Proceedings of Engineering and Technology Innovation, vol. 19, 2021, pp. 16-27

Fish Species Detection Application (FiSDA) in Leyte Gulf Using Convolutional Neural Network

Gil Gabornes Dialogo[1,*], Larmie Santos Feliscuzo[2], Elmer Asilo Maravillas[2]

[1]College of Computer Studies, Eastern Samar State University, Samar, Philippines

[2]College of Computer Studies, Cebu Institute of Technology-University, Cebu, Philippines

Received 15 June 2021; received in revised form 26 July 2021; accepted 27 July 2021

DOI: https://doi.org/10.46604/peti.2021.7892

Abstract

This study presents an application that employs a machine-learning algorithm to identify fish species found in Leyte Gulf. It aims to help students and marine scientists with their identification and data collection. The application supports 467 fish species in which 6,918 fish images are used for training, validating, and testing the generated model. The model is trained for a total of 4,000 epochs. Using convolutional neural network (CNN) algorithm, the best model during training is observed at epoch 3,661 with an accuracy rate of 96.49% and a loss value of 0.1359. It obtains 82.81% with a loss value of 1.868 during validation and 80.58% precision during testing. The result shows that the model performs well in predicting Malatindok and Sapsap species, after obtaining the highest precision of 100%. However, Hangit is sometimes misclassified by the model after attaining 55% accuracy rate from the testing results because of its feature similarity to other species.

Keywords: fish species, fish detection, mobile application, convolutional neural network

1. Introduction

The Leyte Gulf is among the major fishing grounds in the Philippines with a shelf area of about 2,724 square kilometers, covering the islands of Samar and Leyte. It is home to more than 467 different fish species [1]. Fishes are cold-blooded animals, typically with backbone, gills, fins, and lungs. They range from about 15,000 to 17,000 species [2]. Fishes have many kinds and have varied colors, shapes, and sizes [3]. Fish species recognition is a multi-class classification problem and is a compelling field of study that employs machine learning and computer vision [4]. Some researchers noticed that detection is a crucial part of a fish classification and counting system [5]. Moreover, manual species identification is not only time-consuming but also prone to misclassification especially in the Leyte Gulf, in which over 467 fish species exist [6].

There is relatively poor documentation for most groups of fishes, and the information on the inventory of species present within the gulf is especially scarce. Thus, employing computer vision and machine learning in fish species identification with developed technologies would transform marine science [7]. Various promising techniques for the identification of fishes emerged particularly in genetics, interactive computer software, image recognition, hydro-acoustics, and morphometry [8]. Recently, there is a paradigm shift of set-based classification for object recognition [9]. The convolutional operation is frequently used in computer vision, especially for noise reduction and edge detection [10].

An automated system for the identification and classification of fish species was created using a reduced version of AlexNet based on deep convolutional neural networks (CNNs). The results show that the modified AlexNet model has achieved the testing accuracy of 90.48% while the original AlexNet model achieved 86.65% over the untrained dataset [11]. In addition, a CNN system for aquarium family fish species identification achieved 85.59% testing accuracy [12].

* Corresponding author. E-mail address: dialogogil1992@gmail.com

Tel.: +63-965-0739-502

Optical fish detection network was applied to a system that is capable of parameterizing fish schools in underwater images. This was based on deep learning object detection architectures, and carried out the task of fish detection, localization, and species classification using visual data obtained by underwater cameras. Based on the experiments, it successfully detects 66.7% of the fish included, and further classifies 89.7% correctly [13]. Rekha et al. [14] used CNN with different architectures to extract and analyze the features in the detection and classification of various fish species to help and protect endangered species. The system exhibits an accuracy of 90% and 92% on the detection and classification respectively.

On the other hand, Fabic et al. [15] used blob counting and shape analysis for fish detection, counting, and species classification from underwater video sequences to identify the two most common fish species found in the Tubbathaha reef in the Sulu Sea, Philippines. Moreover, a two-step deep learning approach was used for the detection and classification of temperate fishes without pre-filtering. It employed the You Only Look Once (YOLO) object detection technique. In the second step, it adopted CNN with squeeze-and-excitation (SE) architecture for classifying each fish in the image without pre-filtering. The system achieved an accuracy of 99.27% using the pre-training model. Using the post-training model, it obtained 83.68% and 87.74% with and without image augmentation [16].

Given the above challenges in fisheries, the agreed unifying strategic objective for the classification and identification of fish is to develop information and communications technology (ICT) software or systems such as mobile applications, which have quickly become useful tools and are widely used today for their diversity and portability [17].

Thus, this study intends to develop a handy mobile application to identify fish species present in Leyte Gulf. The mobile users can capture an unknown fish image to the developed mobile application, in which the proposed model embedded in the application will then attempt to recognize the fish species. The application displays the recognition results on the application's graphical user interface. Aside from helping the non-professional fish enthusiasts, the produced information is essential in the decision-making processes of fisheries, marine conservation managers, and scientists, as well as in the documentation of species present within the gulf.

2. Materials and Methods

2.1. Data preparation

The available dataset is collected from the Bureau of Fisheries and Aquatic Resources (BFAR) Regional Office No. VIII. The dataset includes the list of species in Leyte Gulf and their local names. The corresponding images of the various species come from the BFAR publications and Fishbase [18]. There are 6,918 images used covering 467 fish species present in the areas along the Leyte Gulf as shown in Fig. 1. These images are clustered into 35 classes according to their local names. A ratio of 80-10-10 of the images is allocated, i.e., 5, 548 images for training, 685 images for validation and testing respectively as presented in Table 1.

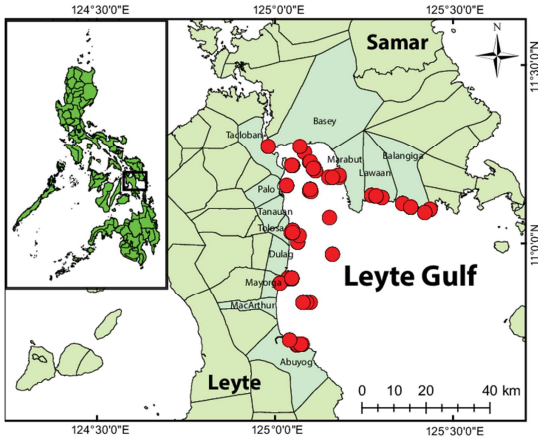


Fig. 1 Map of Leyte Gulf

Table 1 Image allocation per class

Class	Number of images			Total
	Training	Validation	Testing	
Abo	109	13	13	135
Alho	163	20	20	203
Alibangbang	100	12	12	124
Baga-baga	122	15	15	152
Bisugo	87	11	11	109
Bon-ak	184	24	24	232
Danggit	316	40	40	396
Dumpilas	199	8	8	215
Gangis	102	13	13	128
Ganting	120	15	15	150
Hamorok	104	13	13	130
Hangit	120	15	15	150
Katambak	121	15	15	151
Kirawan	140	17	17	174
Labungan	179	23	23	225
Lapu-lapu	320	40	40	400
Lubayan	122	16	16	154
Malatindok	102	13	13	128
Mamsa	115	15	15	145
Mangagat	119	15	15	149
Mol-mol	305	38	38	381
Pakol	162	21	21	204
Palad	181	23	23	227
Panit	100	13	13	126
Pating	187	24	24	235
Sagisi-on	88	11	11	110
Sapsap	122	16	16	154
Siri	182	23	23	228
Sulid	118	15	15	148
Surahan	239	30	30	299
Talakitok	102	13	13	128
Tamban	189	24	24	237
Tarukitok	123	16	16	155
Ti-aw	184	24	24	232
Tingag	322	41	41	404
Grand total	5,548	685	685	6,918

2.2. *Conceptual framework*

The application starts with the user capturing an image of fish using a mobile camera. The captured image will then be processed by the generated fish classification model. Finally, the application displays its prediction result with the details of the identified fish species as shown in Fig. 2.

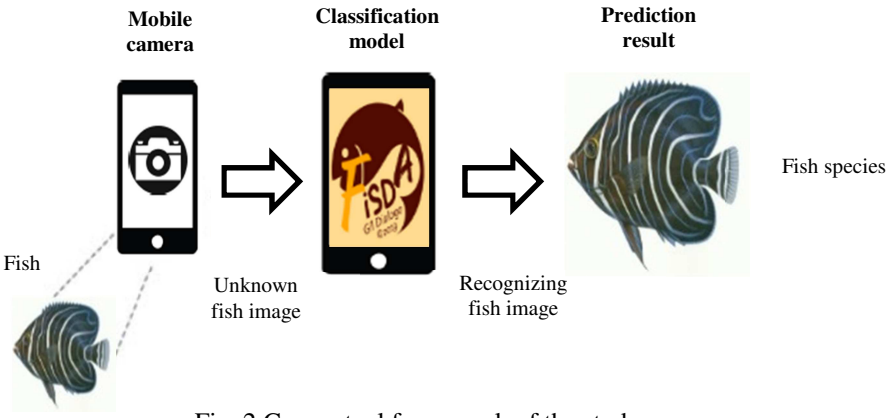


Fig. 2 Conceptual framework of the study

2.3. Building the CNN model

The proposed model is built by using Python 3.6. The architecture of the deep network for the fish species identification is introduced in details in Fig. 3. It depicts the architecture for this study, which takes a fish image, processes the image, and then classifies the image under a certain type of fish species. The input image sequentially goes through a series of convolution-pooling layers for extracting low-level to high-level features and fully connected layers for mapping the extracted features into the final output.

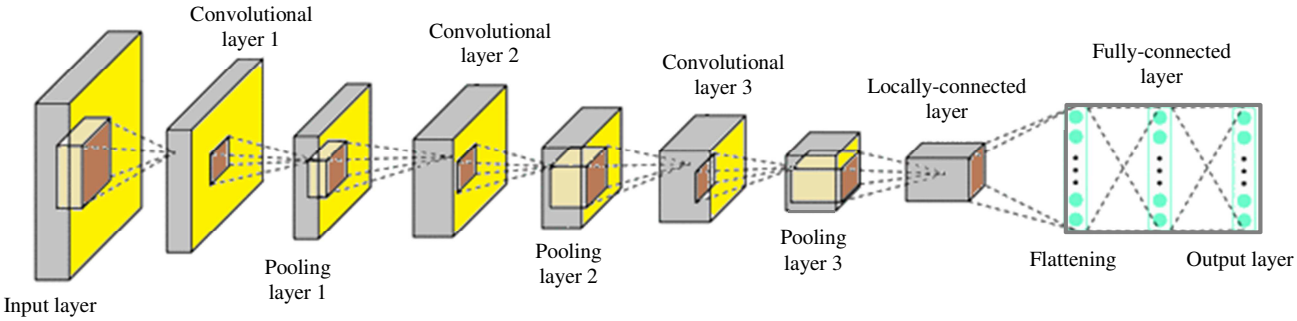


Fig. 3 CNN architecture

The convolutional layer contains learnable filters or kernels which are applied across the width and height of the input tensor. It then performs element-wise products between the entries of the filters and the input at any image positions and summed to obtain the feature maps. The output feature maps of convolution are passed through a rectified linear activation function, which returns the input directly if it is positive or zero as it receives any negative input. This function allows the model to learn faster and to perform better.

The pooling layer will then perform a downsampling operation, which progressively reduces the spatial size of the representation to decrease the number of subsequent learnable parameters as well as the computation in the network. In this study, max pooling with a filter of 2 × 2 and with a stride of 2 is applied, which outputs the maximum value in each patch extracted from the input feature maps.

The operations will be repeated until all the convolution-pooling layers have been finished, in which the final feature maps will be transformed to a one-dimensional array of numbers and connected to the fully connected or dense layers. The flattened output is being fed to a feed-forward neural network and applied backpropagation to every iteration of training. Over an iterated epoch, the model can distinguish between domination and certain low-level features in the images and classify them through the softmax activation function, wherein each value ranges between 0 and 1, and all values sum up to 1.

2.4. Model performance evaluation

The next phase is to determine how effective the model is, based on some basic performance metrics using the test dataset. The metrics include accuracy (Eq. (1)), precision (Eq. (2)), recall (Eq. (3)), and specificity (Eq. (4)).

$$A = \frac{tp + tn}{tp + tn + fp + fn} \tag{1}$$

$$P = \frac{tp}{tp + fp} \tag{2}$$

$$R = \frac{tp}{tp + fn} \tag{3}$$

$$S = \frac{tn}{tn + fp} \tag{4}$$

where *tp* represents that when the actual class is true and the predicted is also true; *tn* represents that when the actual class is false and the predicted is also false; *fp* represents that when the actual class is false and the predicted is true; *fn* represents that when the actual class is true and the predicted is false.

3. Results and Discussion

3.1. CNN model for fish recognizer

Fig. 4 illustrates that the 64 × 64 input image becomes 62 × 62 after the 3 × 3 filter in the first convolutional layer. It gets reduced in half after each pooling layer, from 31 × 31 on the first, then to 14 × 14 on the second, and 6 × 6 on the last pooling layer. This will then be flattened, resulting in 4,608 (6 × 6 × 128), the shape of the data once it comes out of the convolutions. The first dense layer with 256 neurons has a total of 1,179,904 (256 × (4,608 + 1)) parameters, while the second dense layer with 35 neurons as well has 8,995 parameters (35 × (256 + 1)).

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 62, 62, 32)	896
activation_1 (Activation)	(None, 62, 62, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 31, 31, 32)	0
conv2d_2 (Conv2D)	(None, 29, 29, 64)	18496
activation_2 (Activation)	(None, 29, 29, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 14, 14, 64)	0
conv2d_3 (Conv2D)	(None, 12, 12, 128)	73856
activation_3 (Activation)	(None, 12, 12, 128)	0
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 128)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_1 (Dense)	(None, 256)	1179904
activation_4 (Activation)	(None, 256)	0
dense_2 (Dense)	(None, 35)	8995
activation_5 (Activation)	(None, 35)	0

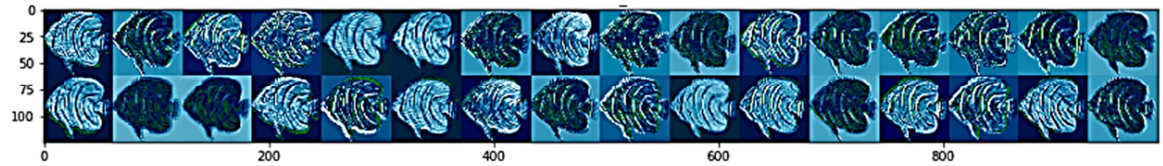
Total params: 1,282,147
Trainable params: 1,282,147
Non-trainable params: 0

Fig. 4 Summary of model layers

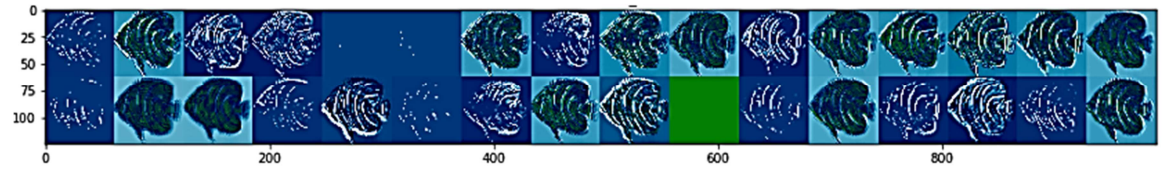
A sequential neural network with input shape (64, 64, and 3) is configured wherein 64 × 64 represents the image dimension, while 3 indicates that the input image is colored (RGB). The network is composed of a linear stack of 3 sets of convolutional (Conv2D) - pooling (MaxPooling2D) layers before the dense or fully connected layers at the bottom. The Conv2D layers have 32, 64, and 128 output channels respectively and a kernel size of 3 × 3. The activation function for each Conv2D layer is the Rectified Linear Unit (ReLU), followed by a MaxPooling2D layer, which reduces the number of parameters in the model by sliding a 2 × 2 pooling filter across the previous layer and taking the max values in the filter. Between the convolutional layers and the dense layers, there is a flatten layer that connects them. The first two dense layers both have 256 nodes, each activated by a ReLU function. The last dense layer has 35 nodes activated by the softmax activation function, which allows the output to be interpreted as probabilities. Thus, the model will take the class option, which obtains the highest probability.

Figs. 5-7 demonstrate the visualization of every channel for each intermediate activation phase. It shows how CNN finds the patterns in the images and how it carries the information from one layer to another layer. It can be noticed that the activations in the above layers retain almost all of the information present in the initial image. However, when the layers get more in-depth, the

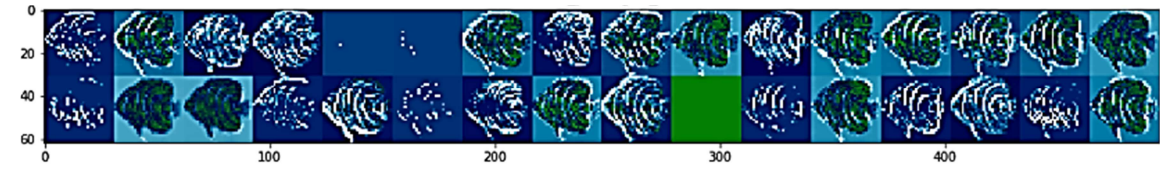
activations become increasingly abstract and less visually interpretable. The network begins to encode higher-level presentations which carry gradually less information about the visual contents of the image and more information related to the class of the image. There are instances that the filters are left blank, which indicates that they are not activated at all.



(a) Feature map from the first set of Conv2D layer

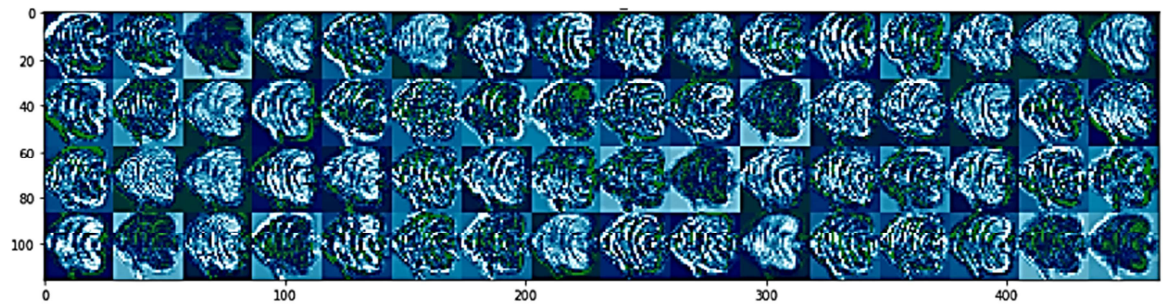


(b) Feature map from the first set of ReLU

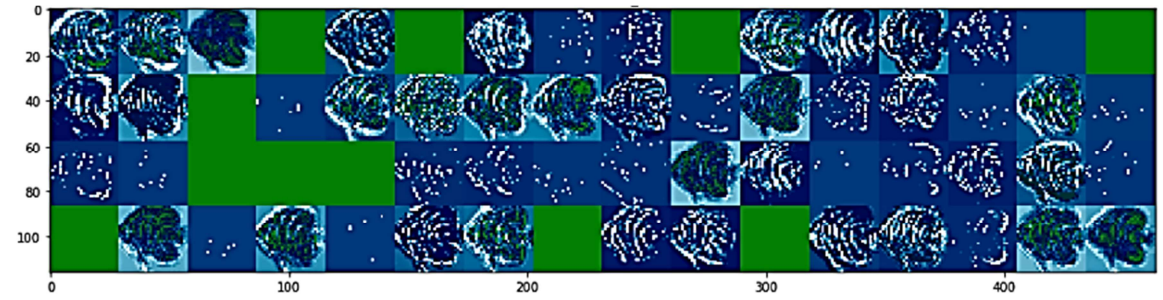


(c) Feature map from the first set of MaxPooling2D layer

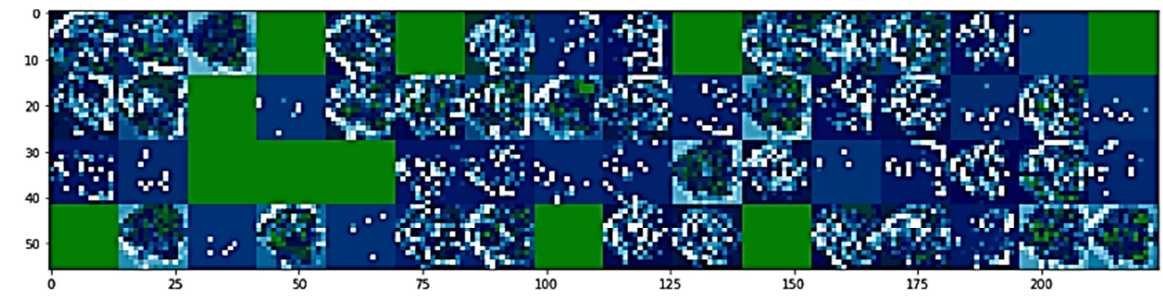
Fig. 5 Visualization of feature maps from the first set of Conv + ReLU + Pool



(a) Feature map from the second set of Conv2D layer

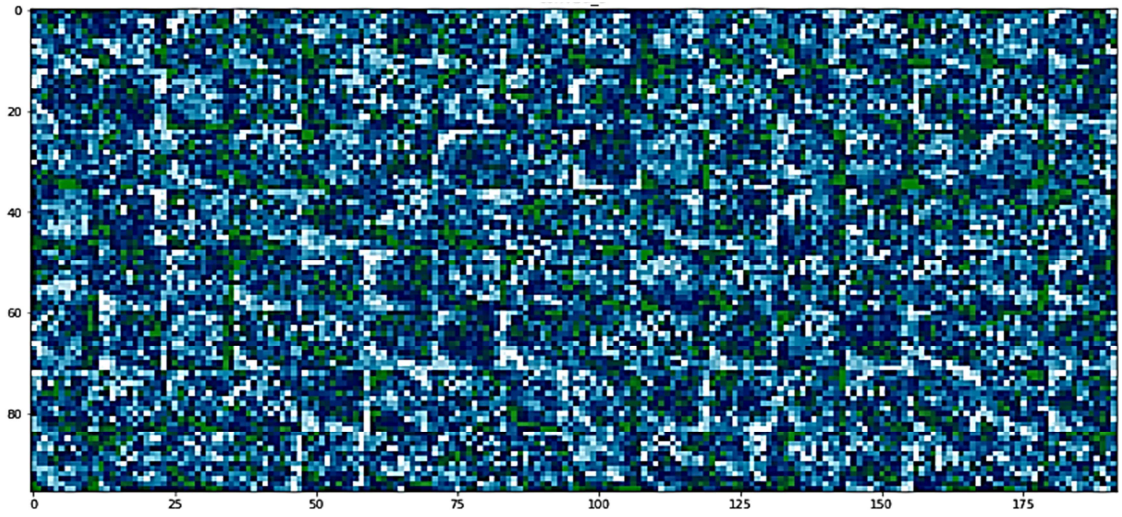


(b) Feature map from the second set of ReLU

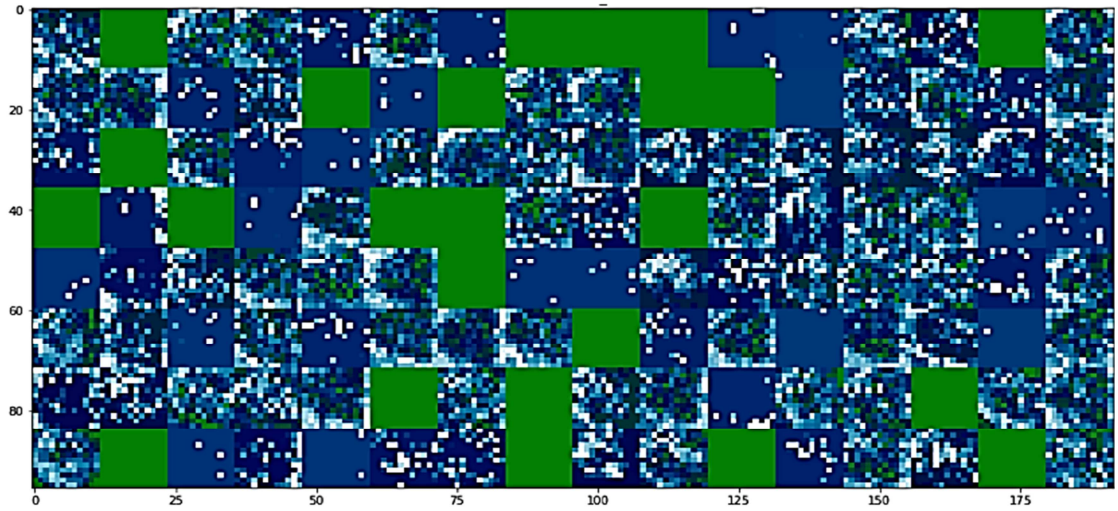


(c) Feature map from the second set of MaxPooling2D layer

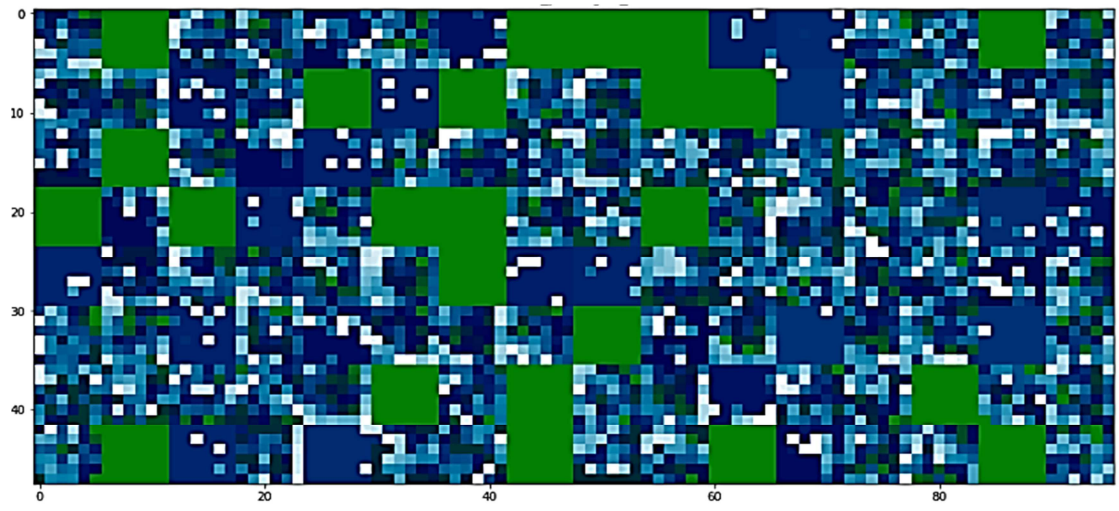
Fig. 6 Visualization of feature maps from the second set of Conv + ReLU + Pool



(a) Feature map from the third set of Conv2D layer



(b) Feature map from the third set of ReLU



(c) Feature map from the third set of MaxPooling2D layer

Fig. 7 Visualization of feature maps from the third set of Conv + ReLU + Pool

Fig. 8 shows the training and validation accuracy as a function of the epoch. The accuracy metric is calculated to measure the algorithm's performance in an interpretable way. It is the measure of how accurate the model's prediction is compared to the actual data.

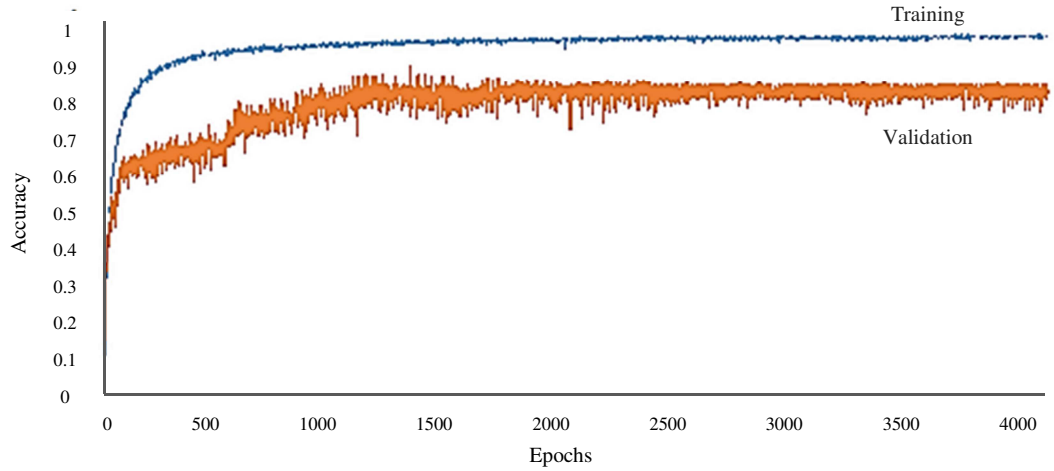


Fig. 8 Training accuracy plot of the CNN model

The loss is also calculated to serve as a reference in determining how well the model is during the training and validation phase. Unlike accuracy, a loss is not a percentage. It is the sum of errors made for each sample in training or validation sets, and implies how poorly or well a model behaves after each iteration of optimization. The loss function of the model is illustrated in Fig. 9, which shows that the loss for the training phase is nearly zero indicating that the model's prediction on the trained datasets is almost perfect. On the other hand, as to the validation phase, there is an occurrence of overfitting showing that the model does not affirmatively well in predicting new datasets.

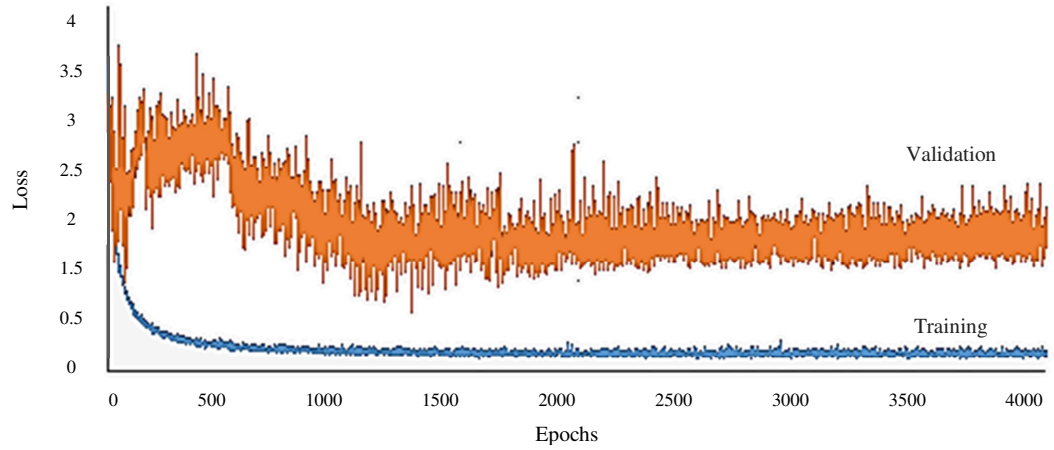


Fig. 9 Training loss plot of the CNN model

The model is trained for a total of 4,000 epochs. The best model during training is observed at epoch 3,661 obtaining an accuracy rate of 96.49% with a loss value of 0.1359. However, the performance of the model for the validation set is only 82.81% with a loss value of 1.868. The result implies that the model performs less well on the unseen dataset.

3.2. Model performance

The confusion matrix shown in Table 2 describes the performance of the model on the testing dataset. It displays the number of correct predictions (diagonal) made by the model, as well as the number of incorrect predictions (off-diagonal). Based on the generated confusion matrix, the model garners an accuracy of 80.58% during testing. As observed also, Malatindok and Sapsap are the best species predicted by the model since they obtain 100% accuracy. It is because of their distinct morphological features susch as shape and color. On the other hand, Hangit gains the least accuracy after it results in a 55% precision rate. It means that Hangit is the most misclassified species by the model because of its feature similarity to other species.

Table 2 Confusion matrix

-	Abo	Alho	Alibangbang	Baga-baga	Bisugo	Bon-ak	Danggit	Dumpilas	Gangis	Ganting	Hamorok	Hangit	Katambak	Kirawan	Labungan	Lapu-lapu	Lubayan	Malatindok	Mamsa	Mangagat	Mol-mol	Pakol	Palad	Panit	Pating	Sagisi-on	Sapsap	Siri	Sulid	Surahan	Talakitok	Tamban	Tarukitok	Ti-aw	Tingag	Actual total
Abo	12															1																				13
Alho	1	16									1											1											1			20
Alibangbang			10													1																		1	12
Baga-baga				13											1																			1	15
Bisugo					8	1																				1								1		11
Bon-ak						20	1								1	1												1								24
Danggit							36									1						1								2						40
Dumpilas								6														1						1								8
Gangis									11		1																			1						13
Ganting				2						10					1	2																				15
Hamorok										1	10																					1			1	13
Hangit											1	11													1										2	15
Katambak						1							12	1														1								15
Kirawan				1			1	1			1			11		1				1																17
Labungan							1	1				1			18					1								1								23
Lapu-lapu							2					2				31							1												4	40
Lubayan			1						1					1			12			1																16
Malatindok												1						9	1													1		1		13
Mamsa																			10			1						1					3			15
Mangagat															1					12		1												1		15
Mol-mol							1	1				1					1				31					1		1		1						38
Pakol							1														1	18								1						21
Palad							1																22													23
Panit												1							1					11												13
Pating	1	2																							21											24
Sagisi-on					1		1																			8								1		11
Sapsap											2																13							1		16
Siri							2		1			1	1															17						1		23
Sulid																1				1								1	12							15
Surahan							2									2					1	1								23	1					30
Talakitok											1																	1			11					13
Tamban												1												1								21		1		24
Tarukitok																															1		15			16
Ti-aw								1					1																1			1		19	1	24
Tingag				1			3					1				3														1					32	41
Predicted total	14	18	11	17	9	22	52	10	13	11	17	20	14	13	22	44	13	9	12	16	33	24	23	12	22	10	13	25	13	29	13	24	21	24	42	552

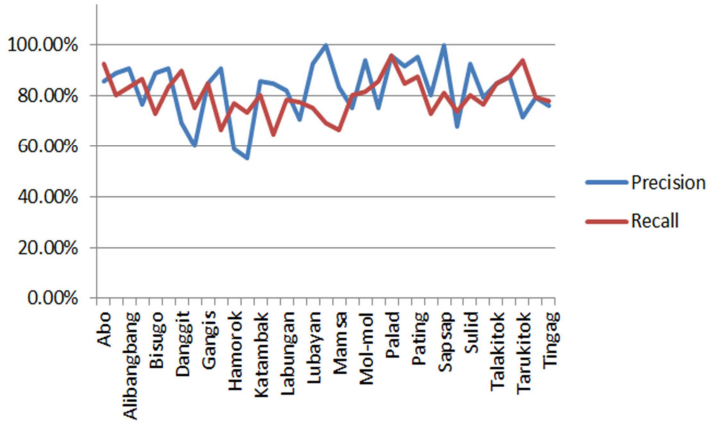


Fig. 10 Precision versus recall

Table 3 presents the results of calculating some of the performance measures that can be derived from the confusion matrix. Palad, as observed also in Fig. 10, earns the highest recall of 95.65%, and Hangit suffers from low precision of 55%, which denotes that Hangit positive values are unpredictable. However, only a few of these positive predictions are correct. On

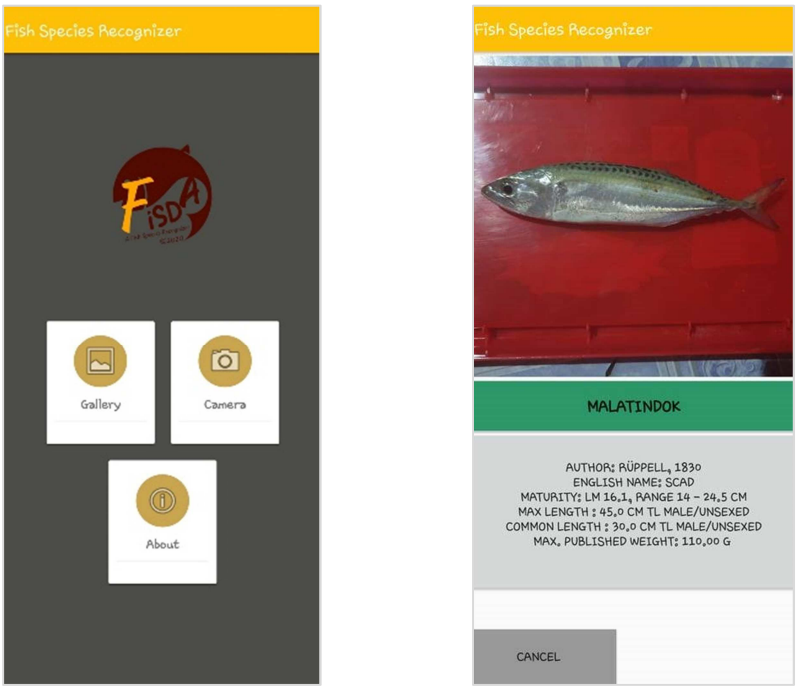
the other hand, Malatindok and Sapsap have the highest precision of 100% while Kirawan garnered the lowest recall of 64.71%. This is a manifestation that the model is returning very few results, yet most of its predicted labels are correct when compared to the training labels. Moreover, Malatindok and Sapsap classes obtain the highest specificity percentage which signifies that the model is 100% accurate in recognizing the fish images.

Table 3 Performance evaluation result

Class	Precision	Recall	Specificity
Abo	85.71%	92.31%	99.70%
Alho	88.89%	80.00%	99.70%
Alibangbang	90.91%	83.33%	99.85%
Baga-baga	76.47%	86.67%	99.40%
Bisugo	88.89%	72.73%	99.85%
Bon-ak	90.91%	83.33%	99.70%
Danggit	69.23%	90.00%	97.52%
Dumpilas	60.00%	75.00%	99.41%
Gangis	84.62%	84.62%	99.70%
Ganting	90.91%	66.67%	99.85%
Hamorok	58.82%	76.92%	98.96%
Hangit	55.00%	73.33%	98.66%
Katambak	85.71%	80.00%	99.70%
Kirawan	84.62%	64.71%	99.70%
Labungan	81.82%	78.26%	99.40%
Lapu-lapu	70.45%	77.50%	97.98%
Lubayan	92.31%	75.00%	99.85%
Malatindok	100.00%	69.23%	100.00%
Mamsa	83.33%	66.67%	99.70%
Mangagat	75.00%	80.00%	99.40%
Mol-mol	93.94%	81.58%	99.69%
Pakol	75.00%	85.71%	99.09%
Palad	95.65%	95.65%	99.85%
Panit	91.67%	84.62%	99.85%
Pating	95.45%	87.50%	99.85%
Sagisi-on	80.00%	72.73%	99.70%
Sapsap	100.00%	81.25%	100.00%
Siri	68.00%	73.91%	98.79%
Sulid	92.31%	80.00%	99.85%
Surahan	79.31%	76.67%	99.08%
Talakitok	84.62%	84.62%	99.70%
Tamban	87.50%	87.50%	99.55%
Tarukitok	71.43%	93.75%	99.10%
Ti-aw	79.17%	79.17%	99.24%
Tingag	76.19%	78.05%	98.45%

3.3. Graphical user interface

The graphical user interface for the mobile-based application is created using Android Studio 4.1 in which the model is embedded in the application. The generated model, which is a Keras file (.h5), is converted into a Tensorflow file (.tflite) using TensorFlow Lite which is a set of tools to help developers run TensorFlow models on mobile. This file is then deployed into the mobile-based application. The graphical user interface of the application is shown in Figs. 11 (a) and (b) which display the final prediction of the classification model on the uploaded or selected image as well as the necessary information of the predicted fish.



(a) Main interface of the application (b) Recognition result interface

Fig. 11 Graphical user interface of the application

4. Conclusions and Recommendations

This study was able to come up with a fish species recognizer model which was successfully embedded in a mobile-based application. By using CNN, the generated model obtained a training accuracy rate of 96.49% somehow higher than the study of Pudaruth et al. [19], which only gained 96% using k-nearest neighbors (kNN) classifier. During the validation process, the model achieved 82.81% accuracy with a loss value of 1.868, which indicates that the model has good reliability when it comes to predicting the fish images. Moreover, during the testing process, the model gained an accuracy rate of 80.58%. The result revealed that the model performs well in predicting Malatindok and Sapsap species, which gained the highest precision of 100%. It is because of their distinct morphological features such as shape and color. On the other hand, Hangit was sometimes misclassified by the model after obtaining a 55% accuracy rate from the testing results because of insufficient dataset for this specific species as well as of its feature similarity to other species. In the future, there is a need to further improve the performance of the model by using more datasets of fish images to have a better prediction result.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] J. Davies, P. M. Magsalay, R. Rigor, A. Mapalo, and H. Gonzales, “Directory of Philippine Wetlands: A Preliminary Compilation of Information on Wetlands of the Philippines,” Philippines: Asian Wetland Bureau Philippines Foundation, 1990.

[2] M. C. Francisco, N. A. Dayap, L. A. Tumabiene, R. A. Francisco, M. J. Candole, J. H. De Veyra, et al., “Status of Leyte Gulf Fisheries CYs 2001-2011,” The Philippine Journal of Fisheries, vol. 25, no. 1, pp. 136-155, 2018.

[3] K. F. Lagler, J. E. Bardach, and R. R. Miller, “Ichthyology: The Study of Fishes,” New York: John Wiley & Sons, 1962.

[4] D. Rathi, S. Jain, and S. Indu, “Underwater Fish Species Classification Using Convolutional Neural Network and Deep Learning,” 9th International Conference on Advances in Pattern Recognition, December 2017, pp. 1-6.

[5] J. Jäger, E. Rodner, J. Denzler, V. Wolff, and K. Fricke-Neuderth, “SeaCLEF 2016: Object Proposal Classification for Fish Detection in Underwater Videos,” 7th Conference and Labs of the Evaluation Forum, September 2016, pp. 481-489.

[6] X. Lan, J. Bai, M. Li, and J. Li, "Fish Image Classification Using Deep Convolutional Neural Network," International Conference on Computers, Information Processing, and Advanced Education, October 2020, pp. 18-22.
[7] F. Shafait, A. Mian, M. Shortis, B. Ghanem, P. F. Culverhouse, D. Edgington, et al., "Fish Identification from Videos Captured in Uncontrolled Underwater Environments," ICES Journal of Marine Science, vol. 73, no. 10, pp. 2737-2746, November 2016.
[8] J. Fischer, Fish Identification Tools for Biodiversity and Fisheries Assessments: Review and Guidance for Decision-Makers, Rome: Food and Agriculture Organization of the United Nations, 2013.
[9] M. M. M. Fouad, H. M. Zawbaa, N. El-Bendary, and A. E. Hassanien, "Automatic Nile Tilapia Fish Classification Approach Using Machine Learning Techniques," 13th International Conference on Hybrid Intelligent Systems, December 2013, pp. 173-178.
[10] A. Domínguez, "A History of the Convolution Operation," IEEE Pulse, vol. 6, no. 1, pp. 38-49, January-February 2015.
[11] M. A. Iqbal, Z. Wang, Z. A. Ali, and S. Riaz, "Automatic Fish Species Classification Using Deep Convolutional Neural Networks," Wireless Personal Communications, vol. 116, no. 2, pp. 1043-1053, January 2021.
[12] N. E. M. Khalifa, M. H. N. Taha, and A. E. Hassanien, "Aquarium Family Fish Species Identification System Using Deep Neural Networks," International Conference on Advanced Intelligent Systems and Informatics, September 2018, pp. 347-356.
[13] J. H. Christensen, L. V. Mogensen, R. Galeazzi, and J. C. Andersen, "Detection, Localization and Classification of Fish and Fish Species in Poor Conditions using Convolutional Neural Networks," IEEE/OES Autonomous Underwater Vehicle Workshop, November 2018, pp. 1-6.
[14] B. S. Rekha, G. N. Srinivasan, S. K. Reddy, D. Kakwani, and N. Bhattad, "Fish Detection and Classification Using Convolutional Neural Networks," International Conference on Computational Vision and Bio-Inspired Computing, September 2019, pp. 1221-1231.
[15] J. N. Fabic, I. E. Turla, J. A. Capacillo, L. T. David, and P. C. Naval, "Fish Population Estimation and Species Classification from Underwater Video Sequences Using Blob Counting and Shape Analysis," IEEE International Underwater Technology Symposium, March 2013, pp. 1-6.
[16] K. M. Knausgård, A. Wiklund, T. K. Sørdalen, K. T. Halvorsen, A. R. Kleiven, L. Jiao, et al., "Temperate Fish Detection and Classification: A Deep Learning Based Approach," Applied Intelligence, in press.
[17] L. Wang, "WeFish: An Interactive Information Design Mobile App for Local Fishing," Thesis, College of Imaging Arts and Sciences, Rochester Institute of Technology, New York, NY, 2017.
[18] R. Froese and D. Pauly, "2021 FishBase," http://www.fishbase.org, June 2021.
[19] S. Pudaruth, N. Nazurally, C. Appadoo, S. Kishnah, M. Vinayaganidhi, I. Mohammoodally, et al., "SuperFish: A Mobile Application for Fish Species Recognition Using Image Processing Techniques and Deep Learning," International Journal of Computing and Digital Systems, vol. 10, no. 1, pp. 1-14, January 2021.