Betta fish species classification using light weight deep learning algorithm

Danishah Hana Muhammad Muhaimin Lim, Norizan Mat Diah, Zaidah Ibrahim, Zolidah Kasiran
School of Computing Sciences, College of Computing, Informatics and Mathematics, Universiti Teknologi MARA, Selangor, Malaysia

Article Info

Article history:

Received Mar 12, 2024
Revised Sep 27, 2024
Accepted Oct 8, 2024

Keywords:

Betta fish
Classification
Deep learning
Lightweight
MobileNet

ABSTRACT

Betta fish sellers and breeders often face challenges in accurately identifying Betta fish species due to variations in colors, patterns, and shapes, leading to potential financial losses and deceptive transactions. To address this issue, we developed a mobile application that employs MobileNet, a deep learning (DL) technique, to classify Betta fish species. The dataset, acquired from online stores, comprises 400 images, with 100 images representing each of the four studied Betta fish species: comb tail, delta tail, spade tail, and veil tail. Prior to model implementation, the dataset undergoes pre-processing with data augmentation techniques, including rotation, shear, zoom-in, horizontal flip, and brightness adjustments, enhancing the model performance. Training utilizes 80% of the data, with the remaining 20% allocated for testing. Three distinct MobileNet models are developed for males, females, and both genders combined, achieving accuracies of 70, 83.75, and 65%, respectively. These trained models are the foundation for a mobile application developed for the Android platform that enables users, particularly Betta fish sellers, and breeders, to efficiently classify Betta fish species, empowering them to set accurate prices based on the identified species.

Corresponding Author:

Norizan Mat Diah
School of Computing Sciences, College of Computing, Informatics and Mathematics
Universiti Teknologi MARA
40450 Shah Alam, Selangor, Malaysia
Email: norizan289@uitm.edu.my

1. INTRODUCTION

In recent years, the hobby of collecting and breeding Betta fish has evolved into a lucrative venture for many enthusiasts [1]. The allure of Betta fish lies in their captivating beauty, especially evident in their unique shapes and distinctive tail patterns. However, breeders often encounter challenges in accurately classifying different Betta fish species, such as Crowntail Betta, Veiltail Betta, Half Moon Betta, and Doubletail Betta [2]. In addition, because the male Betta fish has more attractive and colorful features than the females, they are worth more economically. This classification issue has prompted the need to implement Betta fish species recognition using deep learning (DL) tailored explicitly for mobile applications.

The pervasive presence of mobile devices in our daily lives is undeniable, with a projected ownership rate of over 90% among adults in developed countries by the end of 2023 [3]. The success of DL in various machine learning (ML) tasks has fuelled the integration of this technology into mobile applications. Recognizing this trend, implementing DL for Betta fish species classification in mobile applications becomes essential.

Betta fish, also known as Fighting fish, has gained popularity among fish enthusiasts due to its ease of care and vibrant aesthetics. The hobby of collecting these fish has transformed into a profitable source of income, with individual fish fetching high prices, reaching up to RM700 [4]. The unique color patterns and shapes of Betta fish contribute to their market value, resulting in an increasing demand and subsequent rise in selling prices. However, the variability in tail shapes, colors, and patterns poses a challenge for sellers and buyers alike. Not all enthusiasts can accurately recognize the species of Betta fish, leading to potential financial losses for sellers and the risk of buyers paying inflated prices for misidentified fish [5].

This research focuses on developing a mobile application for Betta fish species classification using lightweight DL models to address these challenges. The objective is to provide a user-friendly tool to identify Betta fish species accurately, empowering sellers and buyers. Considering the computational constraints of mobile devices, incorporating lightweight models is crucial, and it aims to optimize the classification process for real-time applications in the dynamic Betta fish market.

Not much research has been conducted on Betta fish classification, and no publicly available dataset can be used for comparative analysis. The ML approach has been applied to identify five species of Betta fish by extracting grey-level co-occurrence matrix (GLCM) features as input to the K-nearest neighbor (K-NN) classifier [6]. With 60 personal collection images per species for the data set, which totals up to 300 images, excellent classification has been achieved; however, the fish has to be in a specific angular direction. Mookdarsanit and Mookdarsanit [7] conducted research to create a region-based convolutional neural network (R-CNN) model named "SiamFishNet" that classifies the breed of an unknown Betta fish image based solely on the image itself. The researchers formulated this model using a dataset of 87,560 Betta fish images representing 12 different breeds of Bettas. The findings revealed that the model achieved an average precision of 84%, indicating its effectiveness in accurately identifying the breed of Betta fish. Another ML approach used the Gabor feature and artificial neural network classifier, but the results were not encouraging [8]. DL methods such as ResNet-50 have been utilized for Betta fish classification based on personal data collection and achieved 80% accuracy [9]. ResNet-50 is a heavyweight DL that consists of 48 convolutional layers, one MaxPool layer, and one average pool layer. However, a heavyweight DL usually needs high storage and high-power devices [10], which may be a barrier for users, especially small pet shops. Therefore, this research proposes to use a lightweight DL model that can achieve high accuracy at minimal cost and memory requirements while still being competitive with heavyweight models.

This paper is organized as follows. The next section discusses the works related to DL. Section 3 explains the classification method utilized in this research, the dataset used, and the various experimental results based on fine-tuning various hyperparameters. Section 4 discusses the result analysis, followed by a conclusion in the last section.

2. BACKGROUND STUDY

Operating DL models on edge devices poses significant challenges due to limited resources and computational capabilities. Using lightweight DL models has emerged as a crucial strategy to address this. Lightweight algorithms are designed to be computationally efficient with a small memory footprint, making them suitable for deployment on resource-constrained devices like mobile phones, internet of thing (IoT) devices, and edge computing platforms [11], [12].

These lightweight models aim to reduce computational demands by optimizing network structures and employing efficient building methods. The concept of lightweight algorithms minimizes the number of parameters and computations, making them ideal for real-time processing on devices with limited computational resources [13]. Additionally, these models typically have smaller memory footprints, an advantageous feature for devices with limited random-access memory (RAM).

Several notable lightweight convolutional neural network (CNN) models have been proposed to address these challenges. SqueezeNet, introduced by the Berkeley and Stanford research teams in 2016, and MobileNet, presented by the Google team in 2017, are noteworthy examples. ShuffleNet, proposed by Ignore Technology in 2017, and EfficientNet, introduced by Google in 2018, further contribute to the arsenal of lightweight models. These models optimize network structures, decrease the number of parameters, and enhance accuracy, even achieving full convolution accuracy [14]–[16]. ShuffleNet utilizes both group convolution and channel shuffle operations to simplify pointwise convolutions. Group convolution divides the input channels into groups, reducing computational complexity. Channel shuffle is then applied to exchange information between groups, promoting information flow and maintaining model efficiency [17], [18] using group convolution and channel shuffle to simplify pointwise convolution. MicroNet's micro-factorized convolution and adjusting node connectivity and network width aim to balance model efficiency and expressive power [19].

MobileNet stands out as a particularly lightweight deep CNN. It is smaller and faster than many well-known classification models, making it suitable for image detection, face attributes, and image analysis [20]. MobileNet utilizes a simplified architecture with depth-wise separable convolutions, providing an efficient solution for both mobile and embedded devices [21], [22]. The advantages of MobileNet lie in its real-time performance on tasks like image classification, object detection, and segmentation. The architecture's flexibility allows users to control the trade-off between model accuracy and computational efficiency through hyperparameters like the width and resolution multipliers. This adaptability proves invaluable when optimizing models for specific deployment scenarios, offering low-latency responses for various applications [20].

In the context of this research, the focus is on constructing a network based on MobileNet for Betta fish species classification. MobileNet's main contribution is the proposal of deep separable convolution, a decomposition form significantly reducing computational complexity and model size. This research aims to leverage MobileNet's lightweight design to create an efficient Betta fish species classification system, catering to the unique demands of edge devices and contributing to real-time applications in the field of aquatic species identification.

3. METHOD AND MATERIAL

This conceptual framework outlines the systematic procedures employed in the research, delineating key phases encompassing dataset collection, pre-processing, model architecture design, training, evaluation, and experimentation. It serves as a structured guide, offering insight into the anticipated methods and materials deployed in the study.

3.1. Data collection

This study collected 300 images of seven types of Betta fish from Betta fish sellers that do E-commerce in Lazada, Instagram, and Facebook. These data are all in .jpg format, smaller than the .png file. Table 1 lists the collected data and total images for every Betta fish species. The number of images for comb tail, delta tail, and double tail is 34, respectively. Forty-six images were collected each for spade tail and veil tail. Crown tail and halfmoon tail have 64 and 58 images, respectively. Overall, 316 images were obtained from online stores. The data was then augmented, and the images were resized as part of the pre-processing.

Table 1. The total number of images acquired for each species

Betta Fish Species	Number of Images
Comb Tail	34
Crown Tail	64
Delta Tail	34
Double Tail	34
Halfmoon Tail	58
Spade Tail	46
Veil Tail	46
Total	316

3.2. Data pre-processing

Data pre-processing is the most significant and influential factor in the generalization performance of a supervised ML algorithm [23]. After the dataset was collected, all the images were resized to 224×244 pixels for fitting into MobileNet. Then, as shown in Figure 1, the dataset was augmented due to the small amount of data by using Figure 1(a) a rotation range of 0.2, Figure 1(b) a shear range of 0.2, Figure 1(c) a zoom-in range of 0.2, Figure 1(d) a horizontal flip is equal to true, and Figure 1(e) a brightness range of 0.5 to 1.5.

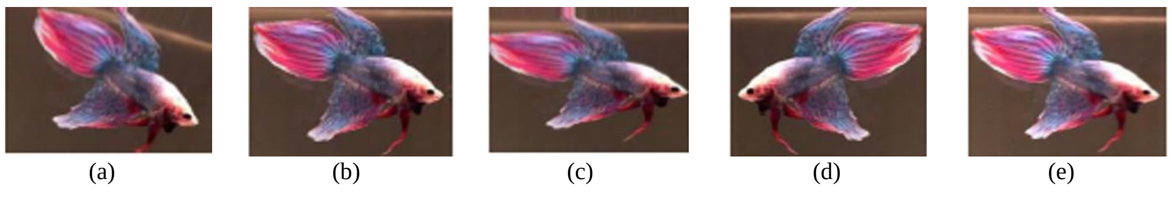

Figure 1. Augmentation codes and sample results of (a) rotation range, (b) shear range, (c) zoom-in range, (d) horizontal flip is equal, and (e) brightness range

It is crucial to augment image data properly to increase accuracy and prevent overfitting [24]. The parameters of augmentation were adjusted four times to gain more training datasets. After augmentation, the total number of Betta fish images for every species has increased significantly. Before the data augmentation process, the total number of images for every species was mostly lower than 100. Based on Table 2, a total of 700 images were created from the original 316 images, where each of the seven classes contains 100 images. From each of the seven classes, 80 images (80%) were used for training, and 20 images (20%) were used for testing. Figure 2 shows a snippet of the Betta fish dataset that was divided accordingly.

Table 2. The total number of Betta fish images before and after augmentation

	Total Number of Images	
Betta Fish Species	Before Data Augmentation	After Data Augmentation
Comb Tail	34	240
Crown Tail	64	240
Delta Tail	34	240
Double Tail	34	240
Halfmoon Tail	58	240
Spade Tail	46	240
Veil Tail	46	240
Total	316	1680

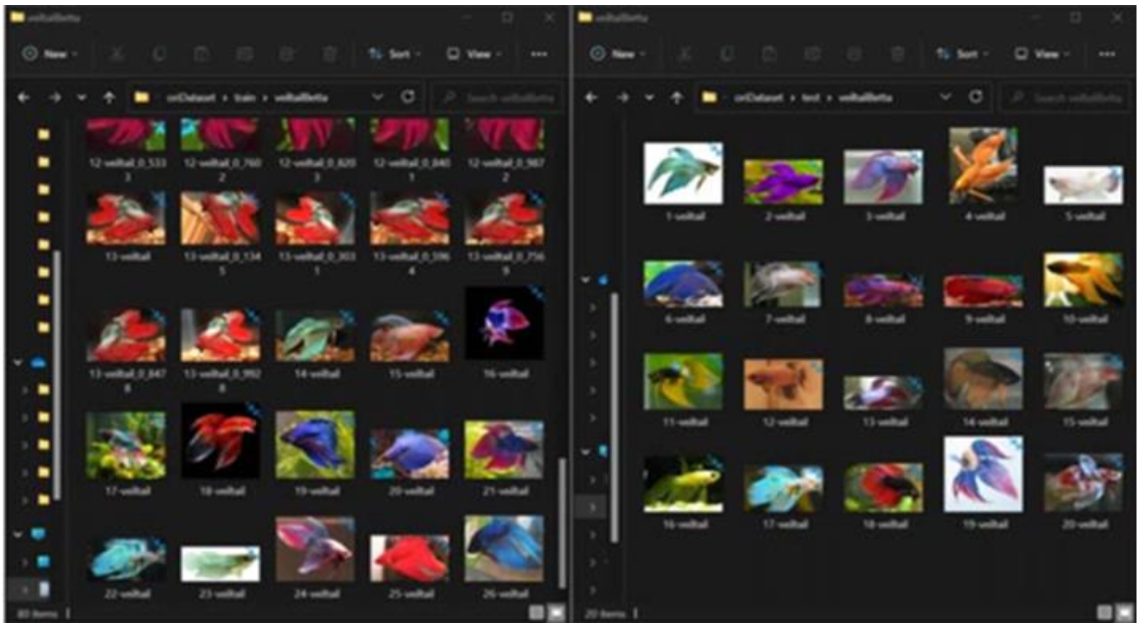

Figure 2. The snippet of the Betta fish dataset

3.3. Model architecture

This research used MobileNet to classify Betta fish species by specific architectural configurations. The primary purpose of choosing MobileNet is to address the challenges associated with deploying DL models on resource-constrained devices, such as mobile phones while achieving accurate and efficient Betta fish classification. MobileNet's architecture is characterized by its lightweight design, making it well-suited for real-time image processing on devices with limited computational capabilities.

The critical innovation in MobileNet is the use of depthwise separable convolutions, which significantly reduces the number of parameters and computations compared to traditional convolutional layers [25]–[27]. This design enables MobileNet to maintain satisfactory accuracy while significantly lowering the model's size, making it practical for deployment on mobile platforms. An overview of the MobileNet architecture is illustrated in Figure 3. It consists of 28 layers, including a deep convolution layer, 1×1 point convolution layer, batch norm, ReLU, average collecting layer, and SoftMax.

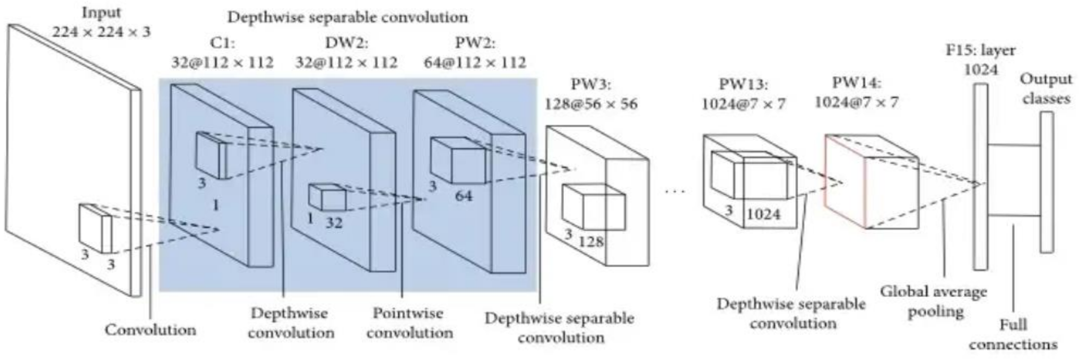

Figure 3 MobileNet architecture [28], [29]

3.4. Experimental approach and optimization for MobileNet training

Several experiments have been meticulously conducted to create a reliable and accurate model for Betta fish species classification using MobileNet. These experiments aimed to enhance the classification accuracy through two primary approaches: increasing the number of training images and fine-tuning four key hyper-parameters, namely epoch, dropout layer, pooling layer, and batch size. The culmination of these efforts involved eight distinct experiments, each contributing valuable insights to the overall model performance.

3.4.1. Experiment 1: comparison between two sets of datasets

In this experiment, there are two sets of datasets. In the first set, the testing images consist of only the original images, while in the second set, the testing images consist of a combination of the original and augmented images. The experiment involved seven classes representing different Betta fish species and 700 images. The division between training and testing datasets was executed with 80% for training and 20% for testing, adhering to DL model development standards [30]. A hyperparameter called "epochs" determines how many times the learning algorithm will run over the training dataset [31], [32]. This experiment was run for 80 epochs, and it took two hours for each model to learn. Table 3 illustrates the results which show that the model with combined images (validation accuracy of 0.3357) performs better than the one with the original images (validation accuracy of 0.2786). However, an overfitting problem occurs where the value of the validation accuracy is very much lower than the training accuracy. Therefore, in the next experiment, a dropout layer is added.

Table 3. Experiment 1 training results on the original image dataset and combined dataset

Dataset	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
Original Images	0.9036	0.3298	0.2786	4.5028
Combined Images (Original and Augmented)	0.9125	0.3007	0.3357	6.4944

3.4.2. Experiment 2: comparison between using a dropout layer and without a dropout layer

Dropout is an efficient way to reduce overfitting [33]. It randomly sets input units to 0 with a pre-determined percentage at each step during training time [34]. The dataset for this experiment is the same as in Experiment 1. This experiment was run for 40 epochs, and it took one hour for each model to learn. Table 4 illustrates the outcome of the experiment and it indicates that the model with combined images (validation accuracy of 0.3643) showed a better result than the model with original images (validation accuracy of 0.2357) when a dropout layer was added. Furthermore, the performance has slightly improved compared to Experiment 1. However, overfitting still occurs. Hence, the total number of images is added with different pooling layers in Experiment 3.

Table 4. Experiment 2 training results on the original image dataset and combined dataset with a dropout layer

Dataset	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
Original Images	0.6464	1.0404	0.2357	4.5466
Combined Images (Original and Augmented)	0.5875	1.2169	0.3643	3.8393

3.4.3. Experiment 3: comparison between max pooling and average pooling

Max pooling and average pooling are the two types of pooling layers. The maximum value from the portion of the image that the kernel (filter) has covered is returned by max pooling [35], [36]. On the contrary, the average of all the values from the portion of the image covered by the kernel is returned by average pooling [37]. Two things were selected, which are the pool size and a stride, to perform max pooling and average pooling. The stride controls how many pixels the window will move across the image pooling [38], [39]. This experiment compared max pooling and average pooling kernel size 3×3 with a pool size of 7 and stride 1. Since the two previous experiments have proven that a combination dataset of original images and augmented images, with a dropout layer is a better option, this case is also applied in Experiment 3. Moreover, 1680 images were added, 200 for training (80%) and 40 for testing (20%) for each of the seven classes. At the end of this experiment, two models were trained. This experiment was run for ten epochs, and it took five hours for each model to learn. Table 5 illustrates the results of the experiments. By referring to Table 5, we can see that the model with max pooling is better than the model with average pooling since there is no overfitting for max pooling. Overfitting occurs with average pooling where the training accuracy is higher than its validation accuracy. However, the accuracy achieved by the model with max pooling was not high. Thus, different pooling sizes were experimented with and compared in the next experiment.

Table 5. Experiment 3 training results on max pooling and average pooling

Pooling Layer	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
Max Pooling	0.2000	2.2227	0.2000	2.2842
Average Pooling	0.2736	1.8567	0.2464	2.2027

3.4.4. Experiment 4: comparison between pool size

In the previous experiment, the pool size used was 7. In this experiment, a comparison between max pooling and average pooling with a pool size of 6 is performed. This experiment was run for ten epochs and took five hours for each model to learn. Table 6 lists the results and the model with max pooling performs slightly better than the model with average pooling since the overfitting that occurs by max pooling is less than average pooling. However, the accuracy was still not high. Thus, in the next experiment, different dropout hyperparameters were compared.

Table 6. Experiment 4 training results on max pooling and average pooling with a pool size of 6

Pooling Layer	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
Max Pooling	0.4193	1.5644	0.2429	3.0617
Average Pooling	0.5157	1.3223	0.2500	5.5521

3.4.5. Experiment 5: comparison between dropout hyperparameter

Experiment 5 compared the dropout hyperparameters of 0.2 and 0.5. This experiment was run for ten epochs and took five hours for each model to learn. Table 7 illustrates that the model with a dropout hyperparameter of 0.2 (validation accuracy of 0.2429) shows better results than the model with a dropout hyperparameter of 0.5 (validation accuracy of 0.2). The following experiment compared two different hyperparameter values of the batch size to determine which would arrive at a better validation accuracy.

Table 7. Experiment 5 training results on dropout hyperparameters

Dropout Hyperparameters	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
0.2	0.4793	1.4193	0.2429	2.4662
0.5	0.2000	2.2227	0.2000	2.2842

3.4.6. Experiment 6: comparison between batch size hyperparameter

The batch size, a gradient descent hyperparameter, determines how many training samples must be examined before the model's internal parameters are updated [40], [41]. Experiment 6 compared the batch size hyperparameters of 2 and 5. This experiment was run for ten epochs and took five hours for each model to learn. Table 8 shows that the model with a batch size hyperparameter of 5 produces better results (validation accuracy of 0.2464) than the model with a batch size hyperparameter of 2 (validation accuracy of 0.2).

Table 8. Experiment 6 training results on batch size hyperparameters

Batch Size Hyperparameters	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
2	0.2000	2.2227	0.2000	2.2842
5	0.2507	3.2279	0.2464	4.4298

3.4.7. Experiment 7: retrain experiment 3 with 200 epochs

Since the best model so far was achieved in experiment 3 where the model was trained with max pooling with a pool size of 7, dropout hyperparameter of 0.5, and batch size hyperparameter of 2, this experiment used the same hyperparameters but with 200 epochs. It took three days for this model to learn, but the validation accuracy produced was lower than in experiment 3. Since this model did not produce high validation accuracy as shown in Table 9, the collected data was analyzed again. It was found that the background images and the appearances of the male Betta fish and female Betta fish differ significantly. Therefore, in the next experiment, the dataset was separated according to the Betta fish gender, male and female, with four classes only due to the lack of female Betta fish images, time, and hardware constraints.

Table 9. Experiment 7 training results on retrain experiment 3 with 200 epochs

Epoch	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
200	0.9993	0.0043	0.4464	5.1301

3.4.8. Experiment 8: three models

Three models were trained in this experiment: the male Betta fish dataset, the female Betta fish dataset, and the combination of the male and female Betta fish dataset. The dataset was reduced to 400 images in each model. Based on experiment 3, it turned out that max pooling, pool size of 7, dropout of 0.5, and batch size of 2 showed the best result. Therefore, these hyperparameters were used in this experiment. Besides, these models used stochastic gradient descent (SGD) as an optimizer and SoftMax as an activation function. This experiment was run for 80 epochs and took three hours for each model to learn. As shown in Table 10, the validation accuracy was much improved compared to the previous experiments.

Table 10. Experiment 8 training results on three models

Model	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
Male	0.9875	0.0237	0.7000	7.4765
Female	0.9563	0.1721	0.8375	0.9622
Male and Female	0.9656	0.1016	0.6500	2.0234

4. RESULTS AND DISCUSSION

This study conducted a comprehensive series of eight experiments to optimize Betta fish species classification through DL, yielding 14 distinct models. The primary objective was to develop a robust classification system to classify Betta fish species. These experiments were executed with meticulous attention to factors such as data augmentation, hyperparameter tuning, and the utilization of the MobileNet model for its efficiency in large-scale image classification processing. The culmination of these efforts is encapsulated in the comparison of the 14 models, detailed in Table 11. Within Table 11, the performance of each model is scrutinized, and the findings reveal that the final three models from experiment 8 exhibit the most promising results. The ensuing section delves into a detailed analysis of these outcomes, shedding light on the key factors influencing the superior performance of the selected models.

The Betta fish classification model, employing MobileNet architecture, demonstrates exceptional accuracy by utilizing max pooling, a pool size of 7, a dropout rate of 0.5, and a batch size of 2, coupled with SGD optimizer and SoftMax as the activation function across 80 epochs. The validation accuracy for male Betta fish reaches 0.7, while female Betta fish achieves an impressive 0.8375. However, combining both male and female images results in a slightly lower accuracy of 0.65. Notably, the decision to employ separate models for male and female classification proves advantageous, highlighting the substantial differences in shape and color between male and female Betta fish that impact accurate classification. It underscores the importance of tailored models for distinct genders to optimize classification performance.

Table 11. Summary of experimental results with various training models

Experiment	Dataset	Accuracy		Loss	
		Training	Validation	Training	Validation
1	Original	0.9036	0.2786	0.3298	4.5028
	Combined	0.9125	0.3357	0.3007	6.4944
2	Original	0.6464	0.2357	1.0404	4.5466
	Combined	0.5875	0.3643	1.2169	3.8393
3	Combined (Max Pooling)	0.2000	0.2000	2.2227	2.2842
	Combined (Average Pooling)	0.2736	0.2464	1.8567	2.2027
4	Combined (Max Pooling, Pool Size = 6)	0.4193	0.2429	1.5644	3.0617
	Combined (Average Pooling, Pool Size = 6)	0.5157	0.2500	1.3223	5.5521
5	Combined (Dropout (0.2))	0.4793	0.2429	1.4193	2.4662
	Combined (Dropout (0.5))	0.2000	0.2000	2.2227	2.2842
6	Combined (Batch Size = 2)	0.2000	0.2000	2.2227	2.2842
	Combined (Batch Size = 5)	0.2507	0.2464	3.2279	4.4298
7	Combined (Epoch = 200)	0.9993	0.4464	0.0043	5.1301
8	Male	0.9875	0.7000	0.0237	7.4765
	Female	0.9563	0.8375	0.1721	0.9622
	Male and Female	0.9656	0.6500	0.1016	2.0234

4.1. Mobile application development

The mobile application development involves harnessing the capabilities of the trained models to produce a user-friendly application tailored for Betta fish species classification on the Android platform. Integrating an intuitive interface within the application enables users to seamlessly capture or upload images for instantaneous species identification. This feature-rich application aims to provide users with a straightforward and engaging experience, allowing them to actively participate in the real-time identification of Betta fish species easily and accurately.

Figure 4 illustrates a sample interface of the Betta fish species classification mobile application. Figure 4(a) illustrates the main page, which consists of two buttons, namely '*Buka Kamera*' to capture the image of the Betta fish using the mobile's camera and '*Buka Galeri*' to select a saved image stored in the gallery to classify the species. Figure 4(b) will be displayed if a user clicks the '*Buka Kamera*' button, while Figure 4(c) will be shown if a user clicks the '*Buka Galeri*' button. Users can also press the back arrow button to navigate to the previous page. Figure 5 shows the result page, where the name of the Betta fish species and its range of prices in Ringgit Malaysia (RM) are shown. The user can then repeat this process for the next Betta fish image classification.

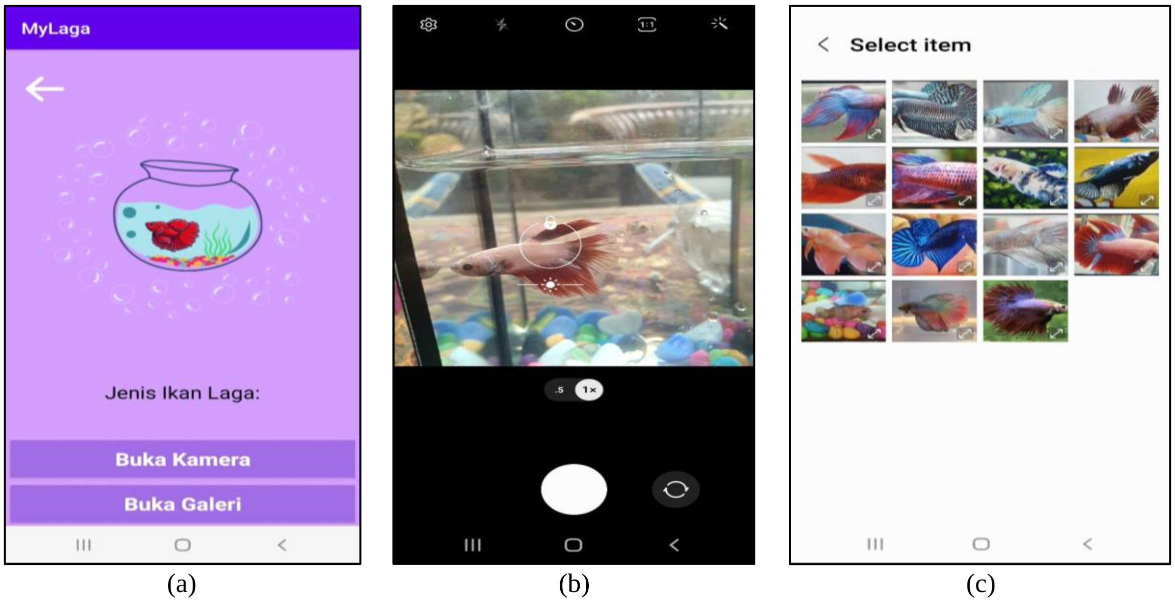

Figure 4. Sample interface of the mobile application of (a) main page, (b) page to capture the image after clicking the '*Buka Kamera*' button, and (c) page to select an image after clicking the '*Buka Galeri*' button

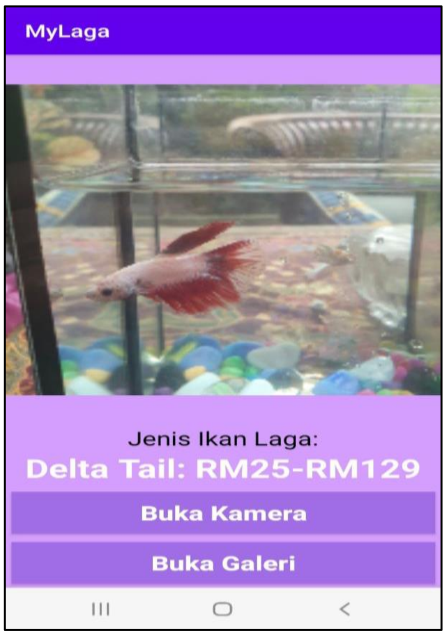

Figure 5. Sample interface for the result page

5. CONCLUSION AND FUTURE WORK

In conclusion, this research successfully explores and implements the MobileNet architecture to classify Betta fish species, addressing challenges faced by breeders and sellers in accurately identifying these aquatic species. MobileNet, which uses depthwise separable convolutions, is an efficient solution for deployment on resource-constrained devices. The crucial finding is that having separate models for male and female classifications improves accuracy significantly, highlighting the nuanced differences in shape and color between genders. This research not only contributes to the advancement of the Betta fish classification technique but also provides a practical tool for industry stakeholders to enhance their decision-making processes and reduce monetary losses associated with misclassification.

Concerning future work, several avenues exist for enhancing the Betta fish species classification model based on MobileNet architecture. Expanding the dataset to include a broader spectrum of Betta fish images with diverse colors, patterns, and tail shapes can contribute to the improvement of model generalization and reduce the overfitting problem. Considering the potential benefits, utilizing pre-trained models, and experimenting with advanced data augmentation techniques could further enhance the model's robustness. Investigating ensemble learning approaches and integrating user feedback mechanisms in the mobile application can contribute to continuous model improvement. Optimization for real-time deployment, compatibility with various mobile devices, and collaboration with aquatic experts to incorporate domain knowledge are crucial considerations. These future directions aim to refine the model’s accuracy, usability, and adaptability to benefit Betta fish breeders and sellers.

ACKNOWLEDGMENTS

Thanks are due to the College of Computing, Informatics and Mathematics at Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia, for their support in facilitating this work.

REFERENCES

[1] A. Sinha and P. K. Pandey, *Breeding and culture of freshwater ornamental fish*. London: CRC Press, 2023. doi: 10.1201/9781003456858.
[2] K. Tantayakul and W. Panichpattanakul, “A comparative study of machine learning for IOS based on Siam Betta mobile application,” in *2020 - 5th International Conference on Information Technology (InCIT)*, IEEE, Oct. 2020, pp. 104–109. doi: 10.1109/InCIT50588.2020.9310932.
[3] J. Wang, B. Cao, P. Yu, L. Sun, W. Bao, and X. Zhu, “Deep learning towards mobile applications,” in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, Jul. 2018, pp. 1385–1393. doi: 10.1109/ICDCS.2018.00139.

[4] Bernama, “Fighting fish stall attracts large crowd,” New Straits Times. Accessed: Oct. 18, 2022. [Online]. Available: https://www.nst.com.my/news/nation/2022/03/783570/fighting-fish-stall-attracts-large-crowd

[5] Y. Heningtyas, F. Rahmi, and K. Muludi, “Implementation of density-based clustering in betta fish image segmentation (in Indonesian: *Implementasi density-based clustering pada segmentasi citra betta fish*),” *J. Teknoinfo*, vol. 16, no. 1, p. 8, Jan. 2022, doi: 10.33365/jti.v16i1.1273.

[6] Z. Abidin, Rusliyawati, Permata, F. Ariany, I. Solehudin, and A. Junaidi, “Betta fish image identification using feature extraction GLCM and k-nearest neighbour classification,” in *2022 International Conference on Information Technology Research and Innovation (ICITRI)*, IEEE, Nov. 2022, pp. 156–161. doi: 10.1109/ICITRI56423.2022.9970209.

[7] L. Mookdarsanit and P. Mookdarsanit, “SiamFishNet: the deep investigation of siamese fighting fishes,” *Int. J. Appl. Comput. Technol. Inf. Syst.*, vol. 8, no. 2, 2019, [Online]. Available: http://203.158.98.12/actisjournal/index.php/IJACTIS/article/view/256

[8] S. Hidayat, A. Y. Rahman, and Istiadi, “Betta fish image classification using artificial neural networks with Gabor extraction features,” in *2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*, IEEE, Jun. 2022, pp. 270–273. doi: 10.1109/CyberneticsCom55287.2022.9865509.

[9] D. M. Hibban and W. F. Al Maki, “Classification of ornamental betta fish using convolutional neural network method and grabcut segmentation,” in *2021 International Conference on Data Science and Its Applications (ICoDSA)*, IEEE, Oct. 2021, pp. 102–109. doi: 10.1109/ICoDSA53588.2021.9617213.

[10] O. Ukwandu, H. Hindy, and E. Ukwandu, “An evaluation of lightweight deep learning techniques in medical imaging for high precision COVID-19 diagnostics,” *Healthc. Anal.*, vol. 2, p. 100096, Nov. 2022, doi: 10.1016/j.health.2022.100096.

[11] S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, “Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions,” *J. Ambient Intell. Humaniz. Comput.*, vol. 15, no. 2, pp. 1625–1642, Feb. 2024, doi: 10.1007/s12652-017-0494-4.

[12] M. G. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, “Machine learning at the network edge: a survey,” *ACM Comput. Surv.*, vol. 54, no. 8, pp. 1–37, Nov. 2022, doi: 10.1145/3469029.

[13] K. Kim, S.-J. Jang, J. Park, E. Lee, and S.-S. Lee, “Lightweight and energy-efficient deep learning accelerator for real-time object detection on edge devices,” *Sensors*, vol. 23, no. 3, p. 1185, Jan. 2023, doi: 10.3390/s23031185.

[14] A. Howard *et al.*, “Searching for MobileNetv3,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2019, pp. 1314–1324. doi: 10.1109/ICCV.2019.00140.

[15] J. Yang, L. Zhang, X. Tang, and M. Han, “CodnNet: a lightweight CNN architecture for detection of COVID-19 infection,” *Appl. Soft Comput.*, vol. 130, p. 109656, Nov. 2022, doi: 10.1016/j.asoc.2022.109656.

[16] M. Tan and Q. V. Le, “EfficientNet: rethinking model scaling for convolutional neural networks,” in *Proceedings of the 36 th International Conference on Machine Learning*, Long Beach, California: PMLR 97, May 2019. [Online]. Available: http://arxiv.org/abs/1905.11946

[17] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, “ShuffleNet v2: practical guidelines for efficient CNN architecture design,” in *Lecture Notes in Computer Science*, Computer V., Springer, Cham, 2018, pp. 122–138. doi: 10.1007/978-3-030-01264-9_8.

[18] X. Zhang, X. Zhou, M. Lin, and J. Sun, “ShuffleNet: an extremely efficient convolutional neural network for mobile devices,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018, pp. 6848–6856. doi: 10.1109/CVPR.2018.00716.

[19] Y. Li *et al.*, “MicroNet: improving image recognition with extremely low flops,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2021, pp. 458–467. doi: 10.1109/ICCV48922.2021.00052.

[20] K. Dey, M. M. Hassan, M. M. Rana, and M. H. Hena, “Bangladeshi indigenous fish classification using convolutional neural networks,” in *2021 International Conference on Information Technology (ICIT)*, IEEE, Jul. 2021, pp. 899–904. doi: 10.1109/ICIT52682.2021.9491681.

[21] U. Kulkarni, M. S.M., S. V. Gurlahosur, and G. Bhogar, “Quantization friendly MobileNet (qf-MobileNet) architecture for vision based applications on embedded platforms,” *Neural Networks*, vol. 136, pp. 28–39, Apr. 2021, doi: 10.1016/j.neunet.2020.12.022.

[22] G. Lu, W. Zhang, and Z. Wang, “Optimizing depthwise separable convolution operations on GPUs,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 1, pp. 70–87, Jan. 2022, doi: 10.1109/TPDS.2021.3084813.

[23] K. Maharana, S. Mondal, and B. Nemade, “A review: data pre-processing and data augmentation techniques,” *Glob. Transitions Proc.*, vol. 3, no. 1, pp. 91–99, Jun. 2022, doi: 10.1016/j.gltp.2022.04.020.

[24] N. E. Khalifa, M. Loey, and S. Mirjalili, “A comprehensive survey of recent trends in deep learning for digital images augmentation,” *Artif. Intell. Rev.*, vol. 55, no. 3, pp. 2351–2377, Mar. 2022, doi: 10.1007/s10462-021-10066-4.

[25] S. Bouguezzi, H. Ben Fredj, T. Belabed, C. Valderrama, H. Faiedh, and C. Souani, “An efficient fpga-based convolutional neural network for classification: ad-MobileNet,” *Electronics*, vol. 10, no. 18, p. 2272, Sep. 2021, doi: 10.3390/electronics10182272.

[26] K. KC, Z. Yin, M. Wu, and Z. Wu, “Depthwise separable convolution architectures for plant disease classification,” *Comput. Electron. Agric.*, vol. 165, p. 104948, Oct. 2019, doi: 10.1016/j.compag.2019.104948.

[27] W. Wang, Y. Hu, T. Zou, H. Liu, J. Wang, and X. Wang, “A new image classification approach via improved MobileNet models with local receptive field expansion in shallow layers,” *Comput. Intell. Neurosci.*, vol. 2020, pp. 1–10, Aug. 2020, doi: 10.1155/2020/8817849.

[28] A. G. Howard *et al.*, “MobileNets: efficient convolutional neural networks for mobile vision applications,” *arXiv*, Apr. 2017, [Online]. Available: http://arxiv.org/abs/1704.04861

[29] W. Wang, Y. Li, T. Zou, X. Wang, J. You, and Y. Luo, “A novel image classification approach via dense-MobileNet models,” *Mob. Inf. Syst.*, vol. 2020, pp. 1–8, Jan. 2020, doi: 10.1155/2020/7602384.

[30] S. W. P. Listio, “Performance of deep learning inception model and MobileNet model on gender prediction through eye image,” *Sinkron*, vol. 7, no. 4, pp. 2593–2601, Nov. 2022, doi: 10.33395/sinkron.v7i4.11887.

[31] S. Kaur, H. Aggarwal, and R. Rani, “Hyper-parameter optimization of deep learning model for prediction of Parkinson’s disease,” *Mach. Vis. Appl.*, vol. 31, no. 5, p. 32, Jul. 2020, doi: 10.1007/s00138-020-01078-1.

[32] Y. N. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto, “Attack classification of an intrusion detection system using deep learning and hyperparameter optimization,” *J. Inf. Secur. Appl.*, vol. 58, p. 102804, May 2021, doi: 10.1016/j.jisa.2021.102804.

[33] C. Ha, V.-D. Tran, L. Ngo Van, and K. Than, “Eliminating overfitting of probabilistic topic models on short and noisy text: the role of dropout,” *Int. J. Approx. Reason.*, vol. 112, pp. 85–104, Sep. 2019, doi: 10.1016/j.ijar.2019.05.010.

[34] J. Zhang, Y. Sun, L. Guo, H. Gao, X. Hong, and H. Song, “A new bearing fault diagnosis method based on modified convolutional neural networks,” *Chinese J. Aeronaut.*, vol. 33, no. 2, pp. 439–447, Feb. 2020, doi: 10.1016/j.cja.2019.07.011.

[35] J. Hyun, H. Seong, and E. Kim, “Universal pooling – a new pooling method for convolutional neural networks,” *Expert Syst. Appl.*, vol. 180, p. 115084, Oct. 2021, doi: 10.1016/j.eswa.2021.115084.

[36] S. S. Husain and M. Bober, “REMAP: multi-layer entropy-guided pooling of dense CNN features for image retrieval,” *IEEE Trans. Image Process.*, vol. 28, no. 10, pp. 5201–5213, Oct. 2019, doi: 10.1109/TIP.2019.2917234.

[37] S. Balachandran, “Machine learning - max and average pooling,” DEV Community. Accessed: Oct. 18, 2023. [Online]. Available: https://dev.to/sandeepbalachandran/machine-learning-max-average-pooling-1366
[38] A. Zafar *et al.*, “A comparison of pooling methods for convolutional neural networks,” *Appl. Sci.*, vol. 12, no. 17, p. 8643, Aug. 2022, doi: 10.3390/app12178643.
[39] C. J. Shallue, J. Lee, J. Antognini, J. Sohl-Dickstein, R. Frostig, and G. E. Dahl, “Measuring the effects of data parallelism on neural network training,” *J. Mach. Learn. Res.*, vol. 20, pp. 1–49, 2019.
[40] N. Akhtar and U. Ragavendran, “Interpretation of intelligence in CNN-pooling processes: a methodological survey,” *Neural Comput. Appl.*, vol. 32, no. 3, pp. 879–898, Feb. 2020, doi: 10.1007/s00521-019-04296-5.
[41] C. Garbin, X. Zhu, and O. Marques, “Dropout vs. batch normalization: an empirical study of their impact to deep learning,” *Multimed. Tools Appl.*, vol. 79, no. 19–20, pp. 12777–12815, May 2020, doi: 10.1007/s11042-019-08453-9.

BIOGRAPHIES OF AUTHORS

Danishah Hana Muhammad Muhaimin Lim received her Bachelor's degree in Computer Science (Hons.) Multimedia Computing at Universiti Teknologi MARA, Shah Alam, Selangor Malaysia. She has work experience as a Business Analyst at Sime Darby Plantation Berhad and as a coordinator at Voxel Studio. Her research lines are in image recognition and classification using deep learning. She can be contacted at email: danishahhanaaa@gmail.com.

Norizan Mat Dia is an Associate Professor at the School of Computing Sciences, College of Computing, Informatics, and Mathematics, Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia. She holds a Ph.D. in information sciences from the National University of Malaysia. Her research focuses on gamification, game engines, real-time feedback (algorithms), and machine learning. She can be contacted at: norizan289@uitm.edu.my.

Zaidah Ibrahim is currently an Associate Professor at the School of Computing Sciences, College of Computing, Informatics and Media, Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia. Her research areas are computer vision and deep learning. She can be contacted at email: zaidah@tmsk.uitm.edu.my.

Zolidah Kasiran is a senior lecturer in the College of Computing, Informatics, and Mathematics, Universiti Teknologi MARA, Shah Alam Selangor Malaysia. She has a PhD in Information Technology from UiTM. Her research focuses on Computer Networks, Cybersecurity, and Web applications. She can be contacted at email: zolid808@uitm.edu.my.