Implementation of Convolutional Neural Networks for Batik Image Dataset

Vina Ayumi, Ida Nurhaida and Handrie Noprisson

***Abstract*—One method of image recognition that can be used is a convolutional neural network (CNN). However, the training model of CNN is not an easy thing; it takes tuning parameters that take a long time in the training process. This research will do Batik pattern recognition by using CNN. From the experiment that we conducted, the result shows that the feature extraction, selection, and reduction give the accuracy more significant than raw image dataset. The feature selection and reduction also can improve the execution time. Parameters value that gave best accuracy are: epoch = 200, batch_size = 20, optimizer = adam, learning_rate = 0.01, network weight initialization = lecun_uniform, neuron activation function = linear.**

***Index Terms*—Image recognition, convolutional neural networks, optimization of tuning parameters.**

I. Introduction

INDONESIA has a cultural heritage in the form of diversity Batik patterns [1]. In general, batik has a pattern of batik motifs geometry and non-geometry [2]. The existence of patterns on this batik motif raises the topic of research in the field of pattern recognition, especially image processing [3], [4], [5].

Image processing technology lately can be used to help the process of recognition of batik motifs. Methods that exist in image processing can help machine learning and recognizing the motif on batik.

Previous research that has been done in the field of image processing technology with the object of batik image has been started in 2009 to 2012 using various methods such as Gray-Level Co-Occurrence Metrics (GLCM), Scale Invariant Feature Transform, Canny Edge Detection, Gabor Filter, K-mean, Log Gabor Filter and Color Histogram [3], [6], [7], [8], [9].

One method of image recognition that can be used is a convolutional neural network (CNN). This research is conducted for Batik pattern recognition by using CNN.

II. Literature Review

Introduced by Yann & Yoshua (1998), Convolutional Neural Networks (CNN) is inspired by the workings of nerves in the visual cortex that have complex structures [10], [11].

V. Ayumi and I. Nurhaida are with the Faculty of Computer Science, Universitas Mercu Buana, Jakarta, Indonesia e-mail: vina.ayumi@mercubuana.ac.id, ida.nurhaida@mercubuana.ac.id.

H. Noprisson is with the Faculty of Computer Science, Universitas Mercu Buana, Jakarta, Indonesia and the e-Government and e-Business Research Lab, Universitas Indonesia, Depok, Indonesia e-mail: handrie.noprisson@mercubuana.ac.id.

Manuscript received March 31, 2019; accepted January 12, 2022.

The visual cortex is one part of the brain that has a role in processing information that enters through the five senses. In the visual cortex has cells that are very sensitive to the receptive field (receptive field) is a small part of the area in the input space. CNN works on input space locally and is good at searching for strong local spatial correlations in an image [10].

CNN belongs to one type of Multilayer Neural Network. Like other neural network theories, CNN is trained to use back-propagation algorithms. CNN is designed to recognize the visual pattern of image pixels directly by minimizing the pre-process [10].

The advantages of CNN are able to recognize patterns that have different variations, as well as robust against simple geometric distortion and transformation. CNN uses three architectural ideas to know some degree of shift, scale and distortion invariance. These three architectural ideas are respectively local receptive, shared weights, and spatial or temporal subsampling.

An example of one of the CNN architectures used for character recognition is the LeNet-5 architecture, shown in Fig. 1.

The CNN network consists of a set of layers where each layer consists of one or more sub-layers. Each sub-layer unit receives input from the surrounding environment that resides on the previous layer. Using the local receptive field, CNN can extract basic visual features such as the orientation of edges, endpoints, and corners. In the next layer, these features are combined to detect features at a higher level.

Figure 2 shows a standard neural network with two hidden layers before Dropout (a) is applied, and after a dropout (b) is applied. Implementing a dropout on a neural network is equivalent to reducing the number of networks within a neural network to a thinned network. This thinner network consists of all network units that survive the dropout process. Neural network with n units can be seen as a collection of possible thinner $2n$ networks. The whole network shares the weights so that the number of parameters remains $O(n^2)$ or less than $O(n^2)$ [12].

III. Methodology

In this research, we will recognize Batik pattern by using CNN optimized by regularisation technique namely Dropout. The complete research phases can be seen in Fig. 3.

This research work is completed through five phases:

1) First, we prepared Batik Image as dataset based on the defined criteria.

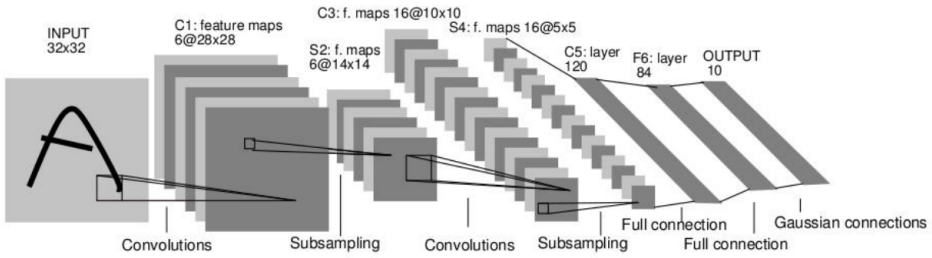

Fig. 1. Architecture of LeNet5 [10].

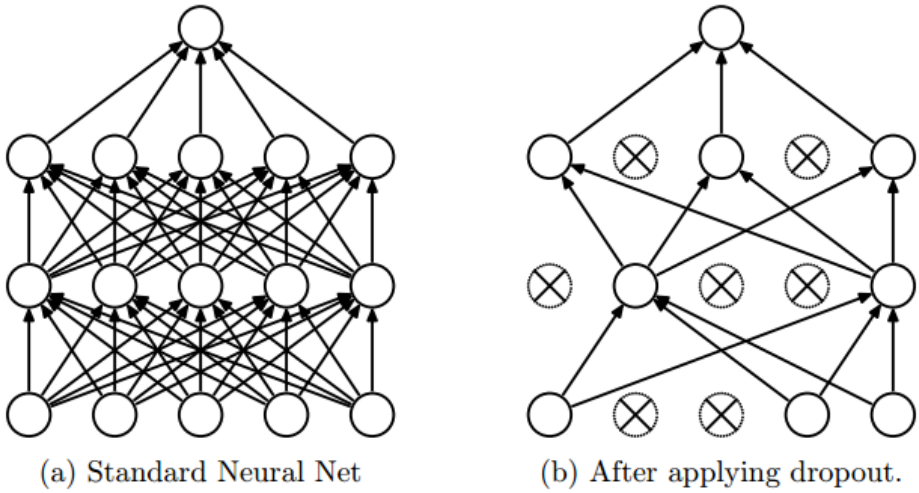

Fig. 2. Dropout Neural Net Model [12].

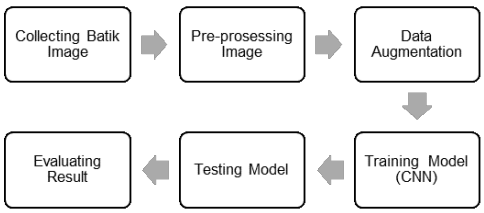

Fig. 3. Phases of research.

2) Second, we pre-processed Batik image in order to be ready for data augmentation. This phase is completed by increasing the contrast of the image and converting the image into a suitable format.
3) Third, we conducted data augmentation to generate new data. The purpose of this phase is to enrich data for training. Data augmentation process is elaborated below:
   a) Image rotation randomly in (0-180) degree.
   b) Image translation randomly in vertical or horizontal.
   c) Shearing transformation.
   d) Zoom range to zoom out image randomly.
   e) Flip horizontal randomly.
   f) Fill mode to fill out new pixel after image rotation or translation.
4) Fourth, we trained a model using CNN with dropout regularisation.
5) Fifth, we tested a model to know the performance of the proposed model.
6) Sixth, we evaluated result by computing error rate in predicting data testing.

IV. Experimental Results

A. Experimental Setup

The experiment to classify batik pattern was conducted in Ubuntu 14.04 LTS 64-bit on a PC with Processor Intel® Core™ i7-6500U CPU @ 2.50GHz × 4, Memory DDR2 RAM 8 GB and Hard Disk 160 GB. In this study, we implemented experiments in Python using Keras deep learning library. We implement Deep Learning method, i.e., Convolutional Neural Network with Dropout regularisation. We performed cross-validation and divided the dataset into training and testing data. The testing data took 30% of the dataset while 70% for data training. Then the data training samples are trained using deep learning methods to generate the model.

B. Feature Extraction

In this study we extract four texture features; they are Gabor filters, Log Gabor filters, GLCM, and LBP. The feature

TABLE I
CLASSIFICATION REPORT

Class	Precision	Recall	F1-score	Support
1	0.76	0.90	0.83	448
2	0.50	0.17	0.25	72
3	0.70	0.59	0.64	97
4	0.74	0.66	0.70	186
5	0.77	0.77	0.77	147
avg / total	0.73	0.75	0.73	950

extraction and fusion process successfully generated a feature vector with a length of 217 for each image. After the feature extraction and the feature fusion, we make the feature selection and reduction process using PCA.

C. Feature Selection

We experiment using PCA features selection and reduction to select the best component features. In this experiment we used the same batch_size and epoch value; we used batch_size = 32 and epoch = 50. We experimented PCA using 25, 50, 100, 150, 200, 210, and all component to select the best component of the feature vector. The PCA features selection and reduction result shown in Table VI.

From Table VI, the best accuracy in this experiment is 0.7463 achieved when the number of components (N) is 50. From this feature selection and reduction process, we can reduce the vector feature to new feature vector with length 50, so that we can decrease the computational time, and gives more accuracy than using all features. Table I presented the classification report in each class for the $N = 50$.

As presented in Table I, the class "1" has the best score in recall and f1-score, while the class "5" has the best score in precision.

D. Tuning Hyperparameters

In this scenario, we conducted the parallel processing to tuning the hyperparameters. As a result in an experiment using PCA features selection, in this experiment, we used 50 features component that achieved the best result in PCA features selection. In this experiment, we used Grid Search algorithm that provided in the GridSearchCV class in Scikit-learn. We imported KerasClassifier and GridSearchCV in our code; this will allow us to use sklearn's Grid Search to tuning hyperparameters in parallel processing. The hyperparameters that we tuned they are: epoch, batch_size, training optimisation algorithm, learning rate, network weight initialisation, and neuron activation function.

1) Epoch and batch size: The result in tuning epoch and batch_size gives the best accuracy 75.56% and std score 0.006 using some epoch 200, and batch_size 20. The computational time in this experiment took 264.059723 seconds.

2) Optimiser algorithm: The result in tuning optimiser algorithm gives the best accuracy 74.29% and std score 0.014 using Adam optimiser algorithm. The computational time in this experiment took time 323.357711 second.

TABLE II
EPOCH AND BATCH SIZE

Epoch/ batch_size	Epoch=50	Epoch=100	Epoch=200
Bs=20	0.693315 (0.010280)	0.740289 (0.003556)	0.755646 (0.006093)
Bs=32	0.688799 (0.014939)	0.719964 (0.008004)	0.740289 (0.016122)
Bs=40	0.666667 (0.009581)	0.746612 (0.007978)	0.741192 (0.009581)

TABLE III
OPTIMIZER ALGORITHM

Optimizer	Accuracy
Adadelta	0.447606 (0.010039)
Adam	0.742999 (0.014009)
Adamax	0.729901 (0.013106)

TABLE IV
LEARNING RATE

Learning rate	Accuracy
0.001	0.738482 (0.012320)
0.01	0.447606 (0.010039)
0.1	0.447606 (0.010039)
0.2	0.447606 (0.010039)
0.3	0.347787 (0.135001)

TABLE V
NETWORK WEIGHT INITIALIZATION

Init_mode	Accuracy
uniform	0.735321 (0.009011)
lecun_uniform	0.755194 (0.005569)
normal	0.747516 (0.005110)

3) Learning rate: Table IV presented the result in tuning learning rate; the result shows that learning rate 0.001 gives the best accuracy and the processing time in this experiment took 288.3194 seconds.

4) Network weight initialisation: Table V shows that the best network weight initialisation on the dataset is lecun_uniform, the accuracy achieved is 75.51% and the processing time took 270.827701 seconds.

TABLE VI
PCA FEATURES SELECTION AND REDUCTION RESULTS

Parameter	**all_component = 217**	$N = 25$	$N = 50$	$N = 100$	$N = 150$	$N = 200$	$N = 210$
Accuracy	0.70631	0.71263	0.74631	0.72526	0.73157	0.72210	0.71578
F1-Score	0.67940	0.68884	0.72897	0.71226	0.70985	0.70005	0.69216
Precision	0.70015	0.71908	0.73148	0.70936	0.73157	0.71836	0.70187
Recall	0.70631	0.71263	0.74631	0.72526	0.71571	0.72210	0.71578
Kappa	0.55153	0.57890	0.62013	0.59991	0.59605	0.58703	0.56700
Time (second)	197.766	33.487	64.212	90.866	135.505	176.733	185.864

TABLE VII
NEURON ACTIVATION FUNCTION

Activation function	**Accuracy**
sigmoid	0.447606 (0.010039)
hard_sigmoid	0.447606 (0.010039)
linear	0.759711 (0.003556)

5) Neuron activation function: Experiment in tuning the neuron activation function gives the result accuracy 75.97% using the linear function. The computational time in this scenario took 295.212576 s.

V. CONCLUSION

From the experiment that we conducted, the result shows that the feature extraction, selection, and reduction give the accuracy more significant than raw image dataset. The feature selection and reduction also can improve the execution time. The parameters tuning and using the dropout in layers can improve the accuracy more. Parameters value that gave best accuracy are: epoch = 200, batch_size = 20, optimizer = Adam, learning_rate = 0.01, network weight initialization = lecun_uniform, neuron activation function = linear.

ACKNOWLEDGMENT

This research was fully supported Centre of Research, Universitas Mercu Buana through an internal research grant (named penelitian internal).

REFERENCES

[1] A. Wulandari, *Batik Nusantara : Makna Filosofis, Cara Pembuatan & Industri Batik*. Yogyakarta, Indonesia: Penerbit ANDI, 2011.
[2] S. Samsi, *Techniques, Motifs, Patterns Batik Yogya and Solo*. Titian Foundation, 2011.
[3] I. Nurhaida, R. Manurung, and A. Arymurthy, "Performance comparison analysis features extraction methods for batik recognition," in *International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2012, pp. 207–212.
[4] H. Noprisson, E. Hidayat, and N. Zulkarnaim, "A preliminary study of modelling interconnected systems initiatives for preserving indigenous knowledge in indonesia," in *International Conference on Information Technology Systems and Innovation (ICITSI)*, 2015, pp. 1–6.
[5] M. Sadikin and I. Wasito, "Toward object interaction mining by starting with object extraction based on pattern learning method," in *Proceedings of the Pattern Learning Method Asia-Pacific Materials Science and Information Technology Conference (APMSIT'14)*, 2014.
[6] I. Nurhaida, A. Noviyanto, R. Manurung, and A. Arymurthy, "Automatic indonesian's batik pattern recognition using sift approach," *Procedia Computer Science*, vol. 59, pp. 567–576, 2015.
[7] R. Akta, "Batik motif classification using scale invariant feature transform method," Ph.D. dissertation, Universitas Indonesia, 2012.
[8] K.-S. Loke and M. Cheong, "Efficient textile recognition via decomposition of co-occurrence matrices," in *IEEE International Conference on Signal and Image Processing Applications*, 2009, pp. 257–261.
[9] L. Rahadianti, R. Manurung, and A. Murni, "Clustering batik images based on log-gabor and colour histogram features," *University of Indonesia*, 2010.
[10] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.
[11] D. Hubel and T. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of physiology*, vol. 195, no. 1, pp. 215–243, 1968.
[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.