OPEN ACCESS
VOL. 6, NO. 2, PP.116-125, MEI 2024

ISSN : 2622-8106 (ONLINE)
DOI: 10.20895/INISTA.V6I2.1466

Journal of Informatics, Information System, Software Engineering and Applications (INISTA)

# Marine Fish Classification Based on Image Using Convolutional Neural Network Algorithm and VGG16

Dimas Adira Wibisono *[1], Auliya Burhanuddin [2]

*[1,2] *Informatics Engineering, Telkom Institute of Technology Purwokerto*
*Jl. D.I. Panjaitan No.128, Purwokerto, Jawa Tengah, Indonesia*

[1] 17102078@ittelkom-pwt.ac.id
[2] auliya@ittelkom-pwt.ac.id

Received on 27-05-2024, revised on 14-06-2024, accepted on 21-06-2024

## Abstract

Marine fish are one of the many natural resources that frequently consumed by people. However, several types of marine fish are prohibited from consumption due to being nearly extinct. Additionally, some fish species contain high levels of mercury that can be harmful to human if consumed. Because of the large number of marine fish species, it becomes challenging to identify them without knowledge of fisheries. Moreover, computers have become highly advanced devices that facilitate various human activities. This advancement allows for the creation of systems capable of processing information from images, known as image classification. There are numerous methods that can be employed in designing an image classification system, one of which is transfer learning. This study was aimed to design an image classification system using the transfer learning method with a pre-trained VGG16 model and Convolutional Neural Network algorithm. The research results showed a training data accuracy of 100% and a validation data accuracy of 99.3%, with an overall accuracy of 84% and a loss value of 0.6591.

**Keywords:** Marine fish, Machine learning, Convolutional Neural Network, Transfer learning, VGG16

***Corresponding Author:***
*Dimas Adira Wibisono
Informatics Engineering, Telkom Institute of Technology Purwokerto
Jl. D.I. Panjaitan No.128, Purwokerto, Jawa Tengah, Indonesia
Email: 17102078@ittelkom-pwt.ac.id

## I. INTRODUCTION

INDONESIA is the largest archipelago in the world, with over 17,000 islands and 81,000 km of coastline. According to data from the Ministry of Marine Affairs and Fisheries, Indonesia has a territorial area of 7.81 million km², consisting of 2.55 million km² of exclusive economic zone (EEZ) and 3.25 million km² of ocean, providing vast marine resources, especially for fisheries. One of the industries that can support economic growth through exports is fisheries. In 2019, Indonesia's fishery exports reached 73,681,883,000 rupiahs, an increase of 10.1% from the export value in 2018 [1]. According to data from the Australian Museum, it is estimated that there are 8,500 fish species in Indonesian waters, with 440 species being endemic freshwater fish and over 140 species being endemic marine fish [2]. Marine fish are one of the many natural resources frequently utilized by the community as food. Marine fish have a diversity that can be distinguished by shape, pattern, and color. However, some species of marine fish are prohibited from consumption due to their near extinction. Additionally, some species of fish contain high levels of toxic mercury, which can be harmful to humans if consumed. With the vast number of marine fish species, it becomes difficult to identify them without knowledge in the field of fisheries [3].

Recently, it is undeniable that the rapid development of computer technology and information exchange has a positive impact on daily life. Computers have become sophisticated tools that facilitate various human activities, allowing for the creation of systems capable of processing data from images, known as image processing. Image processing works by manipulating or modifying images and analyzing them to extract information from the objects within those images. There are several methods or algorithms that can be used in the image processing process [4]. In this case, a method is needed to help recognize different types of marine fish.

Machine learning, is the study that explores algorithms and methods used in computer systems by training programs so that they can perform specific tasks without being explicitly programmed [5]. The advantage of using machine learning is that the system can acquire and learn from data with existing commands in the system and work automatically. There are many algorithms and methods for solving data problems in machine learning. One of the methods in machine learning is deep learning. Deep learning is a learning method based on algorithms that simulate how the human brain's neural network works [6], [7]. The computer system is trained using existing data, allowing it to achieve satisfactory results. The systems that employ deep learning methods are trained to classify documents in the form of text, images, or sound. The results of deep learning methods have better accuracy and performance, making them superior to human performance [7]–[10].

There are many methods in deep learning that can be utilized. One of them is transfer learning. Transfer Learning is a technique in deep learning that allows training CNN models quickly and accurately using pre-trained weights, so that the weights do not need to be initialized from scratch [4], [7], [11], [12]. Transfer learning is feasible because the early layers in a CNN architecture typically capture basic features like edges, brightness, and corners, while the deeper layers learn more complex and specific features that are valuable for tasks [13]. These features are applied by using a network trained on millions of image data and training it with a modified network which will later be trained using a small amount of data [4]. VGG16, one of the pre-trained models that utilizes transfer learning techniques, is a model that uses a basic convolutional neural network architecture with a total of 16 convolution layers [10], [14], [15]. The VGG16 model uses a small convolution size of 3x3 [13], [15]–[19]. Therefore, this research was conducted by applying the Convolutional Neural Network algorithm and the pretrained VGG16 model which utilizes transfer learning on the basic architecture of a Convolutional Neural Network in the image classification system for marine fish.

## II. RESEARCH METHOD

In this research, a series of stages had been undertaken by the author. The following is the sequence of the research process carried out in Figure 1

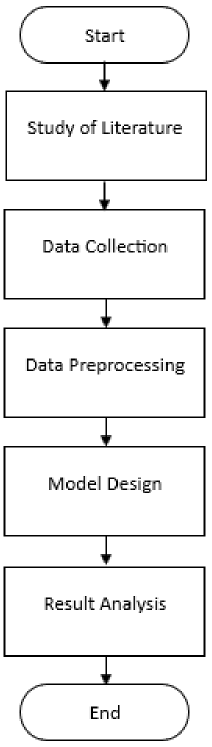


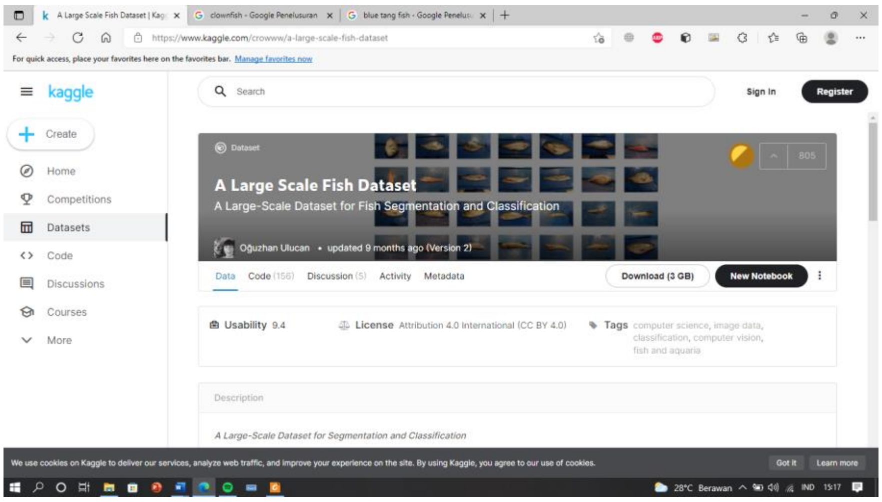
*Fig 1.* Research Flowchart

## A. *Study of Literature*

In this research, a literature analysis was conducted. The author carried out the literature analysis by collecting and reviewing references from various sources such as journals, articles, and websites. From the literature study phase, the author identified a problem for this research. Additionally, by conducting the literature review, the author gained knowledge about related research, including theories and methods used, which facilitated the writing of this research.

## B. *Data Collection*

The dataset used in this research consisted of images of various types of marine fish. The data was obtained from Google and the Kaggle website in .jpg format. The collected image data were divided into four classes, each representing a type of marine fish. Each class contained 1,500 images, with 30 original images and 50 images being augmentations of the original images. The source of the marine fish image dataset was obtained from the Kaggle website. Kaggle is an online community website that discusses and studies data science, machine learning, and other related fields. The website also provides datasets shared by researchers on the platform for free. The dataset used in this research is available at the following link: [https://www.kaggle.com/crowww/a-large-scale-fish-dataset](https://www.kaggle.com/crowww/a-large-scale-fish-dataset). The list of datasets can be seen in Figure 2 for the source, and Figure 3 shows the results.



For quick access, place your favorites here on the favorites bar. Manage favorites now

A Large Scale Fish Dataset

A Large-Scale Dataset for Fish Segmentation and Classification

Oğuzhan Ulucan • updated 9 months ago (Version 2)

Usability 9.4 License Attribution 4.0 International (CC BY 4.0) Tags computer science, image data, classification, computer vision, fish and aquaria

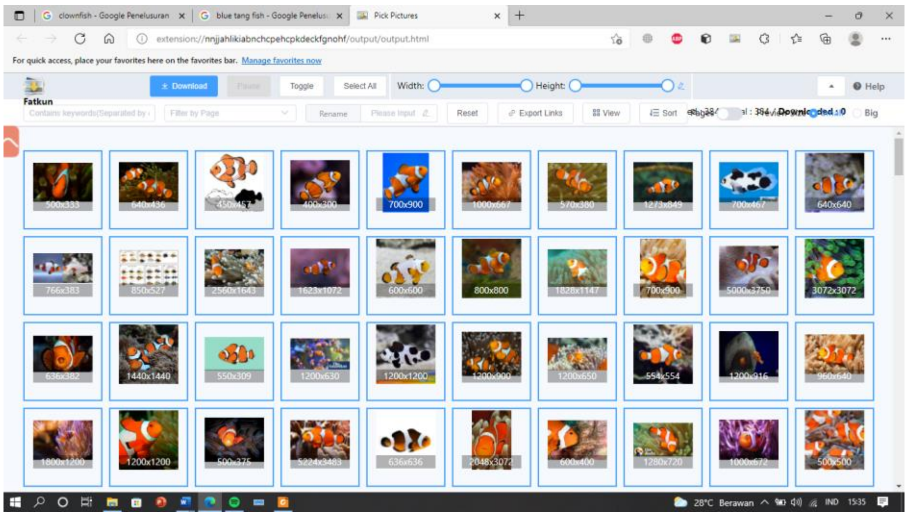
A Large-Scale Dataset for Segmentation and Classification

We use cookies on Kaggle to deliver our services, analyze web traffic, and improve your experience on the site. By using Kaggle, you agree to our use of cookies.

28°C Berawan IND 15:17

*Fig 2.* The dataset source comes from the Kaggle website

The image data collection on Google was done using an additional extension on the web browser called Fatkun Batch Download Image. With the help of this extension, the image data was downloaded simultaneously, thus shortening the data collection time.



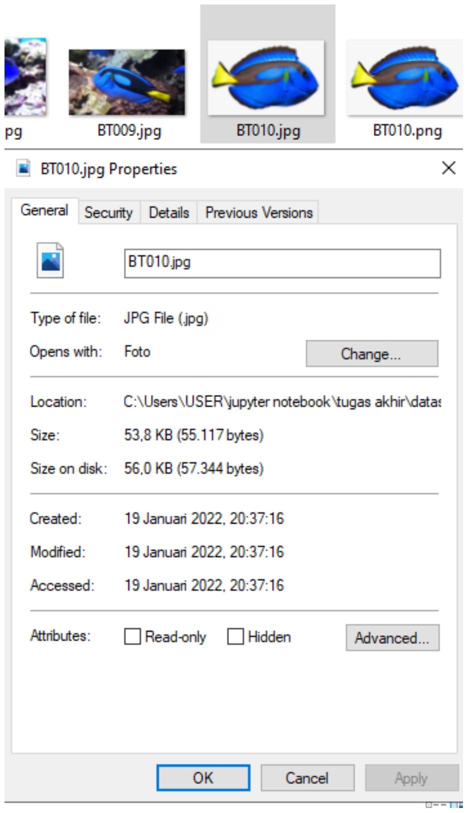
For quick access, place your favorites here on the favorites bar. Manage favorites now

28°C Berawan IND 15:35

Figure 3. The data collection process uses Fatkun Batch Download Image

### C. *Data Preprocessing*

The obtained data then proceed to the preprocessing stage. In this stage, data in formats other than .jpg (.jpeg and .png) were converted to .jpg format to facilitate the resizing and augmentation processes. See the Figure 4.

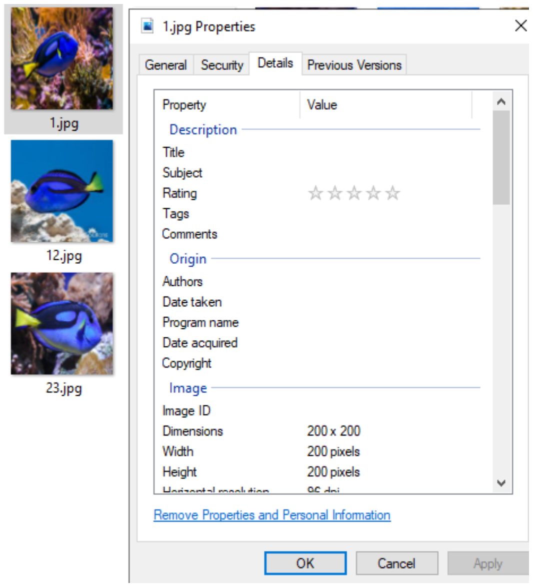


BT010.jpg

Type of file: JPG File (.jpg)
Opens with: Foto
Location: C:\Users\USER\jupyter notebook\tugas akhir\datas
Size: 53,8 KB (55.117 bytes)
Size on disk: 56,0 KB (57.344 bytes)
Created: 19 Januari 2022, 20:37:16
Modified: 19 Januari 2022, 20:37:16
Accessed: 19 Januari 2022, 20:37:16
Attributes: ☐ Read-only ☐ Hidden

*Fig 4.* Sample image after conversion

After the conversion process using the program code, as shown in Figure 4, the sample image format changed to .jpg. This process applies to all sample images with .png or other formats. After the image dataset went through the format conversion process, the image data were resized. The process was applied to all datasets. The purpose of the resize process was to ensure uniformity during the image processing and prevent training loss [20]. See the Figure 5.



| Property | Value |
|---|---|
| Description | |
| Title | |
| Subject | |
| Rating | ☆☆☆☆☆ |
| Tags | |
| Comments | |
| Origin | |
| Authors | |
| Date taken | |
| Program name | |
| Date acquired | |
| Copyright | |
| Image | |
| Image ID | |
| Dimensions | 200 x 200 |
| Width | 200 pixels |
| Height | 200 pixels |
| Horizontal resolution | 96 dpi |

Figure 5. Sample image after resizing

After the sample data underwent the resizing process, the resolution of the image sample changed to 200 x 200 pixels, as shown in Figure 5. This process was also applied to other sample data. The final stage of the preprocessing process was augmentation. The augmentation process aimed to increase the number of data samples used as training and test data so that the model could recognize an image with various patterns and shapes [21]–[23]. In this research, the augmentation process was performed on the dataset, with each class augmented 50 times from the original 30 sample data with rotation and zoom variations, resulting in each class having 1500 image data. The results can be seen in Figure 6.

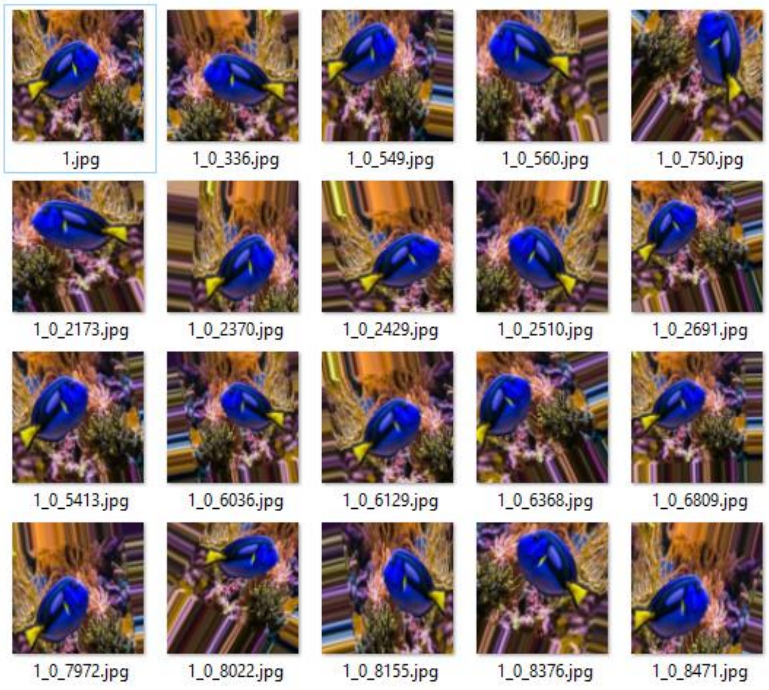


Figure 6. Sample image after augmentation

After entering the augmentation process with a rotation variation of 40 degrees and a zoom variation of 20%, the image samples were modified with a total of 50, including the original images, as shown in Figure 6.

## *D. Model Design*

The initial stage of creating the VGG16 model was begun by inputting the dataset, which had undergone preprocessing previously, into the VGG16 model [15], [18]. After inputting the dataset into the VGG16 model, a fine-tuning process was performed on the model. Fine-tuning involved freezing some convolutional layers and pooling layers responsible for feature extraction in the images to be classified [24]. By freezing some layers, the features learned by the VGG16 model will remain. The next step was to add new classification layers to obtain outputs based on classes that correspond to the features of the previously frozen pre-trained model.

The next step was to train or fine-tune the model using a new dataset. It was done in order to adapt the existing features of the initial model with the new features obtained from training the model. The results of the model design can be seen in the Table I. Table I obtains Layer VGG16 into shape model neural network architecture consisting of a pre-trained VGG16 model followed by additional custom layers tailored for a specific task. The VGG16 model, widely used for image classification, acts as a feature extractor. It outputs a tensor of shape (6, 6, 512) and has 14,714,688 parameters, all non-trainable to leverage the pre-trained features effectively. Following the VGG16 model, a Flatten layer transforms the (6, 6, 512) tensor into a 1D tensor of shape (18432), making it suitable for input into fully connected layers.

The first Dense layer then reduces the dimensionality from 18432 to 256 units, involving 4,718,848 trainable parameters. Subsequently, a second Dense layer further reduces the dimensionality to 4 units, likely corresponding to the number of classes in a classification task, and contains 1,028 trainable parameters. Overall, the model has a total of 19,434,564 parameters (74.14 MB), with 4,719,876 parameters being trainable (18.00 MB) and 14,714,688 parameters being non-trainable (56.13 MB). This architecture efficiently utilizes the pre-trained VGG16 model for feature extraction while focusing the training on the smaller, fully connected layers, optimizing performance and computational resources.

TABLE I. VGG16 MODEL ARCHITECTURE

| Layer (type) | Output Shape | Param # |
|---|---|---|
| VGG16 (Functional) | (None, 6, 6, 512) | 14714688 |
| flatten (Flatten) | (None, 18432) | 0 |
| dense (Dense) | (None, 256) | 4718848 |
| dense_1 (Dense) | (None, 4) | 1028 |
| Total params : 19434564 (74.14 MB) | | |
| Trainable params : 4719876 (18.00 MB) | | |
| Non-trainable params : 14714688 (56.13 MB) | | |

## E. *Result Analysis*

From the results of the designed VGG16 model, the model was tested to determine the level of success in classifying image data. The model was tested by using testing data that had been collected previously, with 150 images per class. The model was tested by training the previously collected training dataset with 50 epochs and 141 steps per epoch. After determining the parameters, the next step was testing the model, with the obtained results of the model testing shown in Table II.

The model's training and validation performance over 50 epochs is summarized as follows. In the initial epoch, the model achieved a training loss of 0.1160 with an accuracy of 0.9691, while the validation loss was 0.0160, and the validation accuracy was 0.9956. Over the subsequent epochs, the training loss decreased significantly, reaching values close to zero, with the model consistently achieving a perfect or near-perfect training accuracy of 1.0000 from the second epoch onwards.

Despite occasional fluctuations, the validation loss generally remained low, indicating good generalization to the validation data. Validation accuracy also remained high, typically above 0.99, though it dipped slightly in a few epochs. Notable performance metrics include the lowest validation loss of 0.0110 in epoch 28, corresponding to a validation accuracy of 0.9956. Conversely, the highest validation loss of 0.1058 occurred in epoch 49, dropping validation accuracy to 0.9711. The model demonstrated robust learning throughout the training process and maintained high accuracy, with minor validation loss and accuracy variances, suggesting stable and effective performance across the epochs.

TABLE II. TRAINING DATASET RESULTS ON THE VGG16 MODEL

| Epoch | Loss | Accuracy | Validation Loss | Validation Accuracy |
|---|---|---|---|---|
| 1 | 0.1160 | 0.9691 | 0.0160 | 0.9956 |
| 2 | 6.8495e-04 | 1.0000 | 0.0192 | 0.9967 |
| 3 | 6.3760e-04 | 1.0000 | 0.0129 | 0.9978 |
| 4 | 0.0039 | 0.9984 | 0.0132 | 0.9933 |
| 5 | 0.0040 | 0.9989 | 0.0350 | 0.9922 |
| 6 | 0.0018 | 0.9993 | 0.0193 | 0.9944 |
| 7 | 0.0497 | 0.9891 | 0.0611 | 0.9778 |
| 8 | 0.0039 | 0.9993 | 0.0260 | 0.9922 |
| 9 | 3.2636e-05 | 1.0000 | 0.0195 | 0.9933 |
| 10 | 1.1685e-04 | 1.0000 | 0.0290 | 0.9933 |
| 11 | 1.9947e-05 | 1.0000 | 0.0169 | 0.9944 |
| 12 | 2.3430e-05 | 1.0000 | 0.0231 | 0.9933 |
| 13 | 6.5292e-04 | 0.9998 | 0.0250 | 0.9900 |
| 14 | 0.0041 | 0.9987 | 0.0668 | 0.9833 |
| 15 | 3.0874e-05 | 1.0000 | 0.0407 | 0.9844 |
| 16 | 1.9956e-05 | 1.0000 | 0.0701 | 0.9833 |
| 17 | 1.3915e-05 | 1.0000 | 0.0543 | 0.9844 |
| 18 | 7.7485e-05 | 1.0000 | 0.0281 | 0.9867 |
| 19 | 3.8612e-05 | 1.0000 | 0.1028 | 0.9744 |
| 20 | 2.9362e-05 | 1.0000 | 0.0217 | 0.9911 |
| 21 | 3.5613e-06 | 1.0000 | 0.0189 | 0.9900 |
| 22 | 1.4642e-05 | 1.0000 | 0.0764 | 0.9778 |
| 23 | 3.0205e-06 | 1.0000 | 0.0490 | 0.9811 |
| 24 | 2.8335e-06 | 1.0000 | 0.0367 | 0.9856 |
| 25 | 1.1759e-06 | 1.0000 | 0.0353 | 0.9867 |
| 26 | 3.0432e-05 | 1.0000 | 0.0354 | 0.9867 |
| 27 | 3.3781e-05 | 1.0000 | 0.0617 | 0.9789 |
| 28 | 3.5940e-05 | 1.0000 | 0.0110 | 0.9956 |

| Epoch | Loss | Accuracy | Validation Loss | Validation Accuracy |
|---|---|---|---|---|
| 29 | 2.0844e-06 | 1.0000 | 0.0172 | 0.9933 |
| 30 | 2.5051e-06 | 1.0000 | 0.0314 | 0.9856 |
| 31 | 1.9922e-06 | 1.0000 | 0.0221 | 0.9911 |
| 32 | 2.9293e-06 | 1.0000 | 0.0394 | 0.9833 |
| 33 | 1.4334e-06 | 1.0000 | 0.0299 | 0.9867 |
| 34 | 4.6927e-07 | 1.0000 | 0.0270 | 0.9867 |
| 35 | 2.0926e-06 | 1.0000 | 0.0422 | 0.9822 |
| 36 | 5.4461e-07 | 1.0000 | 0.0474 | 0.9811 |
| 37 | 7.3073e-07 | 1.0000 | 0.0388 | 0.9833 |
| 38 | 5.0438e-07 | 1.0000 | 0.0352 | 0.9856 |
| 39 | 6.2957e-07 | 1.0000 | 0.0333 | 0.9856 |
| 40 | 9.1151e-07 | 1.0000 | 0.0317 | 0.9867 |
| 41 | 3.8073e-07 | 1.0000 | 0.0297 | 0.9867 |
| 42 | 1.0286e-06 | 1.0000 | 0.0174 | 0.9933 |
| 43 | 4.8674e-07 | 1.0000 | 0.0183 | 0.9933 |
| 44 | 6.9560e-07 | 1.0000 | 0.0157 | 0.9933 |
| 45 | 6.2497e-07 | 1.0000 | 0.0174 | 0.9933 |
| 46 | 4.7292e-07 | 1.0000 | 0.0182 | 0.9933 |
| 47 | 3.8131e-06 | 1.0000 | 0.0164 | 0.9933 |
| 48 | 3.2911e-06 | 1.0000 | 0.0317 | 0.9867 |
| 49 | 9.3574e-07 | 1.0000 | 0.1058 | 0.9711 |
| 50 | 6.3467e-07 | 1.0000 | 0.0627 | 0.9778 |

Table II shows the results of training the dataset with 50 epochs, achieving the highest accuracy value of 100% on the training data and the highest accuracy value of 99.3% on the validation data. See the graph for loss values in Figure 7 and the Figure 8 is the model accuracy.

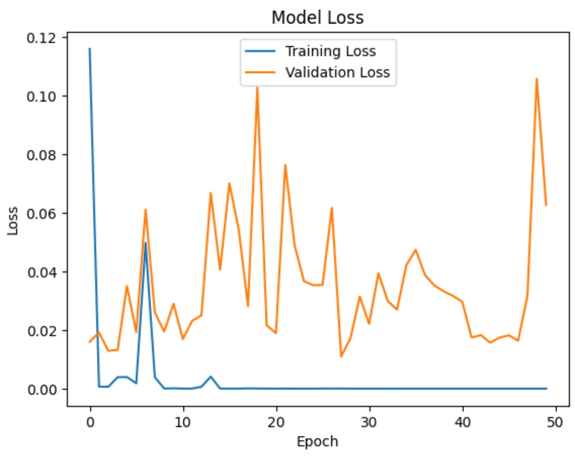


*Fig 7.* Graph of loss values on training and validation data

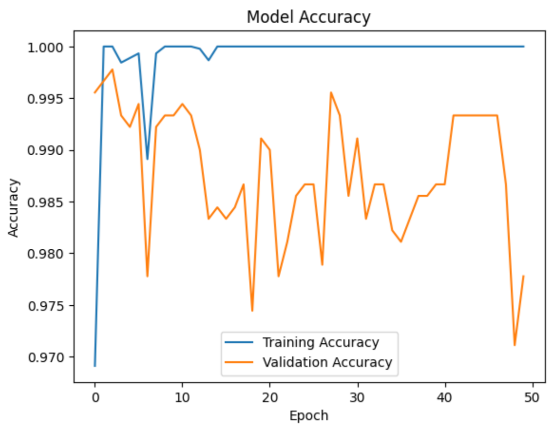
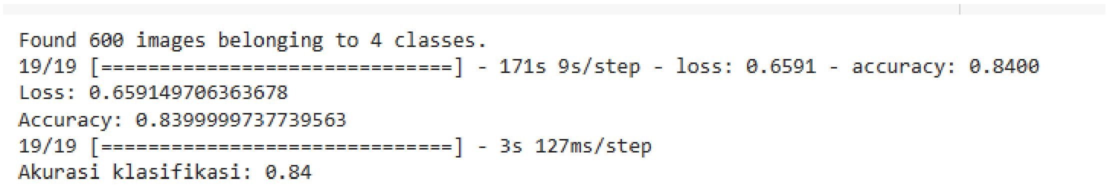


*Fig 8.* Graph of accuracy values on training and validation data

## III. RESULTS AND DISCUSSION

From the training results of the VGG16 model, it can be seen that the testing was conducted on the testing dataset consisting of 600 images. The model testing was performed to evaluate the performance of the VGG16 model in image classification. The testing results can be seen in the Figure 9.



```
Found 600 images belonging to 4 classes.
19/19 [==============================] - 171s 9s/step - loss: 0.6591 - accuracy: 0.8400
Loss: 0.659149706363678
Accuracy: 0.8399999737739563
19/19 [==============================] - 3s 127ms/step
Akurasi klasifikasi: 0.84
```

*Fig 9.* Data testing results on the VGG16 model

From Figure 9, the results of testing data on the VGG16 model were obtained with a loss value of 0.6591 and an accuracy value of 84%.

TABLE III. PRECISION, RECALL, F1-SCORE VALUES IN VGG16 MODEL TESTING

| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| **blue_tang** | 1.00 | 1.00 | 1.00 | 150 |
| **clown_fish** | 1.00 | 1.00 | 1.00 | 150 |
| **sea_bass** | 0.61 | 1.00 | 0.76 | 150 |
| **trout** | 1.00 | 0.36 | 0.53 | 150 |

Table III shows the results of the confusion matrix. The best precision was obtained by the blue_tang, clown_fish, and trout classes which reached a value of 1.0, while the lowest precision obtained by the sea_bass class with a value 0.61. The best recall was obtained by the blue_tang, clown_fish, and sea_bass classes reaching a value of 1.0, while the trout class obtained the lowest recall with a value of 0.36. The best f1-score was obtained by the blue_tang and clown_fish classes, reaching a value of 1.0, while the trout class obtained the lowest f1-score with a value of 0.53.

TABLE IV. ACCURACY VALUE IN VGG16 MODEL TESTING

| | | | | |
|---|---|---|---|---|
| **accuracy** | | | 0.84 | 600 |
| **micro avg** | 0.90 | 0.84 | 0.82 | 600 |
| **weighted avg** | 0.90 | 0.84 | 0.82 | 600 |

Table IV shows the results of the VGG16 accuracy value. The performance metrics for the model are summarized as follows: The overall accuracy of the model is 0.84, based on an evaluation set of 600 samples. Additionally, two averaging methods provide more detailed insights into the model's performance. The micro average metrics show a precision of 0.90, a recall of 0.84, and an F1-score of 0.82, also evaluated on 600 samples. The micro average considers each instance equally, aggregating the contributions of all classes to calculate the average metrics.

Similarly, the weighted average metrics report a precision of 0.90, a recall of 0.84, and an F1-score of 0.82, based on the same 600 samples. The weighted average, however, considers the number of true instances for each label, weighting the metrics accordingly. The highest accuracy value of VGG16 model reached 0.84 with micro averaged and weighted averaged with a value of 0.82.

## IV. CONCLUSION

The neural network architecture, consisting of a pre-trained VGG16 model followed by custom dense layers, demonstrates effective performance in a classification task. The VGG16 model acts as a robust feature extractor with its parameters frozen while the subsequent dense, trainable layers fine-tune the classification. The model achieves a total accuracy of 0.84 on a test set of 600 samples. Further analysis using micro and weighted averages reveals consistent precision (0.90), recall (0.84), and F1-score (0.82), indicating reliable performance across different classes. Overall, the architecture successfully leverages pre-trained features and focuses training on smaller parameters, resulting in an efficient and accurate model. Based on the results and analysis of the research, the following conclusions can be drawn: The highest accuracy achieved by the CNN algorithm using the VGG16 model on the training data is 100% and 99.3% on the validation data. From the evaluation of the VGG16 model, an accuracy value of 84% and a loss value of 0.6591 are obtained.

## REFERENCES

[1] O. Pratama, “Konservasi Perairan Sebagai Upaya menjaga Potensi Kelautan dan Perikanan Indonesia,” 2020. .

[2] Kementerian Kelautan dan Perikanan, “Petunjuk Teknis Pemetaan Sebaran Jenis Agen Hayati Yang Dilindungi, Dilarang, dan Invasif di Indonesia,” *Badan Karantina Ikan, Pengendalian Mutu dan Keamanan Hasil Perikanan*. pp. 1–34, 2015.

[3] M. Ramadhani and D. H. Murti, “Klasifikasi Ikan Menggunakan Oriented Fast and Rotated Brief (Orb) Dan K-Nearest Neighbor (Knn),” *JUTI J. Ilm. Teknol. Inf.*, vol. 16, no. 2, p. 115, 2018, doi: 10.12962/j24068535.v16i2.a711.

[4] H. Darmanto, “Pengenalan Spesies Ikan Berdasarkan Kontur Otolith Menggunakan Convolutional Neural Network,” *Joined J. (Journal Informatics Educ.*, vol. 2, no. 1, p. 41, 2019, doi: 10.31331/joined.v2i1.847.

[5] M. Batta, "Machine Learning Algorithms - A Review ," *Int. J. Sci. Res. (IJ*, vol. 9, no. 1, pp. 381-undefined, 2020, doi: 10.21275/ART20203995.

[6] D. A. Bashar, "Survey on Evolving Deep Learning Neural Network Architectures," *J. Artif. Intell. Capsul. Networks*, vol. 2019, no. 2, pp. 73–82, 2019, doi: 10.36548/jaicn.2019.2.003.

[7] Y. Ke and M. Hagiwara, "CNN-encoded radical-level representation for Japanese processing," *Trans. Japanese Soc. Artif. Intell.*, vol. 33, no. 4, 2018, doi: 10.1527/tjsai.D-I23.

[8] M. Rajnoha, R. Burget, and M. K. Dutta, "Handwriting Comenia Script Recognition with Convolutional Neural Network," pp. 775–779, 2017.

[9] F. F. Maulana and N. Rochmawati, “Klasifikasi Citra Buah Menggunakan Convolutional Neural Network,” *J. Informatics Comput. Sci.*, vol. 01, pp. 104–108, 2019.

[10] D. F. H. Permadi and M. Z. Abdullah, “Implementasi pengenalan citra wajah binatang dengan algoritma convolutional neural network (cnn),” *JUTI J. Ilm. Teknol. Inf.* -, vol. 21, no. 1, pp. 30–41, 2023.

[11] I. Kandel and M. Castelli, "Transfer Learning with Convolutional Neural Networks for Diabetic Retinopathy Image Classification. A Review Ibrahem," *Appl. Sci.*, vol. 10, no. 6, pp. 1–24, 2020.

[12] S. S. Yadav and S. M. Jadhav, "Deep convolutional neural network based medical image classification for disease diagnosis," *J. Big Data*, 2019, doi: 10.1186/s40537-019-0276-2.

[13] C. Narvekar and M. Rao, "Flower classification using CNN and transfer learning in CNN-Agriculture Perspective," *Proc. 3rd Int. Conf. Intell. Sustain. Syst. ICISS 2020*, pp. 660–664, 2020, doi: 10.1109/ICISS49785.2020.9316030.

[14] P. M. Samuel and T. Veeramalai, "Review on retinal blood vessel segmentation - An algorithmic perspective," *Int. J. Biomed. Eng. Technol.*, vol. 34, no. 1, pp. 75–105, 2020, doi: 10.1504/IJBET.2020.110362.

[15] R. Agustina, R. Magdalena, and N. O. R. K. Caecar, “Klasifikasi Kanker Kulit menggunakan Metode Convolutional Neural Network dengan Arsitektur VGG-16,” *Elkomika*, vol. 10, no. 2, pp. 446–457, 2022, [Online]. Available: https://ejurnal.itenas.ac.id/index.php/elkomika/article/view/5674/2879.

[16] Darmatasia, “Analisis Perbandingan Performa Model Deep Learning Untuk Mendeteksi Penggunaan Masker,” *J. IT Media Inf. IT STMIK Handayani*, vol. 11, no. 3, pp. 65–71, 2020.

[17] P. M. Samuel and T. Veeramalai, "VSSC Net: Vessel Specific Skip chain Convolutional Network for blood vessel segmentation," *Comput. Methods Programs Biomed.*, vol. 198, p. 105769, 2021, doi: 10.1016/j.cmpb.2020.105769.

[18] M. A. Pangestu and H. Bunyamin, “Analisis Performa dan Pengembangan Sistem Deteksi Ras Anjing pada Gambar dengan Menggunakan Pre-Trained CNN Model,” *J. Tek. Inform. dan Sist. Inf.*, vol. 4, pp. 337–344, 2018.

[19] Y. Wang, P. Yu, and C. Li, "Offline Handwritten New Tai Lue Characters Recognition Using CNN-SVM," *Proc. 2019 IEEE 2nd Int. Conf. Electron. Inf. Commun. Technol. ICEICT 2019*, pp. 636–639, 2019, doi: 10.1109/ICEICT.2019.8846292.

[20] N. E. W. Nugroho and A. Harjoko, "Transliteration of Hiragana and Katakana Handwritten Characters Using CNN-SVM," *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 15, no. 3, p. 221, 2021, doi: 10.22146/ijccs.66062.

[21] J. Gan, W. Wang, and K. Lu, "A new perspective : Recognizing online handwritten Chinese characters via 1-dimensional CNN," vol. 478, no. Information Sciences 478, pp. 375–390, 2019, doi: 10.1016/j.ins.2018.11.035.

[22] J. Zhang and T. Matsumoto, "Corpus augmentation for neural machine translation with Chinese-

Japanese parallel corpora," *Appl. Sci.*, vol. 9, no. 10, 2019, doi: 10.3390/app9102036.

[23] H. Miyao and M. Maruyama, "Virtual example synthesis based on PCA for off-line handwritten character recognition," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3872 LNCS. pp. 96–105, 2006, doi: 10.1007/11669487_9.

[24] J. Wang *et al.*, "GIT: A Generative Image-to-text Transformer for Vision and Language," *ArXiv*, vol. 2, pp. 1–49, 2022, [Online]. Available: http://arxiv.org/abs/2205.14100.