International Journal of Aviation Science and Engineering
e-ISSN: 2715-6958 p-ISSN: 2721-5342

AVIA

Original Article

Systematic Comparison of Machine Learning Model Accuracy Value Between MobileNetV2 and XCeption Architecture in Waste Classification System

Yessi Mulyani[1], Rian Kurniawan[1,*], Puput Budi Wintoro[1], Muhammad Komarudin[1], and Waleed Mugahed Al-Rahmi[2]

[1] Department of Electrical Eengineering, Faculty of Engineering, Universitas Lampung, Jl. Prof. Sumantri Brojonegoro no.1, Bandar Lampung 35145, Indonesia
[2] Faculty of Social Sciences and Humanities, School of Education, Universiti Teknologi Malaysia, UTM Johor Bahru Skudai 80990, Malaysia
* Correspondence: *rian.kurniawan1018@gmail.com*

Received: 12 August 2022; 16 October 2022; 30 December 2022

Abstract. Garbage generated every day can be a problem because some types of waste are difficult to decompose so they can pollute the environment. Waste that can potentially be recycled and has a selling value is inorganic waste, especially cardboard, metal, paper, glass, plastic, rubber and other waste such as product packaging. Various types of waste can be classified using machine learning models. The machine learning model used for classification of waste systems is a model with the Convolutional Neural Network (CNN) method. The selection of the CNN architecture takes into account the required accuracy and computational costs. This study aims to determine the best architecture, optimizer, and learning rate in the waste classification system. The model designed using the MobileNetV2 architecture with the SGD optimizer and a learning rate of 0.1 has an accuracy of 86.07% and the model designed using the Xception architecture with the Adam optimizer and a learning rate of 0.001 has an accuracy of 87.81%.

Keywords: classification, convolutional neural network, machine learning, MobileNetV2, Xception

1. Introduction

Garbage generated every day can be a problem because some types of waste are difficult to decompose so they can pollute the environment [1,2]. Waste consists of two types, namely organic waste and inorganic waste. Organic waste is waste that comes from the remains of living organisms, while inorganic waste comes from non-living organisms. Waste that can potentially be recycled and has a selling value is inorganic waste, especially cardboard, metal, paper, glass, plastic, rubber and other waste such as product packaging [1].

Various types of waste can be classified using machine learning models. The machine learning model used for classification of waste systems is a model with the Convolutional Neural Network method. The model with this method will recognize garbage images by extracting image features and recognizing patterns according to the labels on the training data [2,3].

In the last few decades, deep learning has become a powerful tool. This is evidenced by its ability to handle large amounts of data and be able to recognize patterns from the data it manages. One of the popular algorithms for handling large amounts of data is the Convolutional Neural Network [3].

The Convolutional Neural Network architecture used to design machine learning models plays an important role. The more precise the choice of Convolutional Neural Network architecture, the better the accuracy of the model made to predict the garbage image. In addition to good accuracy,

e-ISSN: 2715-6958 p-ISSN: 2721-5342

each architecture has a different size, parameters, and cost (CPU/GPU). The Xception architecture is an architecture that has high accuracy, small size, and fewer parameters than some other architectures, so that models trained using this architecture will be used effectively and efficiently to predict images. The MobileNetV2 architecture is an architecture that has high accuracy with the smallest size and fewest parameters compared to other architectures, so that models trained using this architecture will be efficient in terms of training time and can predict the image quite well [4].

2. Materials and Methods

2.1. Machine Learning

Initially, machine learning was a term used to refer to a branch of computer science that studied algorithm design methods that were able to learn or adapt to data patterns without being explicitly programmed. Machine learning has several computational methods that can improve performance by utilizing knowledge derived from experience when learning [5].

2.2. Convolutional Neural Network

Convolutional Neural Network is one of the deep learning algorithms used to classify images. CNN operates using convolution and uses at least one layer. Artificial neural networks can generally change the input value by inserting it into a series of hidden layers. Each layer can consist of a set of neurons, where each layer will be fully related to the previous layer. Finally, the previous layers will be fully connected to generate predictions [6].

2.3. Convolutional Layer

Convolutional Layer or convolutional layer is a layer in which there are feature essence extraction operations. The essence feature extraction operation is the dot product between the weights in the filter and the image pixels to be filtered and then summed afterwards as shown in Figure 1. Convolution Operations [2].

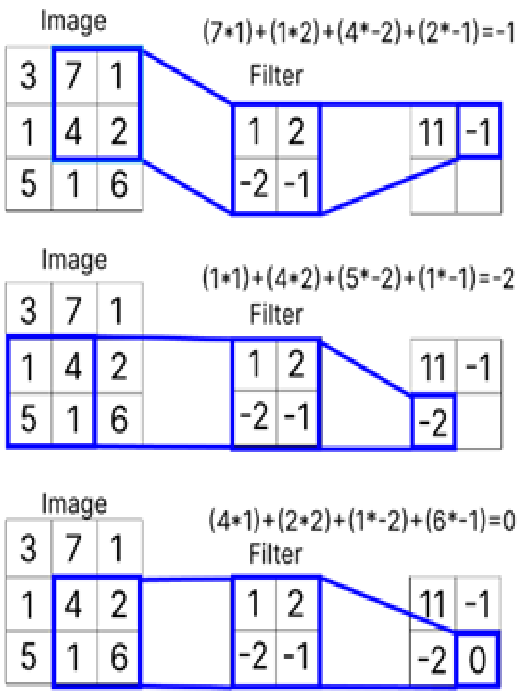

Figure 1. Convolution Operations

2.4. MobileNetV2

MobileNetV2 is one of the Convolutional Neural Network architectures that can solve the problem of computing resources on mobile devices (mobile) that are used to address the need for large computing resources. MobileNetV2 is a complementary CNN architecture to the previous version, MobileNetV1. The difference between the MobileNetV2 architecture and other architectures is the convolution layer used. The thickness of the filter layer on MobileNetV2 corresponds to the thickness of the input image used [7].

2.5. Xception

Xtreme of Inception (Xception) is a Convolutional Neural Network architecture that uses the depthwise separable convolution method. This architecture is the result of the development of the Inception architecture which has 36 convolution layers that form the basis of the feature extraction network[8].

2.6. Optimizer

Optimizer is an algorithm or method used to change the existing attributes on artificial neural networks. The attributes that can be changed using this algorithm are the value of weight and learning rate with the aim of reducing the value of loss during the training process[9].

2.7. Methods

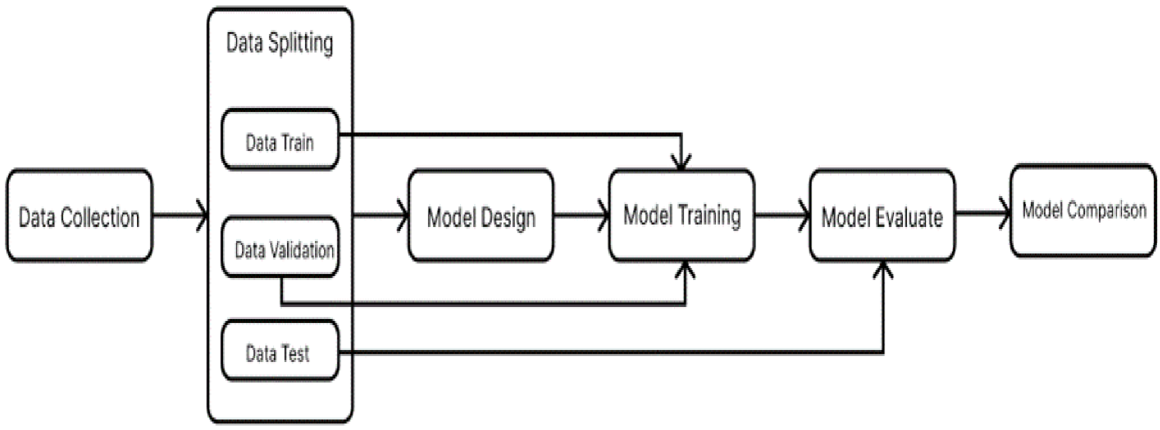

Figure 2. Research Stages

Research materials are in the form of Kaggle datasets, photo datasets obtained independently, datasets sourced from the internet, books, theses, journals, and other scientific sources obtained from various websites. The stages of this research are starting with data collection, data processing, model design, model training, model evaluation, and model comparison.

3. Results

3.1. Data Collection

The results of this study is collected and presented as a data collection system which listed in Table 1.

Table 1. Dataset Description

Name	Files Type	Source	Download Link
Garbage Classification	.jpg	Kaggle	https://www.kaggle.com/datasets/asdasdasasdas/garbage-classification
Independent Datasets	.jpg	Obtained independently	https://www.kaggle.com/datasets/asdasdasasdas/garbage-classification

The data collected consisted of 2,527 data sourced from Kaggle and 120 data produced independently by the author by taking photos directly and searching for them from the internet, so that the total data used was 2,647 data. Each data in the dataset is an image with a .jpg extension (Joint Photographic Experts Group). The following table shows the description of the dataset used, as presented in figure 3.

Figure 3. Kaggle Datasets

The image is some data for cardboard, glass, metal, paper, plastic, and trash classes obtained from the Kaggle site. There are 403 data in the cardboard class, 501 data in the glass class, 410 data in the metal class, 594 data in the paper class, 482 data in the plastic class, and 137 data in the trash class, as in figure 4.

Figure 4. Independent Datasets

The images are some data for cardboard, glass, metal, paper, plastic, and trash classes that were obtained independently. There are 20 data for each class, a total of 120 data.

3.2 Data Splitting

At this stage the author divides the collected dataset into several parts, namely training data, validation data, and testing data, with respective ratios of 0.7, 0.15, and 0.15 of the total data which has a ratio of 1 The distribution ratio aims to distribute the validation data and test data obtained independently, which is distributed more, namely 3 data for each class. If the ratio is reduced to 0.1 for validation data and 0.1 for test data, then there are only 2 scattered data for each class. The distribution ratio of 0.15 for the validation data and test data makes the data distributed 50% more. From dividing by this ratio, 1,850 training data, 402 validation data, and 395 test data are obtained, as on figure 5.

```
#Split datasets into new folder
import splitfolders
splitfolders.ratio('/content/datasets/', output="datasplit", seed=1337, ratio=(0.7, 0.15,0.15))
```

Figure 5. Data Splitting

At this stage the author divides the collected dataset into several parts, namely training data, validation data, and testing data, with respective ratios of 0.7, 0.15, and 0.15 of the total data which has a ratio of 1 The distribution ratio aims to distribute the validation data and test data obtained independently, which is distributed more, namely 3 data for each class. If the ratio is reduced to 0.1 for validation data and 0.1 for test data, then there are only 2 scattered data for each class. The distribution ratio of 0.15 for the validation data and test data makes the data distributed 50% more. From dividing by this ratio, 1,850 training data, 402 validation data, and 395 test data are obtained.

The distribution of training data, test data, and previous validation data was done randomly using the library available in Python. The author installs and uses a library called split-folders to separate datasets into different directories according to data division. The training data will be separated into a directory named "train", the test data will be separated into a directory named "test", and the validation data will be separated into a directory named "val.", as in figure 6.

```
train_generator = train_datagen.flow_from_directory(
    train_dir,
    batch_size = BATCH_SIZE,
    class_mode = 'categorical',
    target_size = TARGET_SIZE,
    subset = 'training',
    save_to_dir = save_to
    )

test_generator = datagen.flow_from_directory(
    test_dir,
    batch_size = BATCH_SIZE,
    class_mode = 'categorical',
    target_size = TARGET_SIZE
    )

validation_generator = datagen.flow_from_directory(
    val_dir,
    batch_size = BATCH_SIZE,
    class_mode = 'categorical',
    target_size = TARGET_SIZE
    )
```

```
Found 1850 images belonging to 6 classes.
Found 402 images belonging to 6 classes.
Found 395 images belonging to 6 classes.
```

Figure 6. Target and Batch Data

After dividing, we create the TARGET_SIZE and BATCH_SIZE variables where these variables are used to store pixel size values and batch data sizes for training, validation and testing processes. Then use the flow_from_directory function of the Image Data Generator class to change the image data to the size according to the TARGET_SIZE variable and put it into a data batch where each data batch has a size equal to the value of the BATCH_SIZE variable. After this process, the resulting data is 1,850 training data, 402 validation data, and 395 test data.

3.3 Model Design

The image is the design of the MobileNetV2 model used. there are 1280 neurons in the Fully-Connected layer before entering into the prediction layer which will predict 6 classes. From the picture, initially the input size was 224x224 and then convolution was carried out until it became 7x7 in size, but the depth increased, as presented in figure 7 and figure 8.

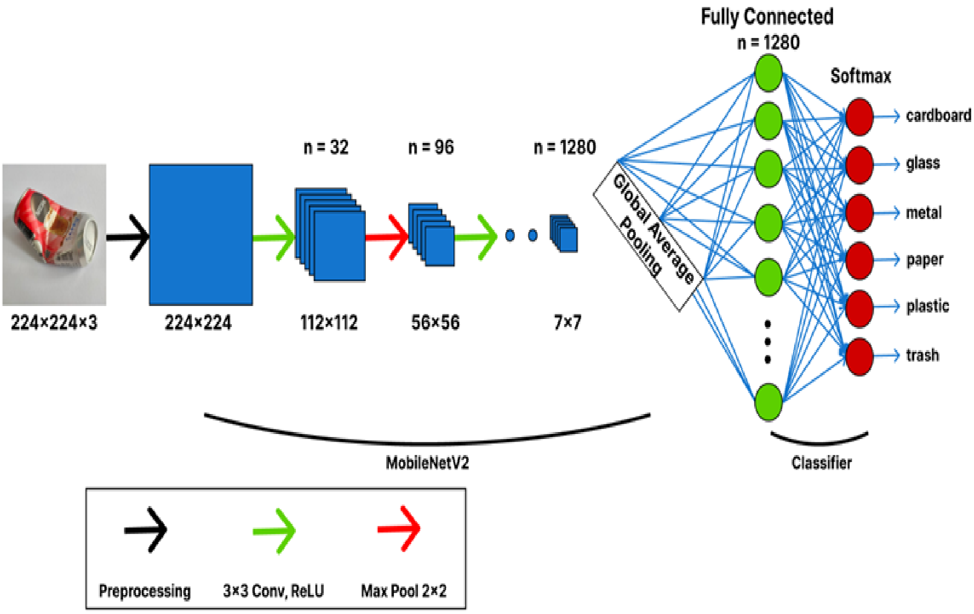

Figure 7. MobileNetV2 Model

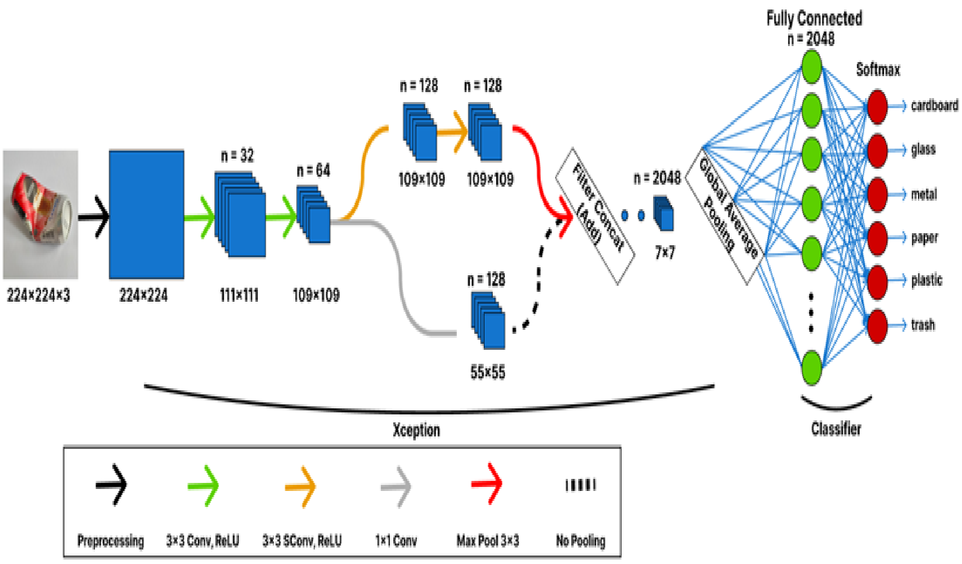

Figure 8. The Xception system

The image is the design of the Xception model used. there are 2048 neurons in the Fully-Connected layer before entering into the prediction layer which will predict 6 classes. From the picture, initially the input size was 224x224 and then the convolution was carried out until it became 7x7 in size, but the depth increased.

3.4 Model Training

At this stage, training is carried out on each model with a different optimizer and learning rate. The epoch value here has no significant effect because during the training process it uses the callbacks function to stop training when the model accuracy is optimal, as in figure 9.

```
history = model.fit(
    train_generator,
    validation_data = validation_generator,
    epochs = 100,
    verbose = 1,
    validation_steps = len(validation_generator),
    callbacks=[model_checkpoint,early_stop]
    )
```

```
Epoch 1/100
58/58 [==============================] - ETA: 0s - loss: 0.9142 - accuracy: 0.6557
Epoch 1: val_loss improved from inf to 0.58106, saving model to check_point_model.hdf5
58/58 [==============================] - 60s 985ms/step - loss: 0.9142 - accuracy: 0.6557 - val_loss: 0.5811 - val_accuracy: 0.8000
Epoch 2/100
58/58 [==============================] - ETA: 0s - loss: 0.6131 - accuracy: 0.7735
Epoch 2: val_loss improved from 0.58106 to 0.49976, saving model to check_point_model.hdf5
58/58 [==============================] - 60s 1s/step - loss: 0.6131 - accuracy: 0.7735 - val_loss: 0.4998 - val_accuracy: 0.8304
Epoch 3/100
58/58 [==============================] - ETA: 0s - loss: 0.5235 - accuracy: 0.8097
Epoch 3: val_loss improved from 0.49976 to 0.47222, saving model to check_point_model.hdf5
58/58 [==============================] - 58s 1s/step - loss: 0.5235 - accuracy: 0.8097 - val_loss: 0.4722 - val_accuracy: 0.8380
Epoch 4/100
```

Figure 9. Model Training

3.5 Model Evaluate

At this stage, the evaluation of the model is carried out using test data and the evaluate method provided by the TensorFlow framework with the aim of obtaining a model accuracy value, as in figure 10.

```
model.evaluate(test_generator, verbose=1)
```

```
13/13 [==============================] - 2s 141ms/step - loss: 0.4137 - accuracy: 0.8582
[0.41370585560798645, 0.858208954334259]
```

Figure 10. Model Evaluate

3.6 Comparison of Models

At this stage, model accuracy is obtained for each optimizer and learning rate using the evaluate method for model variables. The following table is the accuracy value obtained from each model, as presented in table 2.

Table 2. MobileNetV2 and Xception Model Accuracy

	MobileNetV2			Xception		
	lr = 0.001	lr = 0.01	lr = 0.1	lr = 0.001	lr = 0.01	lr = 0.1
Adam	82.84%	85.32%	37.81%	87.81%	82.35%	78.11%
RMSprop	83.58%	80.35%	40.55%	84.58%	83.08%	58.96%
SGD	83.08%	83.58%	86.07%	78.11%	83.83%	85.82%

Based on the table above, the accuracy value of the best MobileNetV2 model is 86.07%, where the model uses the SGD optimizer with a learning rate of 0.1. Meanwhile, the best accuracy for the

the Xception model is 87.81%, where the model uses the Adam optimizer with a learning rate of 0.001. The results of the evaluation of the model using the evaluation method are that the optimal accuracy value is obtained in the Xception model with the Adam optimizer setting and a learning rate of 0.001. Based on these results, the optimizer setting and learning rate have an effect on the accuracy model, as comparable to other results elsewhere [8,9].

5. Conclusions

The machine learning model for a garbage classification system designed using the Xception architecture with the Adam optimizer and a learning rate of 0.001 has better accuracy than the model designed using the MobileNetV2 architecture which uses the SGD optimizer and a learning rate of 0.1. The Xception model with the optimizer and learning rate has an accuracy of 87.81% while the MobileNetV2 model with the optimizer and learning rate has an accuracy of 86.07%.

References

[1] D. Kusumaningsari, J. Fisika, F. Matematika, D. Ilmu, and P. Alam, "Pemanfaatan dan Pengolahan Sampah Organik dan Non-Organik Utilization Waste."

[2] M. D. Payana, D. R. Y. TB, Z. Musliyana, and M. B. Wibawa, "Deteksi masker wajah menggunakan metode convolution neural network (CNN) meningkatkan nilai akurasi melalui aritektur layer konvolusi". Journal of Informatics and Computer Science. Vol. 8, no. 1, pp. 30–35, 2022, doi: 10.33143/JICS.VOL8.ISS1.2123.

[3] "Pengertian dan Cara Kerja Algoritma Convolutional Neural Network (CNN) - Trivusi." https://www.trivusi.web.id/2022/04/algoritma-cnn.html (accessed Jan. 25, 2023).

[4] "Keras API reference." https://keras.io/api/ (accessed Jan. 25, 2023).

[5] Yaya Heryadi and Teguh Wahyono, Machine Learning Konsep dan Implementasi. Yogyakarta: Penerbit Gava Media, 2020.

[6] IdMetafora.com, "Memahami Apa Itu Convolutional Neural Network! Yuk Simak Penjelasannya di Sini, Jasa Pembuatan Website - Metafora Indonesia Tehnology," Apr. 2014.

[7] "MobileNet: Deteksi Objek pada Platform Mobile | by Rizqi Okta Ekoputris | Nodeflux | Medium." https://medium.com/nodeflux/mobilenet-deteksi-objek-pada-platform-mobile-bbbf3806e4b3 (accessed Jan. 25, 2023).

[8] Sandhopi, L. Z. P.C.S.W, and Y. Kristian, "Identifikasi Motif Jepara pada Ukiran dengan Memanfaatkan Convolutional Neural Network," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, vol. 9, no. 4, pp. 403–413, Dec. 2020, doi: 10.22146/JNTETI.V9I4.541.

[9] "Various Optimization Algorithms For Training Neural Network | by Sanket Doshi | Towards Data Science." Based on link https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6 (accessed Jan. 25, 2023).