DOI: 10.14421/ijid.2020.09207 IJID (International Journal on Informatics for Development), *e-ISSN: 2549-7448*
Vol. 9, No. 2, 2020, Pp. 100-110

A Comparative Study of Transfer Learning and Fine-Tuning Method on Deep Learning Models for Wayang Dataset Classification

Ahmad Mustafid[1], Muhammad Murah Pamuji[2], Siti Helmiyah[3]
[1]Department of Computer Science, Technische Universität Kaiserslautern, Kaiserslautern, Germany
[2]Department of Electrical Engineering and Information Technology, Gadjah Mada University, Yogyakarta, Indonesia
[3]Master of Informatics Engineering, Ahmad Dahlan University, Yogyakarta, Indonesia
[1]ahmad.mstfd@gmail.com, [2]murah.pamuji@mail.ugm.ac.id, [3]siti1708048022@webmail.uad.ac.id

Article History
Received Dec 8th, 2020
Revised Dec 30th, 2020
Accepted Dec 31st, 2020
Published Dec, 2020

***Abstract*— Deep Learning is an essential technique in the classification problem in machine learning based on artificial neural networks. The general issue in deep learning is data-hungry, which require a plethora of data to train some model. *Wayang* is a shadow puppet art theater from Indonesia, especially in the Javanese culture. It has several indistinguishable characters. In this paper, we tried proposing some steps and techniques on how to classify the characters and handle the issue on a small *wayang* dataset by using model selection, transfer learning, and fine-tuning to obtain efficient and precise accuracy on our classification problem. The research used 50 images for each class and a total of 24 *wayang* characters classes. We collected and implemented various architectures from the initial version of deep learning to the latest proposed model and their state-of-art. The transfer learning and fine-tuning method showed a significant increase in accuracy, validation accuracy. By using Transfer Learning, it was possible to design the deep learning model with good classifiers within a short number of times on a small dataset. It performed 100% on their training on both EfficientNetB0 and MobileNetV3-small. On validation accuracy, both gave 98.33% and 98.75%, respectively.**

Keywords*— *Artificial Intelligence, Deep Learning; Fine Tuning; Transfer Learning; Wayang.

1 INTRODUCTION

Deep Learning is an essential technique in the classification problem in Machine Learning based on Artificial Neural Networks. Deep learning represents the learning method which for several years was popularly used because of the ability to study representation in depth. It uses multiple layers in the convolution neural network to produce computational models.

Some Deep Convolutional Neural Network architectures have been proposed since AlexNet [1] in 2012 that demonstrated significant results in the ImageNet challenge. In 2014, VGG-16 designed the deeper network [2] and GoogLeNet with codename Inception proposed with the dense module/blocks in their architecture [3]. In the following year, Deep Residual Learning (ResNet) implemented the skip connection and additionally used the batch normalization technique [4]. Inception-ResNets combined the inception architecture with the residual connections in 2016 [5]. In the year that followed, Xception introduced the depthwise convolution followed by a pointwise convolution (depthwise separable convolution layers) [6]. In 2018, DenseNet designed the neural network where each layer in the convolution network is connected to every other layer, directly [7]. NASNet, in the same year, learned the model architectures directly on the dataset of interest. Searching on the smaller dataset for architectural building blocks, subsequently transfer it into the larger dataset [8]. Some architectures were proposed for mobile and embedded vision applications like MobileNet [9], MobileNetV2 [10], and MobileNetV3 [11]. In 2020, EfficientNet introduced a novel scaling method that can uniformly scale the dimension of depth, width and resolution by using compound coefficients [12].

The general issue in deep learning is data-hungry [13], [14], [15]; which means that the deep learning architectures require a plethora of data to train some models. We need to collect lots of data for each class that wants to be modeled. It will take many resources and times to collect and validate some data. The issue comes when we just have unavoidable limited data to train, is it possible to look for a model that even trained with a small number of data for many classes, with good accuracy within a short number of times?

Wayang or *Wayang Kulit* is a shadow puppet art theater from Indonesia, especially in the Javanese culture. It is one of the Javanese performing arts that has a great value in Javanese society, it represents human life in the dimensions of the art of prestige [16]. The stories of *wayang* have the ability to absorb and reflect Javanese Culture due to the natural improvisational performance by *Dhalang* (the master puppeteer). Forty or fifty different characters may be required in a single play. Each character has his own personality, voice quality, and movement style that is related to his physical characteristics [17]. The character of the *wayang* is crucial to be studied by the current generation. It can help to comprehend and appreciate their life, values and culture. But, the reality in our society, especially for the younger age group are unfamiliar with the figures of the *wayang*. The figures have several indistinguishable characters from each other, if we don't have any knowledge of it. Therefore, the implementation of recognizing the *wayang* characters are exceptionally important to do.

Several studies on classification in *wayang* aspect have been conducted like in their *gamelan* music pattern [18] and emotion recognition [19]. The similar studies on the classification of *wayang* characters have been performed using Convolution Neural Network [20] and MLP with GLCP Feature Extraction [21]. However, those studies contain limitations that only recognize five characters with average accuracy and require large data for each class.

Transfer learning is a key technique in Machine Learning to overcome the fundamental problem of the insufficient training data. It tries transferring the knowledge from the source to the target domain by relaxing the assumption that the training and the test data must be independent and identically distributed (i.i.d.)[22]. Several studies showed were able to handle this issue by using deep transfer learning schema [23], [24] even though on small dataset [25], [26], [27], [28].

We collected and implemented various architectures from the initial version of deep learning to the latest proposed model and their state-of-art. This research wants to find and compare deep learning architectures that best fit our *wayang* dataset. We tried proposing some steps and techniques on how to classify the characters and handle the issue on a small *wayang* dataset by using model selection, transfer learning, and fine-tuning and to obtain efficient and precise accuracy on our classification problem.

2 METHOD

This research consists of several stages like data collection, implementation of deep learning algorithms in training, transfer learning using pre-trained models, fine-tuning, selection, and comparison. These stages of the research can be seen in Figure 1.

2.1 Dataset

We collected 24 prominent characters based on a survey that has been conducted on several people about the popularity of *wayang* figures [29]. We selected the well-known figures before collecting the data.

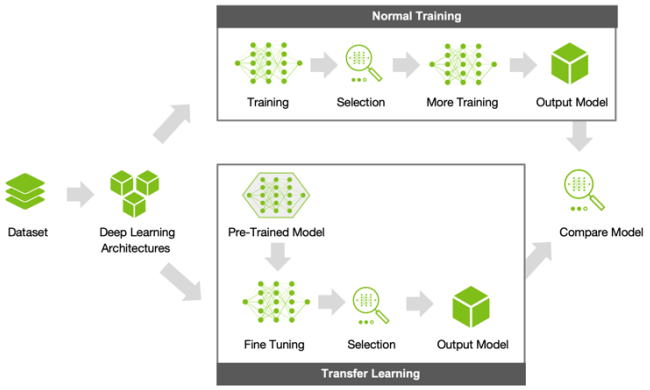


Figure 1. The research procedure stages

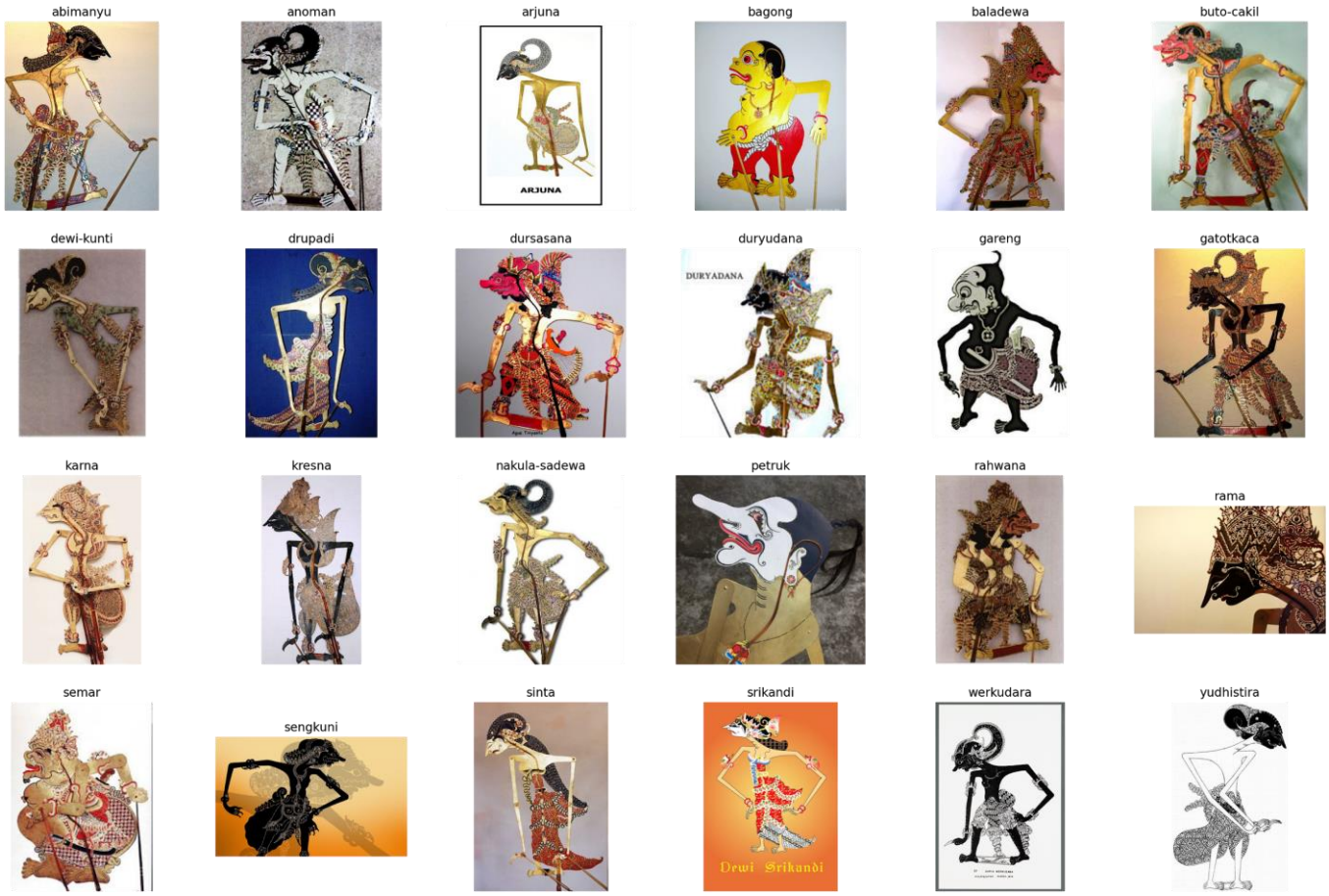


Figure 2. Example of *wayang* dataset with 24 character classes.

The data employed in this paper are called *wayang* dataset which was taken from various sources like books [30], websites, and free-to-use pictures from google images.

The research used 50 images for each class and a total of 24 wayang characters classes. The total combined data for training purposes are 1200 pictures. Figure 2 illustrates the example of the *wayang* dataset.

2.2 *Deep Learning Architectures*

Deep Learning is based on Artificial Neural Networks (ANN) which their inspiration was adopted from biological brain function [31]. It is composed of multiple processing layers to learn representations of data with multiple levels of abstraction in their computational models [32]. Deep learning is a subset of Machine learning (ML) which is also an integral part of Artificial Intelligence (AI). It is based on the Convolutional Neural Network (CNN) with deep layers or use multiple layers on their network architectures. Moreover, It can be used in multi-label classification problems.

In this research we used 16 Deep Convolutional Neural Network architectures in total, like AlexNet, VGG-16, GoogLeNet, ResNet, Inception-Resnet, Xception, DenseNet (121, 201), NASNet, MobileNet, MobileNetV2, MobileNetV3Small, MobileNetV3Large, EfficientNetB0, EfficientNetB2, and, EfficientNetB7. We briefly explained each architecture of the deep learning below.

2.2.1 *AlexNet*

AlexNet architecture designed by Alex Krizhevsky [1]. This is the very first deep learning architecture that gained significant results in the ImageNet challenge. It is the first neural-network that implements Rectified Linear Units (ReLUs) as activation functions. This network contained five convolutional layers with three max pooling utilized in the feature extraction part and three fully connected layers with Softmax activation function on the last network for classification.

2.2.2 *VGG-16*

VGG-16 developed deeper networks using small convolutional filters (3x3). It also generalizes well on other datasets. It consists of 16 trainable layers which include 13 convolutional layers, max pooling, dropout, and three fully connected layers. This architecture was introduced by Karen Simonyan & Andrew Zisserman [2].

2.2.3 *GoogLeNet*

The neural net with codename Inception used the dense module/blocks in their architecture proposed by C. Szegedy et al from Google Inc. [3]. It also

improved the performance inside the architecture by allowing to increase the depth and width of the networks. The model connected some layers in parallel in some module/block instead of stacking the layers. It introduced the 1x1 convolution to reduce the feature-map dimension and also showed the Global Average Pooling that used to decrease the overfitting.

2.2.4 *ResNet*

When the network model was going deeper, it made the training process more difficult, problems that appeared were degradation which decreased the accuracy of training rapidly. To handle this, Deep Residual Learning (ResNet) implemented the skip connection and additionally uses the batch normalization technique. Layer with the skip connection called Residual Block. The skip connection allows each layer to connect the input to their output [4].

2.2.5 *Inception-ResNet*

Inception-ResNets [5] combined the Inception architecture with the Residual Block. It showed the training with residual connection increases the training speed in the Inception network architectures. In this research we used the Inception-ResNet-v2.

2.2.6 *Xception*

F.Chollet introduced Xception (extreme Inception) with depthwise separable convolution operator which a depthwise convolution followed by a pointwise convolution to the underlying Inception model. This architecture focused on improving the efficient use of model parameters rather than the increased capacity. It contained 36 convolutional layers forming the feature extraction of the network. Those layers were structured into 14 modules [6].

2.2.7 *DenseNet*

Dense Convolutional Network or DenseNet made each layer in the convolution network connected to every other layer, directly. This model showed a better parameter efficiency, because it can give comparable results with lower parameters to train. DenseNet contains the regularizing effect which can also reduce overfitting problems [7]. In this research, we use the DenseNet-121 and DenseNet-201.

2.2.8 *NASNetMobile*

Developing the neural network frequently requires significant engineering technique. NASNet tried learning the model architectures directly on the dataset of interest. This technique was expensive on the large data, to handle this it introduced the searching functionality. Searching on the smaller dataset for architectural building blocks, subsequently transfer it into the larger dataset [8].

2.2.9 *MobileNet*

MobileNet presented the efficient models for mobile and embedded vision applications. There are three various versions of this architecture. MobileNet [9] used the depth wise separable convolution and initiated the two simple global hyper-parameters that efficiently trade-off between latency and accuracy. MobileNetV2 [10], introduced the inverted residual with linear bottleneck between the layers. It took the compressed lower-dimensional space (bottleneck) as an input then expanded on the high-dimensional space. After that, filtered using the lightweight deep convolution, subsequently projected back to low-dimensional space with the linear convolution.

MobileNetV3 [11] which is the next generation of the family MobileNet architecture. The MobileNetV3 defined into two models which are MobileNetV3Small and MobileNetV3Large, These models were targeted to the low and high resources devices. It contained the hardware aware network architecture search (NAS) and also NetAdapt algorithm. The both techniques can be combined to effectively find the optimal model for the hardware platform.

2.2.10 *EfficientNet*

Recent study explained the scaling on depth, width and resolution of the deep convolutional neural network can demonstrate a better performance. It used the compound coefficient which is the straightforward and highly effective method in uniform scaling. It can easily scale up a baseline convolution network to the target resource constraint with maintaining the efficiency [12]. In this research we used the EfficientNetB0, EfficientNetB2, and EfficientNetB7 versions.

2.3 *Normal Training*

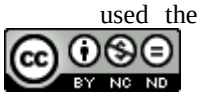
On the Normal Training stage, we train our wayang dataset using 16 architectures that have been pointed out. We separated our dataset to be 80% of the images for training and 20% for validation. Therefore, it would be 960 and 240 data respectively.

The training operated the same initial 30 epochs for all the models, after that we used finer-control to shuffle the dataset randomly. Next, we continued the training by adding more 30 epochs to identify the effect of the finer-control.

Finer-control represented the steps to configure the dataset to obtain better performance. It shuffled the dataset, created some batches, and prepared the batched data to be available as soon as possible. The focus on this finer-control is to make the dataset to be well shuffled. Because, the poor shuffling would affect the accuracy of the training and validation process.

In the selection process, we selected architecture that demonstrated significant progress. We also separated the models that did not show the significant increase in their accuracy as well as the validation accuracy. The overfitting models were also removed in this stage.

After obtaining the possible architectures that potentially increase their accuracy, we trained more with 200 epochs to the selected models. So, the result of these training steps would be able to explain the behaviour of each model on this stage.



2.4 Transfer Learning

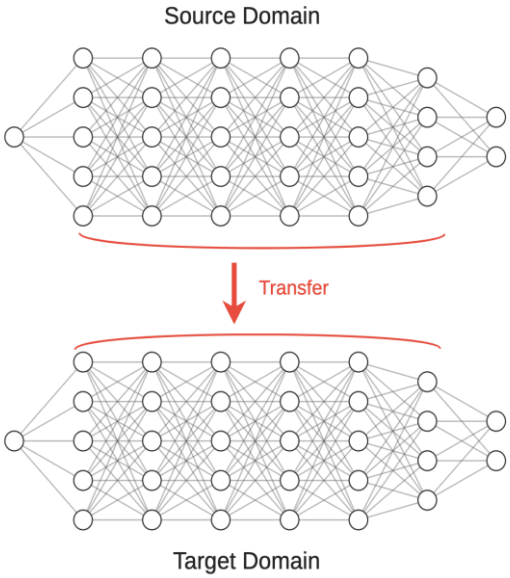


Figure 3. Schematic view of transfer learning. The weights are transferred from the Source Domain to the Target Domain.

Transfer Learning or network-based deep transfer learning refers to the reuse of the partial network that is pre-trained in the source domain, including its network structure and connection parameters, then transfer it to be a part of a deep neural network which is used in the target domain [22]. The resulting weights parameter or pre-trained model from the source domain subsequently used in the target domain. Figure 3 demonstrating how the transfer learning is performed.

For all models that we implemented, we eliminated the fully-connected layer on the last layer which contains 1000 classes from the pre-train ImageNet. On the last layer in each architecture, we added additional layers like a Global Average Pooling layer and a Dense Classifier with 24 classifiers as an output. The output for each model would contain 24 classes to perform classification.

2.4.1 Pre-Trained Imagenet

The Transfer Learning process required Pre-Trained models to be transferred from the source to the target domain. Each model architecture needs a unique pre-trained model which matches their network to perform the transfer learning process. The aim of the transfer learning is to employ the model that has been trained (pre-trained) using a large dataset containing similar problems then apply it to another task. The pre-trained models will help to make generalization of the data.

In this research we used the ImageNet dataset [33] for the Pre-Trained ImageNet. It contained 1.2 million images with 1000 classes. However, our *wayang* dataset consisted of just 1200 images with 24 classes which means that there are insufficient images to train in deep neural networks. Hence, we used the pre-trained weight from ImageNet.

2.4.2 Fine-Tuning

Fine-Tuning is a way of making finer adjustments to improve the performance and accuracy of the pre-trained network, previously. In this research, each model was fine-tuned directly, right after they received the pre-trained model. In other words, the architectures were fine-tuned from the initial of the training or it ready to be fine-tuned from the beginning of the training.

If we made the networks were not to be trainable, this architecture would be a feature extraction. However, if we constructed the network to be trainable, it is considered as fine-tuning. We didn't use the feature extraction in this step, because we made the pre-trained network to be trainable on our *wayang* dataset.

2.5 Compare Model

After running the training on the normal training stage and the fine-tuning on the transfer learning stages, we received the results of training accuracy and validation accuracy of all the models. Furthermore, we visualized the results for each model and tried to investigate all the architecture models. The analysis was performed to obtain good and bad accuracy. It also detected the model that overfitted which showed better accuracy on the training result but worse on the validation result.

3 RESULT AND DISCUSSION

The experiment in this study and assessment of the state-of-art deep convolutional neural network architectures for the wayang dataset was done. Our focus was on searching for the best architectures that performed well on our dataset. The result would be discussed below.

3.1 Experiment Setup

The experiments were performed on a Graphics Processing Unit (GPU). GPU: TeslaT4, NVIDIA CUDA 2,560 cores. 13GB RAM, Intel(R) Xeon(R) CPU @ 2.20GHz (1 core, 2 threads). Google Collab as an environment and machine in this research (colab.research.google.com). In addition, it uses Python as the programming language that is reasonable and comfortable to the machine learning tasks which also supported the number of deep learning algorithms. This research also used Keras (keras.io) and TensorFlow (tensorflow.org) which are simple and powerful neural networks libraries that have lots of building blocks to create the deep learning model architectures, this also has pre-trained networks from ImageNet.

3.2 Result of the Normal Training

In this study, normal training and an assessment of the deep convolutional neural network or deep learning network architectures was done. Our focus was to select the most significant architectures and models that potentially improve their performance. We selected based on their accuracy and validation accuracy measurements result.

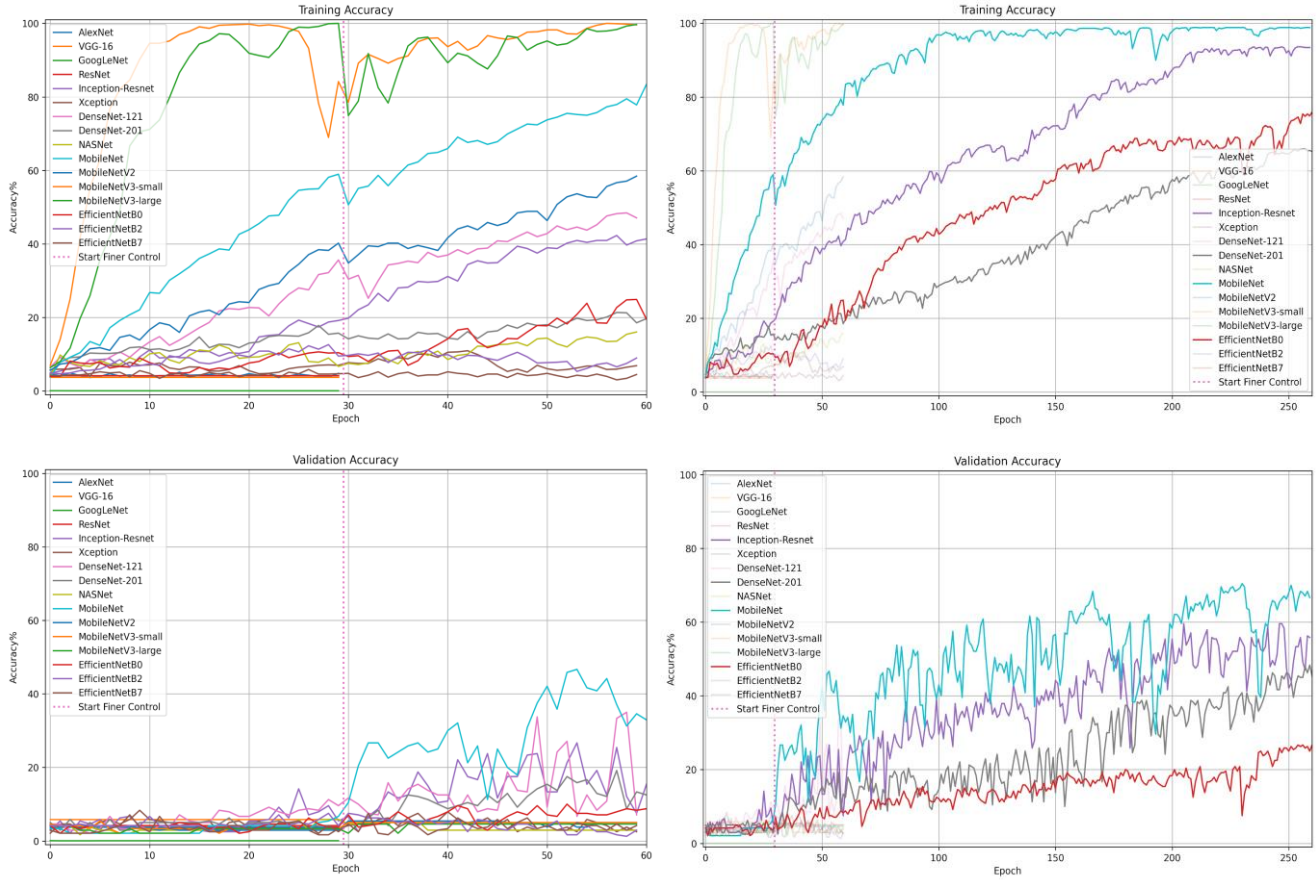


Figure 4. Deep Neural Network Architectures performance (accuracy and validation accuracy), (left) showing the first 60 epochs of training steps with the finer-control process on the 30th epoch, (right) showing the more 200 epochs of training steps with some selection criteria

The accuracy measurement is denoted in the Eq. 1. The value will be marked as accurate when the Predicted Value is correctly matched with the True Value on the one-hot label. The Accuracy is then measured by dividing the number of accurately predicted records (*TP = True Positive; TN = True Negative*) by the total number of records (*TP = True Positives; TN = True Negatives; FP = False Positives; and FN = False Negatives*).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \qquad (1)$$

The first measurements were on the first 60 epochs which showed in Figure 4 on the left-hand side. Next, the selection was performed which removed or didn't continue the training for the bad performance and overfitting model. We also reduced the color transparency on the graphic of the bad performance architecture and only pointed out the good architectures. It showed in Figure 4 on the right side that illustrated the training accuracy and validation accuracy after additional 200 epochs.

Based on the graphic in Figure 4. It was clear most of the models had a bad validation accuracy on the first 30 epochs. However, the training result indicated an increase in accuracy. After the finer-control caused some effect to several models to have increment on their validation accuracy.

Next, each model was executed until the 60 epochs and performed some selection stage, where removed the model that had bad results and also overfitting. After some selection, we got 4 good or potentially increased models. The models were Inception-Resnet, Dense-Net-201, MobileNet, EfficientNetB0. We plotted and highlighted the results of those selected models in Table 1 and in Figure 4 on the right side.

The outcome illustrated that MobileNet had the best among the other results on the normal training stage. It obtained an accuracy of 98.85% and 66.67% in its validation accuracy. It also ran fastest among the other architectures with just 6 seconds per epoch of training. But we still seek some improvement to make enhancements on the validation accuracy. We subsequently resumed the research by using the transfer learning and fine-tuning technique.

Table 1. Accuracy, validation accuracy, loss and validation loss of training and its execution time per epoch on the Normal Training stage

Model and specifications			Normal Training								
			60 epochs				260 epochs				
Model	Input shape	Params	Training accuracy%	Validation accuracy%	Training Loss	Validation Loss	Training accuracy%	Validation accuracy%	Training Loss	Validation Loss	Time ~secs /epoch
Inception-Resnet	299x299x3	54.4 M	40.83	7.50	2.8435	3.1705	93.44	55.83	2.3140	2.6957	35
DenseNet-201	224x224x3	18.4 M	18.54	13.33	3.0572	3.1103	65.52	48.33	2.5892	2.7734	17
MobileNet	224x224x3	3.3 M	77.81	34.58	2.4803	2.9101	98.85	66.67	2.2582	2.5766	6
EfficientNetB0	224x224x3	4.1 M	24.90	8.33	2.9986	3.1572	74.90	25.00	2.4983	2.9853	6.5

3.3 Result of the Transfer Learning

An assessment of Transfer Learning and Fine-Tuning methods on Deep Learning Models for *wayang* dataset was done. Our focus was to select the best performance architecture. We selected based on the evaluation of the models using accuracy metric and categorical cross-entropy loss (loss).

The Accuracy metric for normal training as well as the fine-tuning in transfer learning were calculated by using Eq. 1. The categorical cross-entropy loss (loss) was calculated using Eq. 2 and their softmax activation function in Eq. 3.

$$CE = -\sum_{i=1}^{C} t_i \log(\sigma(z)_i) \tag{2}$$

Using Eq. 2 the categorical cross-entropy loss (loss) was calculated as the sum of separate loss for each class. C denotes the number of classes, t indicates the target value, and then σ denotes the predicted value that is calculated from the Softmax activation function is in Eq. 3 which e is the standard exponential function and z is the input vector which normalized into probability distribution from the last output of the layer before applying the softmax calculation.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{3}$$

The optimization algorithm used on the Normal Training and the Transfer learning + Fine-tuning was Adam optimization. It is an algorithm of stochastic gradient descent (SGD) optimization method based on adaptive estimation of the first and second order moments. This optimizer has little memory requirement, computationally efficient, and also invariant to diagonal rescaling of gradients [34]. The equation showed in Eq. 4.

$$\theta_t = \theta_{t-1} - \frac{\alpha \cdot \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{4}$$

where:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

and where:

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$$

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2$$

and where:

$$g_t = \nabla_\theta J(\theta_{t-1})$$

The g_t is the gradient of the stochastic objective at time t. Variable m_t updates the bias of the first-moment estimate. Variable v_t updates the bias of the second raw moment estimate. While $\hat{m}$ compute the bias-corrected of the first moment estimate and $\hat{v}$ compute the bias-corrected of the second raw moment estimate. β_1 is the hyperparameter for the first momentum term and β_2 is the second momentum term. The epsilon ϵ represent a small term preventing the division by zero. Learning rate denoted as α. The hyperparameter for this optimizer sets the parameter value of learning rate (α) = 0.001, β_1 = 0.9, β_2 = 0.999, and ϵ = 1e-07.

The results of the experiment are presented in Figure 5. Each plot depicts the accuracy and entropy loss of each deep learning architecture. But we highlighted only the good result models. The others which are unperformed well, we turned those to the transparent lines on the plot. After transferring the weight and performing the fine-tuning using 60 epochs for each model, it kept their accuracy even for more epochs.

EfficientNetB0 and MobileNetV3-Small consistently performed the best among the other architectures. The DenseNet family (DenseNet-121, DenseNet-201) indicated good performance after the first two architectures. The last one was ResNet which performed adequately well.

The Finer-Control, in addition, gave a more significant effect on the DeseNet Family and the ResNet. It showed on their accuracy and validation accuracy, an increased number of the accuracy performance after the fine-controlling or shuffling data. And equally made the best two models more robust on their validation accuracy.

The Xception model indicated a good accuracy on the first 30 epochs, but it fell down after finer-control. It is because the architecture was overfitting, performed well on the training accuracy but not well on validation accuracy. Then, we can say that fine-control was able to reveal the overfitting model. The other models on the transparent line did not demonstrate a significant increase in measurement and obtained low accuracy

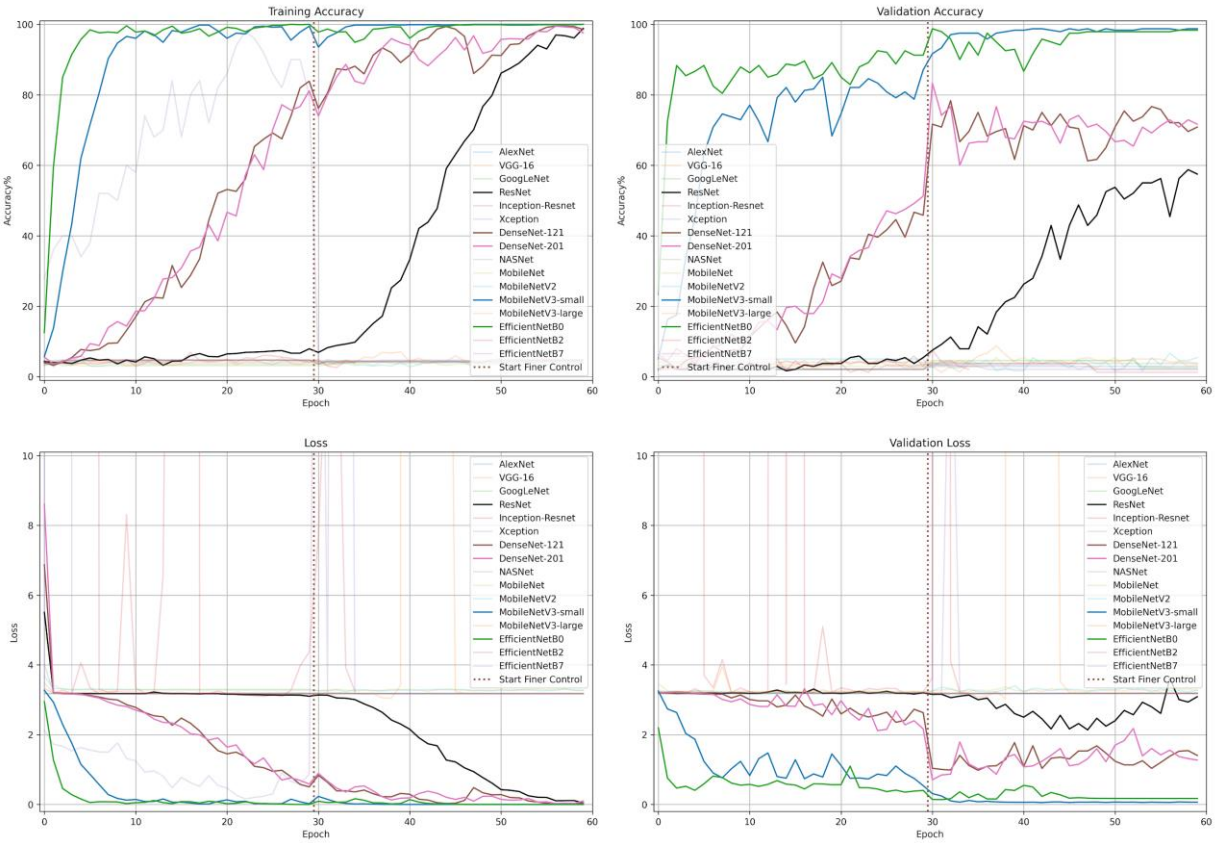


Figure 5. Transfer Learning for Deep Neural Network Architectures performance,
Showing the Accuracy, Validation Accuracy, Loss and Validation Loss.
Showing the first 60 epochs of the Fine-Tuning

Table 2 shows that the training accuracy for the Transfer Learning and Fine-Tuning model in 60 epochs can reach the higher accuracy above 98% for the selected models. The two most reliable models are MobileNetV3-Small and EfficientNetB0 which both give 100% accuracy and give the validation accuracy 98%. The second row is the family of the DenseNet (121 and 201) demonstrating the validation accuracy about 71%. The last is ResNet with just 57% in their validation accuracy.

In the case of the time to training, the EfficientNetB0 shows the most rapid time with 10 seconds per epoch. Not only performed well on the accuracy measurement, but it also demonstrated the fastest time in the fine-tuning. However, indeed showed a good accuracy similar to the EfficientNet result, the MobileNetV3-Small model shows the longest among the selected models to train which took 48 seconds per epoch.

Table 2. Accuracy, validation accuracy, loss and validation loss of training
and its execution time per epoch on the Transfer Learning stage.

Model and specifications			**Transfer Learning + Fine-Tuning**				
			60 epochs				
Model	**Input shape**	**Params**	**Training accuracy%**	**Validation accuracy%**	**Training Loss**	**Validation Loss**	**Time~secs /epoch**
Resnet	224x224x3	23.6 M	98.85	57.49	0.0437	3.0888	21
DenseNet-121	224x224x3	7.0 M	98.54	70.83	0.0427	1.4054	11
DenseNet-201	224x224x3	18.4 M	97.60	71.67	0.0998	1.2709	17
MobileNetV3-small	224x224x3	1.6 M	100	98.75	0.0012	0.0655	48
EfficientNetB0	224x224x3	4.0 M	100	98.33	1.58E-05	0.1717	10

We continued seeing the representation of the data by plotting it using our *wayang* dataset and trying to predict using 1-fold cross validation using the best four models that we had. The confusion matrix in Figure 7 displayed the true data and predicted data. Based on the graphic, we would be able to see where the model failed to predict or train the data representation on the cross validation.

We also calculated the time for each model to predict the dataset without performing the training, just prediction. The evaluating the dataset total 1200 images in one run (1-fold for all dataset) for each model need several times below to finish, sorted based on the fastest model:

- MobileNetV3-small: (total: 31.3 s)

  CPU times: user 30.5 s

  sys: 755 ms

  Wall time: 16.5 s
- EfficientNetB0: (total: 2min 08s)

  CPU times: user 2min 6s

  sys: 2.06 s

  Wall time: 1min 8s
- DenseNet-121: (total: 5min 03s)

  CPU times: user 5min

  sys: 2.85 s

  Wall time: 2min 35s
- DenseNet-201: (total: 7min 42s)

  CPU times: user 7min 38s

  sys: 4.42 s

  Wall time: 3min 57s

At that time, we wanted to visualize how deep learning obtained their decision by plotting using Grad-CAM [35]. It is visual explanations on deep networks using Gradient-Based Localization and class activation heatmap technique for an image classification model that explains the machine behaviour.

Figure 6. Grad-CAM class activation visualization on two characters

We investigated on the image containing two characters, the Grad-CAM showing the 'visual explanation' how they selected the left character rather than the right character. In Figure 6 explained why the model provided the prediction decision by plotting using this visualization technique.

3.4 Discussion

The experiment on the Normal Training stage determined maximum accuracy on training was just 77% and under 40% on their validation accuracy and it took a long time to train and use more epochs.

Transfer Learning method was an effective technique to handle the data-hungry and also increase the accuracy of the architectures. EfficientNetB0 and MobileNetV3-small demonstrated a remarkable result on the Transfer Learning stage. Both models provided 100% on their training accuracy. On validation accuracy, EfficientNetB0 and MobileNetV3-small gave 98.33% and 98.75%, respectively.

The data-hungry issue on deep learning could be handled by using transfer learning schema [23], [24] even though on small dataset [25], [26], [27], [28]. Our contribution is that; we can showed better and efficient model and more labels classification with limited number of data rather than previous study [20], [21]. From this research, we are able to provide insight and comparison between models in terms of performance and efficiency on more label classification when facing data-hungry problems.

Moreover, if we want to produce a system that implements the architectures which rarely update the weight parameters and needs only to test some new data, the MobileNetV3-small is a suitable choice, because it shows fastest on the prediction time with a total 31.3 seconds for the 1200 data. However, if we continuously update our model parameters, it will be better to use EfficientNetB0. Because, it maintains an efficient time to train data with just 10 secs/epoch and additionally provides good accuracy. The EfficientNetB0 is an effective architecture on the very dynamic-model in the implementation that intensively changes their weight parameters.

It is straightforward to get good accuracy on the *wayang* dataset. It is due to the data just the representation of the 2D object characteristics which means the character will remain the same. The effect on the data when changing their pose of the character and slightly be affected by the camera or the environment when capturing some pictures.

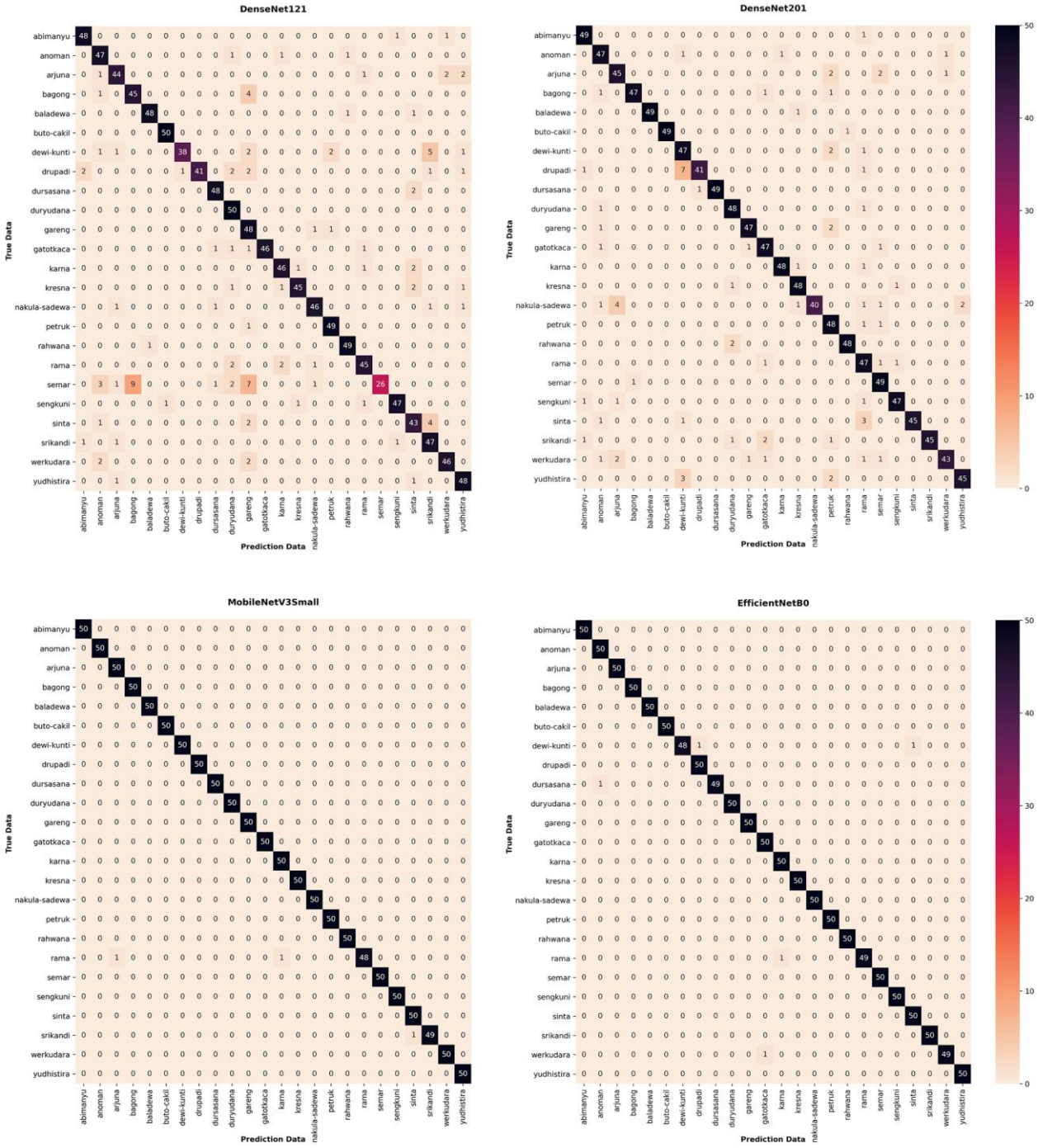


Figure 7. Confusion Matrix of 1-fold cross validation in *wayang* dataset using the selected model

4 CONCLUSION

In this study, evaluation and comparison of the Deep Learning architectures by using the Transfer Learning and Fine-Tuning method were performed. The total architectures were 16 models with two different stage approaches, normal training and transfer learning + fine-tuning.

By using Transfer Learning, it is possible to create the deep learning model with good accuracy in a short period of training, even using a small number of *wayang* dataset for many classes on the classification problems. The Results show that EfficientNetB0 and MobileNetV3-small architectures demonstrated a remarkable result on the training using the transfer learning method. Both models provided 100% on their training accuracy. On validation accuracy, EfficientNetB0 and MobileNetV3-small gave 98.33% and 98.75%, respectively.

However, the transfer learning method contains some limitations, because most of the models cannot be varied widely on the test data, thus this provides lower accuracy. It sometimes made the model overfit excessively early.

ACKNOWLEDGMENT

We would like to express our gratitude to Almarhum Ki Seno Nugroho (ꦱꦺꦤꦺꦴ ꦤꦸgꦫꦺꦴꦃ) for his dedication, devotion, and continual shift to maintain the existence of the *wayang* and how he has equally performed the sacred-form Javanese puppet art more entertaining for youth generation for loving and appreciating their culture.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," 2012.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[3] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, vol. 07-12-June, pp. 1–9.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, pp. 770–778.

[5] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, 2017, pp. 4278–4284.

[6] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-Janua, pp. 1800–1807.

[7] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-Janua, pp. 2261–2269.

[8] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8697–8710.

[9] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv*. 2017.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[11] A. Howard *et al.*, "Searching for mobileNetV3," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, vol. 2019-Octob, pp. 1314–1324.

[12] M. Tan and Q. V Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *36th International Conference on Machine Learning, ICML 2019*, 2019, vol. 2019-June, pp. 10691–10700.

[13] X. W. Chen and X. Lin, "Big data deep learning: Challenges and perspectives," *IEEE Access*, vol. 2. Institute of Electrical and Electronics Engineers Inc., pp. 514–525, 2014.

[14] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *J. Big Data*, vol. 2, no. 1, p. 1, Dec. 2015.

[15] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning," *Arch. Comput. Methods Eng.*, vol. 27, no. 4, pp. 1071–1092, Sep. 2020.

[16] D. Sulaksono and K. Saddhono, "Ecological Concept of Wayang Stories and the Relation with Natural Conservation in Javanese Society," *KnE Soc. Sci.*, vol. 3, no. 9, p. 58, Jul. 2018.

[17] R. Long, "The movement system in Javanese wayang kulit in relation to puppet character type: a study of Ngayogyakarta shadow theatre," University of Hawai'i, Ann Arbor, 1979.

[18] T. P. Tomo, A. Schmitz, G. Enriquez, S. Hashimoto, and S. Sugano, "Wayang Robot with Gamelan Music Pattern Recognition," *J. Robot. Mechatronics*, vol. 29, no. 1, pp. 137–145, Feb. 2017.

[19] T. P. Tomo, G. Enriquez, and S. Hashimoto, "Indonesian puppet theater robot with gamelan music emotion recognition," in *2015 IEEE International Conference on Robotics and Biomimetics, IEEE-ROBIO 2015*, 2015, pp. 1177–1182.

[20] K. Wisnudhanti and F. Candra, "Image Classification of Pandawa Figures Using Convolutional Neural Network on Raspberry Pi 4," in *Journal of Physics: Conference Series*, 2020, vol. 1655, no. 1, p. 12103.

[21] M. Hamdani Santoso and D. Ayu Larasati, "Wayang Image Classification using MLP Method and GLCM Feature Extraction Corresponding Author," *J. Comput. Sci. Inf. Technol. Telecommun. Eng.*, vol. 1, no. 2, pp. 111–119, Sep. 2020.

[22] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018, vol. 11141 LNCS, pp. 270–279.

[23] W. Ge and Y. Yu, "Borrowing Treasures from the Wealthy: Deep Transfer Learning through Selective Joint Fine-Tuning," 2017.

[24] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," *Comput. Electron. Agric.*, vol. 161, pp. 272–279, Jun. 2019.

[25] H. W. Ng, V. D. Nguyen, V. Vonikakis, and S. Winkler, "Deep learning for emotion recognition on small datasets using transfer learning," in *ICMI 2015 - Proceedings of the 2015 ACM International Conference on Multimodal Interaction*, 2015, pp. 443–449.

[26] M. Peng, Z. Wu, Z. Zhang, and T. Chen, "From macro to micro expression recognition: Deep learning on small datasets using transfer learning," in *Proceedings - 13th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2018*, 2018, pp. 657–661.

[27] P. Cao, S. Zhang, and J. Tang, "Preprocessing-Free Gear Fault Diagnosis Using Small Datasets with Deep Convolutional Neural Network-Based Transfer Learning," *IEEE Access*, vol. 6, pp. 26241–26253, May 2018.

[28] M. Shu, "Deep learning for image classification on very small datasets using transfer learning," *Creat. Components*, Jan. 2019.

[29] T. H. Haryadi and Khamadi, "Perancangan Model Wujud Visual Tokoh Pewayangan dalam Pembentukan Identitas dan Watak Tokoh sebagai Acuan Desain Karakter dalam Karya DKV," *DeKaVe*, vol. 7, no. 2, pp. 58–79, Jul. 2015.

[30] S. Tukul and A. I. Darodjat, *Koleksi wayang kulit Museum Basoeki Abdullah*. Departemen Kebudayaan dan Pariwisata, Direktorat Jenderal Sejarah dan Purbakala, Museum Basoeki Abdullah, 2008.

[31] Y. Bengio, D.-H. Lee, J. Bornschein, T. Mesnard, and Z. Lin, "Towards Biologically Plausible Deep Learning," Feb. 2015.

[32] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[33] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[34] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[35] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 336–359, Feb. 2020.