Model, Method, and their interaction significantly influence accuracy. These findings suggest that the differences in model and method, as well as how they interact, significantly impact accuracy.

4. Conclusion

This study successfully demonstrated the application of Transfer Learning in classifying six traditional *ulos* fabric patterns using CNN architecture. Among the evaluated models—VGG16, VGG19, MobileNetV3, Inception-V3, and EfficientNetV2B1—the Inception-V3 model showed the best performance. This model achieved the highest validation accuracy of 98.13% with the lowest loss of 5.67%, while maintaining consistent results in K-fold validation. This indicates an exceptional generalization ability in the classification task. The analysis also revealed significant differences in model performance, as shown by the two-way ANOVA, particularly regarding the impact of model architecture and training configuration. Although VGG16 and VGG19 achieved near-perfect results at certain learning rates, their performance tended to vary significantly when the learning rate was changed, indicating potential overfitting under suboptimal conditions. MobileNetV3 and EfficientNetV2B1, while showing lower initial performance, demonstrated potential for improvement with advanced configurations or fine-tuning strategies.

Acknowledgement

The authors would like to express their gratitude to everyone who provided valuable feedback and support during the completion of this work.

References

[1] J. Wahyudi and I. Maulida, "Pengenalan Pola Citra Kain Tradisional Menggunakana GLCM dan KNN," *JTIULM*, vol. 4, no. 2, pp. 43–48, 2019, doi: https://doi.org/10.20527/jtiulm.v4i2.37.

[2] B. Siregar, I. P. S. Panggabean, Fahmi, and A. Hizriadi, "Classification of traditional ulos of Batak Toba ssing probabilistic neural network," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, May 2021, pp. 1–9. doi: 10.1088/1742-6596/1882/1/012131.

[3] E. R. L. Tinambunan, "Ulos Batak Toba: Makna Religi dan Implikasinya pada Peradaban dan Estetika," *FORUM Filsafat dan Teologi*, vol. 52, no. 2, pp. 122–142, Oct. 2023, doi: 10.35312/forum.v52i2.583.

[4] D. Esterlina Br Jabat, L. Yanti Sipayung, and K. Raih Syahputra Dakhi, "Penerapan Algoritma Recurrent Neural Networks (RNN) Untuk Klasifikasi Ulos Batak Toba," *SNISTIK : Seminar Nasional Inovasi Sains Teknologi Informasi Komputer*, vol. 1, no. 2, pp. 3025–8715, 2024.

[5] D. H. Manurung, I. M. Lattu, and R. Tulus, "Struktur Cosmos Masyarakat Batak dalam Simbol Ulos," *Anthropos: Jurnal Antropologi Sosial dan Budaya (Journal of Social and Cultural Anthropology)*, vol. 6, no. 1, p. 31, Apr. 2020, doi: 10.24114/antro.v6i1.16603.

[6] R. Mawan, "Klasifikasi motif batik menggunakan convolutional neural network," *JNANALOKA*, vol. 1, no. 1, pp. 45–50, 2020, doi: 10.36802/jnanaloka.

[7] A. A. Prahartiningsyah and T. B. Kurniawan, "Pengenalan Pola Angka Menggunakan Pendekatan Optimisasi Sistem Kekebalan Buatan (Artificial Immune System)," *JURNAL MEDIA INFORMATIKA BUDIDARMA*, vol. 5, no. 3, p. 856, Jul. 2021, doi: 10.30865/mib.v5i3.2997.

[8] N. K. Qudsi, R. A. Asmara, and A. R. Syulistyo, "Identifikasi Citra Tulisan Tangan Digital Menggunakan Convolutional Neural Network (CNN)," 2019.

[9] M. Krichen, "Convolutional Neural Networks: A Survey," *Computers*, vol. 12, no. 8, pp. 1–41, Aug. 2023, doi: 10.3390/computers12080151.

[10] A. F. Siregar and T. Mauritsius, "Ulos fabric classification using android-based convolutional neural network," *International Journal of Innovative Computing, Information and Control*, vol. 17, no. 3, pp. 753–766, 2021, doi: 10.24507/ijicic.17.03.753.

[11] Anhar and R. A. Putra, "Perancangan dan Implementasi Self-Checkout System pada Toko Ritel menggunakan Convolutional Neural Network (CNN)," *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika*, vol. 11, no. 2, pp. 466–478, Apr. 2023, doi: 10.26760/elkomika.v11i2.466.

[12] A. E. Wijaya, W. Swastika, and O. H. Kelana, "Implementasi Transfer Learning Pada Convolutional Neural Network Untuk Diagnosis Covid-19 Dan Pneumonia Pada Citra X-Ray," *SAINSBERTEK Jurnal Ilmiah Sains & Teknologi*, vol. 2, no. 1, 2021.

[13] D. Dais, İ. E. Bal, E. Smyrou, and V. Sarhosis, "Automatic crack classification and segmentation on masonry surfaces using convolutional neural networks and transfer learning," *Autom Constr*, pp. 1–18, May 2021, doi: 10.1016/j.autcon.2021.103606.

[14] H. Noprisson, E. Ermatita, A. Abdiansah, V. Ayumi, M. Purba, and H. Setiawan, "Fine-Tuning Transfer Learning Model In Woven Fabric Pattern Classification," *International Journal of Innovative Computing, Information and Control*, vol. 18, no. 6, pp. 1885–1894, 2022, doi: 10.24507/ijicic.18.06.1885.

[15] D. S. Prashanth, R. V. K. Mehta, and N. Sharma, "Classification of Handwritten Devanagari Number - An analysis of Pattern Recognition Tool using Neural Network and CNN," in *Procedia Computer Science*, Elsevier B.V., 2020, pp. 2445–2457. doi: 10.1016/j.procs.2020.03.297.

[16] S. Sennan, D. Pandey, Y. Alotaibi, and S. Alghamdi, "A Novel Convolutional Neural Networks Based Spinach Classification and Recognition System," *Computers, Materials and Continua*, vol. 73, no. 1, pp. 343–361, 2022, doi: 10.32604/cmc.2022.028334.

[17] A. Julianto, A. Sunyoto, D. Ferry, and W. Wibowo, "Optimasi Hyperparameter Convolutional Neural Network Untuk Klasifikasi Penyakit Tanaman Padi," *TEKNIMEDIA*, vol. 3, no. 2, pp. 98–105, 2022, doi: https://doi.org/10.46764/teknimedia.v3i2.77.

[18] I. I. Daqiqi, *MACHINE LEARNING: Teori, Studi Kasus dan Implementasi Menggunakan Python*. UR PRESS, 2021. doi: 10.5281/zenodo.5113507.