Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 3, **Number** 2, November 2023
https://doi.org/10.47709/brilliance.v3i2.3270

Leveraging VGG16 for Fish Classification in a Large-Scale Dataset

Karina Auliasari[1*], Mohamed Wasef[2], Mariza Kertaningtyas[3]
[1,3]Institut Teknologi Nasional Malang, Indonesia, [2] Kafr El-Sheikh University, Egypt
[1]karina.auliasari@lecturer.itn.ac.id, [3]mariza_kertaningtyas@lecturer.itn.ac.id

open access

***Corresponding Author**

Article History:
Submitted: 29-11-2023
Accepted: 07-12-2023
Published: 15-12-2023

Keywords:
Fish; Classification; VGG-16; CNN; Deep Learning

ABSTRACT

This research investigates the performance of two machine learning models: a general deep learning model and a specialized VGG16-based model for fish classification. The deep learning model exhibited promising learning progress, with both accuracy and validation accuracy increasing with training epochs. However, the consistently higher accuracy compared to validation accuracy suggests potential overfitting. Monitoring validation accuracy and stopping training upon plateauing are crucial to mitigating this issue. The VGG16 model demonstrated exceptional effectiveness in fish classification, achieving remarkable precision, recall, and F1-scores exceeding 0.97, 0.96, and 0.98, respectively, for all fish types. This highlights the model's robust generalizability and ability to differentiate diverse fish species. The research identifies significant potential applications for both models in areas like fisheries management, aquaculture, and ecological research. Research attempts may be strengthened, sustainable fishing methods used, and effective fish population monitoring made possible by these models. To further improve the models' capabilities, future work should concentrate on growing training datasets, integrating with other technologies, and investigating explainable AI strategies. To sum up, this study highlights the useful applications of machine learning across a range of domains and stresses the need to detect and manage overfitting in order to guarantee dependable model performance and get ideal outcomes.

INTRODUCTION

Manually classifying fish species is difficult, time-consuming, and experience-based, particularly when dealing with related fish species. By paying closer attention to skin contours, color, and body form, we can distinguish the differences. It is difficult to develop an automatic fish categorization system on an image with different lighting and backdrops. As a result, a number of prior studies have been conducted to create an image-based application for the categorization of fish species. Since fish species were manually recognized by observation, it was required to learn or retain a variety of fish features in order to identify different kinds of fish (Qin et al., 2016). Deep learning techniques combined with advanced technology and artificial intelligence (AI) make it easier to identify a wide variety of fish species. Convolutional neural networks (CNNs) are among the most widely utilized deep learning techniques available today. CNN is a popular object categorization technique that builds upon the multilayer perceptron (MLP). CNN has been effective in classifying images because it uses numerous dimensions that impact an object's overall size (He et al., 2016).

Several studies have classified fish species using CNN ideas. In comparison to previous comparable studies, Rathi et al.'s unique CNN-based approach to fish classification produced an accuracy of 96.29 percent for 21 different species of fish. To improve the categorization findings, 27,142 RGB photos were used in combination with grayscale, noise reduction, and other pertinent approaches (Rathi et al., 2017). Khalifa et al. employed a streamlined version of AlexNet in a different investigation to identify many fish species. Eight species made up the data set, of which 191 subspecies were trained. With AlexNet, the accuracy was 85.59 percent and 85.41 percent, respectively (Khalifa et al., 2018). Since they incorporate superior learning capabilities than AlexNet, other models, such as the VGG16 and VGG19 by Simonyan and Zisserman (Simonnyan et al., 2014), are currently among the best CNN models used for classification. VGG16 produced a 93% accuracy rate in classifying fish underwater in the Kratzert et al. research (Kratzert et al, 2018). In order to improve fish categorization, Mingwang also used VGG16 with dropout and batch normalization techniques, achieving 97% accuracy (Mingwang et al., 2017). In their comparison of VGG16 with VGG19 results for fish image classification, Santos et al. found that VGG19 scored 83%, which is 2% higher than VGG16's 81% (Santos et al., 2019).

Research on deep learning and transfer learning approaches is currently focused on fish species classification, as is evident. It has been discovered that fine-tuning previously trained models is a successful strategy for adjusting to novel classication challenges. Classic convolutional neural network model VGG1 6 has been used extensively in several picture classication tasks after achieving exceptional results in the ImageNet competition. Both the hand-crafted features-based approach and the Convolutional Neural Network (CNN) features-based approach are still being developed in fish species classification research. A fish species classification system was created employing a multi-

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 3, **Number** 2, November 2023
https://doi.org/10.47709/brilliance.v3i2.3270

class support vector machine, color, and texture in a hand-crafted features-based method (Hu et al., 2012). The CNN features-based Deepfish framework was created to identify fish in underwater footage (Qin et al., 2016). Additionally, a fish detection system that detects and localizes fish using a combination of long short-term memory and region-based convolutional neural networks (Labao and Naval, 2019). Belaid and Loudini suggest employing deep learning methods to categorize three different forms of brain malignancies (meningioma, glioma, and pituitary tumor) using combinations of pre-trained VGG-16 CNNs. The utilization of original photographs and gray level of co-occurrence matrix (GLCM) features photos as CNN inputs is the focus of this study. The contrast and energy images are the two GLCM features images that are employed. The original image with the energy image as input, according to our tests, has more distinctive features than other input combinations; accuracy can reach an average of 96.5%, which is higher than accuracy in the most advanced classifiers (Belaid and Loudini, 2020). VGG16 is also used in the suggested architecture, although the top layer is absent. The Multilayer Perceptron (MLP) block was added to the VGG16 to replace its top layer. The Regularizes, Dense, and Flatten layers are all present in the MLP block. The softmax activation function is applied to the output of the MLP block. In the MLP block, three regularizations are taken into consideration: dropout, batch normalization, and regularization kernels. The chosen Regularizes are meant to lessen overfitting. The highest accuracy of 18.42% was achieved on the MLP block with Dropout 0.5, exhibiting the best performance. The accuracy was raised by 10.52% and 2.63%, respectively, by the performance of the Batch Normalization and Regularizes kernels (Pardede et al., 2021). Other proposed approach was tested on a dataset of 253 MRI brain pictures (155 of which showed tumors) for the purpose of diagnosing brain malignancies. Younis et al. method may be able to spot brain cancers in MR pictures. The method achieved an excellent accuracy of CNN 96%, VGG 16 98.5%, and Ensemble Model 98.14% in the testing data, outperforming the current conventional approaches for detecting brain cancers (Precision = 96%, 98.15%, 98.41% and F1-score = 91.78%, 92.6%, and 91.29% correspondingly) (Younis et al., 2022). In order to create Multi-Level Residual (MLR), a new residual network strategy, Prasetyo et al. applied Depthwise Separable Convolution to combine low-level data from the first block with high-level features from the last block. Additionally, we suggested MLR-VGGNet, a new CNN design that was strengthened by batch normalization, residual features, MLR, and asymmetric convolution. VGGNet was the model for this CNN architecture. Based on the Fish-gres and Fish4-Knowledge dataset, our experimental results demonstrate that MLR-VGGNet achieved an accuracy of 99.69%, outperforming the original VGGNet by up to 10.33% and other CNN models by up to 5.24% (Prasetyo et al., 2022). Chen et al. suggests a novel framework for the classication of sh species that combines the capabilities of SPPNet (Spatial Pyramid Pooling network), VGG1 6 (Visual Geometry Group 1 6), and FRCNN (Faster Region-based Convolutional Neural Network). Target objects in photos with multiple objects are detected and extracted using FRCNN. Then, images of different species of sh at varying sizes are fed into VGG1 6-SPPNet, which uses transfer learning theory to extract basic features. SPPNet performs pooling operations of various scales in order to further process the input images. Lastly, VGG1 6 recognizes critical characteristics needed for object classification. With an accuracy rate of 0.931 8, which is 26% higher than traditional single VGG1 6 models, the suggested framework outperforms single VGG1 6 models in terms of accuracy, especially when it comes to identifying objects of varied sizes (Chen et al., 2023).

We present a hybrid deep learning method for classifying fish images in this paper. To obtain the feature vector of the images, we employed a pre-trained 16-layer deep model called VGG16. Subsequently, we utilized an combination of distinct stacking models to obtain the desired outcomes. The suggested model was found to perform better than alternative cutting-edge algorithms. A summary of various evaluation metrics for a classification model has been used, like precision, recall, f1-score, and support.

LITERATURE REVIEW

Convolutional Neural Network (CNN)

One kind of deep ANNs used for visual vision analysis is CNNs (ConvNet). Neurons in CNN are responsible for weight, bias, and activation activities. Like other neural networks, CNN is made up of input, hidden, and output layers that perform operations on data to change it in order to look into certain features. Convolution, activation or rectified linear unit (ReLU), and union are the three most used layers (Standford, 2018). Through a sequence of convolutional filters, convolution exposes the input image and activates the corresponding characteristics of the image. ReLU maintains positive values while mapping negative values to zero, allowing for quicker and more efficient training. Because only the activated feature moves on to the next layer, ReLU is also referred to as activation. By using non-linear down sampling to simplify output, pooling lowers the amount of parameters the network has to learn. Over tens or hundreds of layers, each learning layer connected to a distinct set of features repeat these three steps. The number of classes that the network can predict, K, is represented by the dimension K vector created by the following layer, which is a fully linked layer. Each class's probabilities for any classified image are contained in this vector. The final layer of the CNN architecture employs a classification layer, such as sigmoid for output classifications of fewer than or equal to two classes, or softmax for output classifications of more than two classes (Guo and Gao, 2017).

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 3, **Number** 2, November 2023
https://doi.org/10.47709/brilliance.v3i2.3270

VGG-16 Model

The CNN model created by Simonyan and Zisserman served as the foundation for the updated architecture known as the VGG16 model. With 14 million sets of images and 1000 classifications, VGG16 achieved an accuracy score of 92.7%, making it one of the most popular submissions in the ImageNet 2014 competition (Simonyan and Zisserman, 2014). The outcomes show a lot of promise for future development. Deep CNN models, such as the VGG16 model, have been better over time for classification tasks. Moreover, depth was thought to be the primary element enhancing model performance. The model consists of 16 layers, with a 224x224 RGB input layer and only one 3x3 convolutional layer across the whole model. Through the max-pooling layers, the VGG16 reduces the size of the input photos. In addition, there are 138 million parameters in the model. Given that the model's depth is deeper than that of existing CNN models, training such a huge model would need a lot of time (Liu and Deng, 2015).

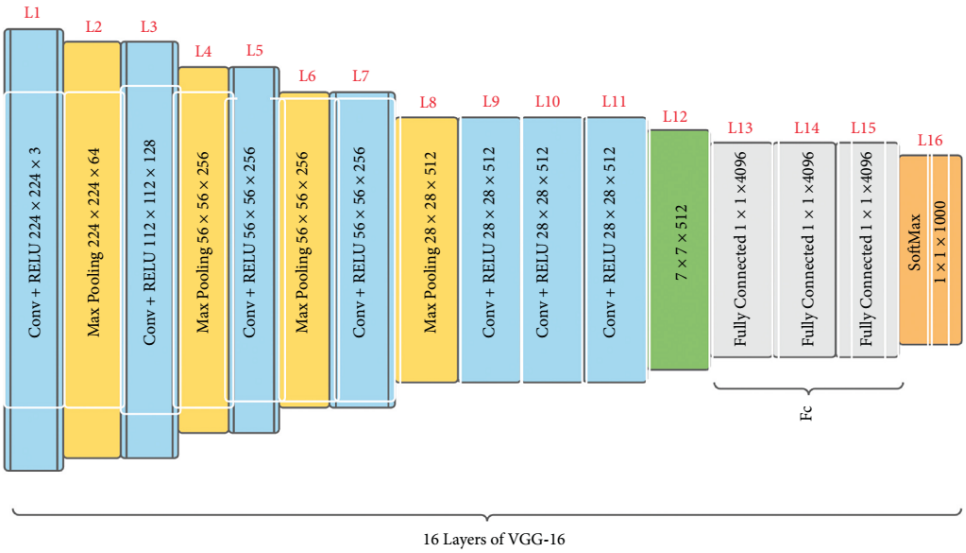

Figure 1. Architecture of VGG16 (Hasan et al., 2021)

The VGG-16 architecture is shown in Figure 1. The five types of layers (conv, ReLU, max-pooling, softmax, and completely linked) comprise all 16 layers of the vgg16 model. A 224 × 224 RGB value with a specified length is used as the source by the convolutional layer. A series of convolutional layers with a 3 × 3 visual field (the smallest size required to preserve the notions of right/left, rise/bottom, and core) convert the data. A max-pooling layer comes after the convolutional layer. ReLU and convolution work together to process the data. In certain configurations, it also has 11 convolutional filters, which are typically thought of as a proportionate alteration of the input networks (nonlinearity coming next). The spatial padding of a source of a convolution layer is defined to be the same as a single pixel for 3 by 3 convolution layers, after convolution, maintaining spatial resolution. Each convolution stride is given as a single pixel. VGG16 focused on having a 3 × 3 filter input layer with a stride of 1 and would always utilize the same padding and max-pool structure instead of having a ton of hyperparameters. Layers of 5 max-pooling, which replicate some of the convolutional layers, are used for spatial pooling (Hasan et al., 2021).

Convolutional Layer

A kernel is a set of discrete values or integers; for each number, the kernel weight is given as a reference. The initial kernel weights for a CNN are a set of integers picked at random. Additionally, the weights are initialized in various ways. In turn, the kernel learns to extract meaningful features because these weights are tweaked during the training process. The convolutional layer is an essential component of CNN's overall structure. It is a set of filters or kernels applied to the data before it is used. Each kernel's width, height, and weight are used to extract characteristics from the input data. Without having to compute the coordinates of the data in that space, the kernel allows them to carry out the operation in a high-dimensional, implicit feature space. Rather, they calculate the inner product of all data linking pictures in feature space (Khousik, 2022). A linear model can be converted into a non-linear model by using the kernel trick. Before the convolutional process starts, the CNN input format is given. The CNN processes an image with several channels, while the standard neural network processes data in a vector format. A grayscale image has just one color "channel," whereas an RGB image has three. One possible configuration for the matrix or kernel to identify picture edges is shown in Figure 2. These matrices or kernel are also called filters because they operate similarly to the traditional filters used in image processing techniques (Fang, 2017).

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 3, **Number** 2, November 2023
https://doi.org/10.47709/brilliance.v3i2.3270

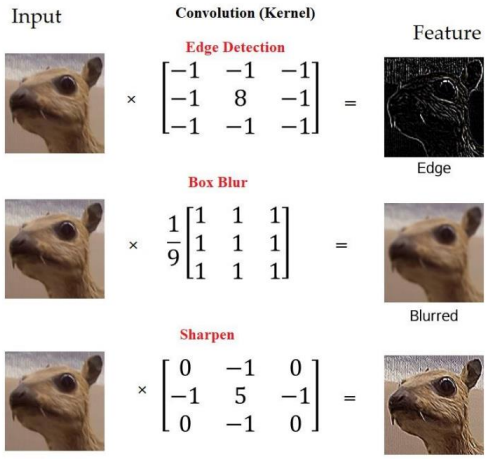

Figure 2. Effects of different convolutional matrices (Taye, 2023)

Pooling Layer

The most crucial data is retained in the feature maps while their dimensionality is reduced by the use of the pooling layer, sometimes referred to as the down-sampling layer. Sliding over the input data in the pooling layer (max, min, avg), a filter applies the pooling operation to the data. Maximum pooling is most commonly used in the literature. Down-sampling is a crucial component of pooling, which is used to lessen the complexity of upper layers. Maybe it's like lowering the resolution in terms of image processing. The pooling has no effect on filter count. Max-pooling is a frequently employed pooling technique. Only the largest number found inside each of the rectangular subregions that make up the image is returned. Among the most common maximum pooling sizes is 2x2. Pooling on the 2-by-2 blocks in the top-left corner causes attention to shift to the top-right corner, moving two steps, as illustrated in Figure 3. Stride 2 is therefore utilized for pooling. It is possible to avoid downsampling by using stride 1, which is not common. Remember that downsampling does not maintain the data's location (Alzubaidi et al., 2021).

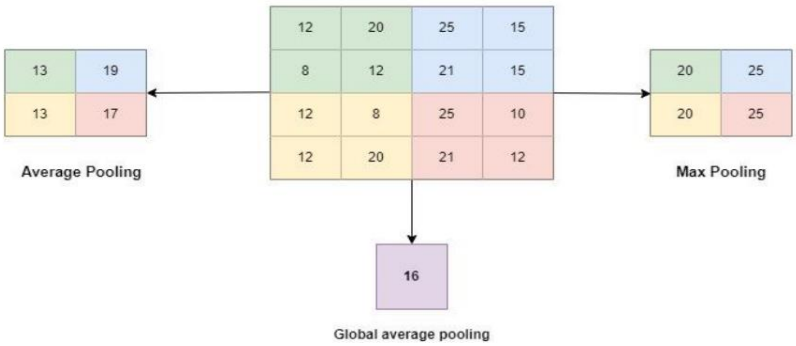

Figure 3. Pooling layer (Taye, 2023)

Non-Linearity (Function of Activation)

The non-linearity layer comes after convolution. The generated output can be altered or stopped thanks to non-linearity. The output can be constrained or oversaturated with this layer. The fundamental task of mapping input to output is performed by all activation function types in all neural network types. The computation of the input value involves summing the weights of the neuron input and any bias, if any. This suggests that by producing the corresponding output, the activation function decides whether or not to fire a neuron in response to a certain input. Non-linear activation layers also referred to as learnable layers, like FC and convolutional layers—are employed in the CNN design subsequent to all weighted layers (Bhatt, 2021). In the context of CNN, the most widely used function is the rectified linear unit (ReLU). The input numbers are all transformed to fall into the positive range. ReLU's main advantage over other algorithms is the amount of time and money it can save. ReLU allows for simplified definitions of functions and gradients. For the positive input, the gradient of the ReLU is constant. The function can be disregarded during implementation even though it cannot be distinguished (Prakash, 2021).

Fully Connected Layer

In the fully-connected layer, neurons are arranged in groups that resemble those found in conventional neural networks. Every node in a layer that is fully linked is consequently directly connected to every other node in the layer above and below, as seen in Figure 4. As can be seen in Figure 4, each node in the most recent frames of the pooling

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 3, **Number** 2, November 2023
https://doi.org/10.47709/brilliance.v3i2.3270

layer is connected as a vector from the fully-connected layer to the top layer. These are the CNN parameters that are used the most inside these layers, however they need a lot of training time (). A fully connected layer's worst flaw is its vast number of parameters, which make training samples need tedious calculation. As such, we make an effort to reduce the quantity of nodes and connections. The dropout technique can satisfy the removed nodes and connections.

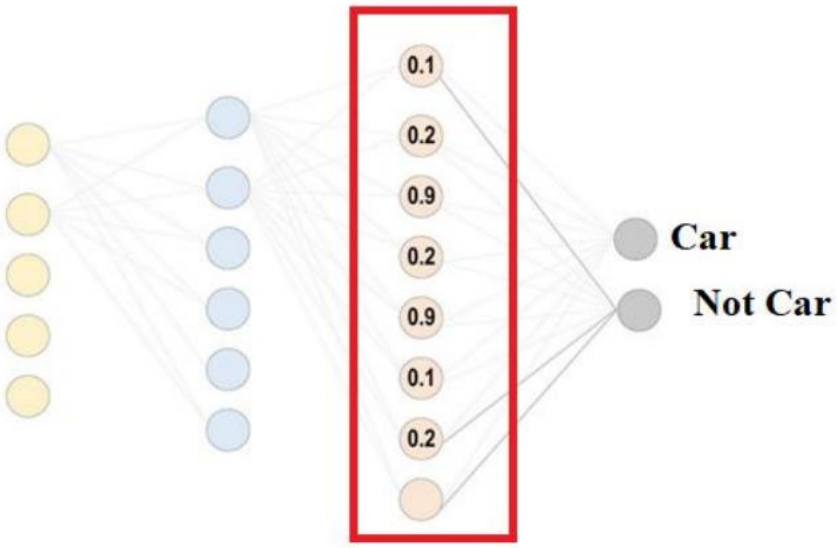

Figure 4. Fully connected layer (Taye, 2023)

Confusion Matrix

A table used in classification to assess a model's performance is called a confusion matrix. It offers a thorough analysis of the model's predictions, indicating the proportion of cases that were classified accurately or inaccurately. A confusion matrix in a binary classification scenario (two classes: positive and negative) looks like in Table 1. True Positive (TP) is the number of instances correctly predicted as positive, for example in a medical diagnosis, it represents the number of patients correctly identified as having a disease. False Negative (FN) is the number of instances that are actually positive but incorrectly predicted as negative, for example in a medical diagnosis, it represents the number of patients with a disease that went undetected. False Positive (FP) is the number of instances that are actually negative but incorrectly predicted as positive, for example in a medical diagnosis, it represents the number of healthy individuals incorrectly identified as having a disease. True Negative (TN) is the number of instances correctly predicted as negative, for example in a medical diagnosis, it represents the number of healthy individuals correctly identified as not having the disease.

Table 1. The Confusion Matrix Table

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Beyond accuracy, the confusion matrix enables a more thorough examination of a model's performance. When working with datasets that are imbalanced that is, when one class substantially outnumbers the other—it is very helpful. Numerous performance metrics can be computed based on these parameters. The precision, recall, F1-score, support, and accuracy are metrics as known as cofusion matrix commonly used to evaluate the performance of a classification model. These metrics are calculated using the equations (1), (2), (3) and (4). Support is the number of actual occurrences of the class in the specified dataset. The accuracy formula is found in equation (1), accuracy is the ratio of correct predictions to the total number of predictions. The precision formula is found in equation (2), Precision is the ratio of true positives to the sum of true positives and false positives. Equation (3) is recall formula, recall is the ratio of true positives to the sum of true positives and false negatives. For F1-score the equation (4) is the harmonic mean of precision and recall (Hasan et. al., 2021).

$$\text{Accuracy} = \frac{True\ Positives + True\ Negatives}{Total\ Samples} \quad (1)$$

$$\text{Precision} = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (2)$$

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 3, **Number** 2, November 2023
https://doi.org/10.47709/brilliance.v3i2.3270

$$\text{Recall} = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (3)$$

$$\text{F1-score} = 2 \times \frac{Precision\ x\ Recall}{Precision + Recall} \quad (4)$$

METHOD

The architecture of the proposed fish classification system is presented in Figure 5. The system uses VGG-16 (Visual Graphics Group), which is one of the Convolutional Neural Network (CNN) architectures. The process begins with an exploratory data analysis (EDA) process to see the distribution of the amount of fish image data owned and present it in a graph. After looking at the data distribution, the VGG-16 architecture was selected as the basic model for transfer learning, using the VGG-16 model architecture, which had been pre-trained on the ImageNet dataset using the Keras deep learning library. Model evaluation was carried out at the final stage of this research to evaluate the performance of the classification model using evaluation metrics such as accuracy and precision for each class, as well as the average of these metrics.

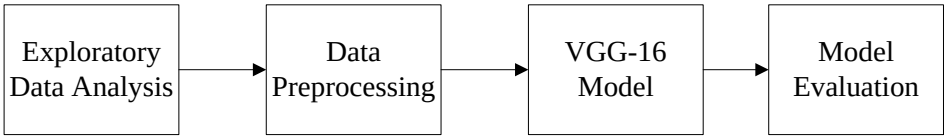

Figure 5. The deep learning method that was used in this research

Exploratory Data Analysis

The initial process carried out in data preparation was the collection of fish image datasets published by ASYU 2020 (The Innovations in Intelligent Systems and Applications Conference ASYU 2020), sourced from https://www.kaggle.com/crowww/a-large-scale-fish-dataset. This dataset contains nine different types of marine fish collected from photos in supermarkets in Izmir, Turkey, for a university-industry collaboration project at the Izmir University of Economics. The dataset includes images of gilthead bream, red sea bream, sea bass, red mullet, horse mackerel, black sea sprat, trout, and shrimp measuring 1024x768 pixels in RGB color format. After the dataset is ready to understand its properties, such as the distribution of various classes of fish species, exploratory data analysis is done. The fish dataset is loaded into a suitable data structure. This could be working with tabular data or a custom structure if dealing with images. An examination is done of the distribution of fish types in the dataset. This involves counting the number of instances (images or samples) belonging to each class. Visualizations are used, such as bar charts or pie charts, to represent the class distribution. To gain a qualitative understanding, randomly sample and visualize images from each class. This can help you observe the visual characteristics of different fish types. The investigation of the resolutions and dimensions of the images in the dataset is done.

Data Preprocessing

One of the most important steps in getting the dataset ready for machine learning model training is data preprocessing. Depending on the characteristics of your data and the needs of our selected model in this case, the VGG-16 the actual preprocessing procedures may change. Every image has been scaled to a standard size. This is crucial since CNNs like VGG-16 frequently need a set input size while working with them. It sizes that work with the VGG-16 architecture have been employed. For instance, input images with 224x224 pixels typically function well with VGG-16. Three sets of the dataset are separated: test, validation, and training. This helps in assessing how well the model performs with unknown data.

VGG-16 Model

Using the VGG-16 architecture for fish classification involves a few key steps: loading the pre-trained VGG-16 model, modifying it to suit the specific task (fish classification), and then training the modified model on your preprocessed fish dataset. The pre-trained VGG-16 model is loaded by importing the VGG-16 model from a deep learning library. Load the pre-trained weights, which are usually trained on a large dataset. Since the top layers of VGG-16 are specific to the ImageNet classification task, modification of the model by adding new layers is needed for the fish classification task. flattening the output from the pre-trained layers and adding new fully connected layers for classification. The modified VGG-16 model on the preprocessed fish dataset must be trained. The training set is used for fitting the model, and the validation set is used to monitor its performance.

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 3, **Number** 2, November 2023
https://doi.org/10.47709/brilliance.v3i2.3270

Model Evaluation

Model evaluation using classification report function that calculates and prints various metrics such as precision, recall, F1-score, and support for each class, as well as macro and weighted averages. The report provides insights into how well the model is performing for each class. The precision is the ability of the classifier not to label as positive a sample that is negative. Recall is the ability of the classifier to find all the positive samples. F1-Score is the harmonic mean of precision and recall. It provides a balance between precision and recall. Support is the number of actual occurrences of the class in the specified dataset. Accuracy is the proportion of true results (both true positives and true negatives) among the total number of cases.

RESULT

Exploratory Data Analysis

After importing several Python libraries that are needed, we initialize an empty list called all_path, where the full paths of images will be stored. Then we make a loop that iterates over the contents of the directory specified by fish_path. os.listdir returns a list containing the names of the entries in the directory. Overall, the code is scanning through a directory (fish_path), excluding certain files, and collecting the full paths of the remaining images in the all_path list. The output in Figure 6 is the result of the code within the loop. It shows that the code is processing different directories containing images of different fish species, and for each directory, it prints the number of files found in that directory. These counts could be useful for understanding the distribution of data or for setting up a deep learning dataset where each class has a balanced number of examples, as shown in Figure 6.

```
found 1000 in Hourse Mackerel
found 1000 in Black Sea Sprat
found 1000 in Sea Bass
found 1000 in Red Mullet
found 1000 in Trout
found 1000 in Striped Red Mullet
found 1000 in Shrimp
found 1000 in Gilt-Head Bream
found 1000 in Red Sea Bream
```

	Filepath	Label
0	/Fish_Dataset/Fish_Dataset/Striped Red Mullet/Striped Red Mullet/00297.png	Striped Red Mullet
1	/Fish_Dataset/Fish_Dataset/Striped Red Mullet/Striped Red Mullet/00931.png	Striped Red Mullet
2	/Fish_Dataset/Fish_Dataset/Black Sea Sprat/Black Sea Sprat/00277.png	Black Sea Sprat
3	/Fish_Dataset/Fish_Dataset/Sea Bass/Sea Bass/00376.png	Sea Bass
4	/Fish_Dataset/Fish_Dataset/Red Mullet/Red Mullet/00804.png	Red Mullet

Figure 6. The number of image files of different fish species found in one directory and the data frame

After getting the number of image files of different fish species found in one directory, the DataFrame is used to inspect the structure and check that the shuffling and indexing operations were successful. The displayed DataFrame will have two columns: 'Filepath' and 'Label'. Each row corresponds to an image, and the 'Label' column contains the class label associated with that image that is shown in Fig 6. Matplotlib and Seaborn libraries created a side-by-side visualization of the class distribution in the dataset. The left subplot is a count plot showing the number of occurrences of each class, which is 1000 images. The right subplot is a pie chart representing the percentage distribution of each class in the dataset, which is 11.1% of each class shown in Figure 7.

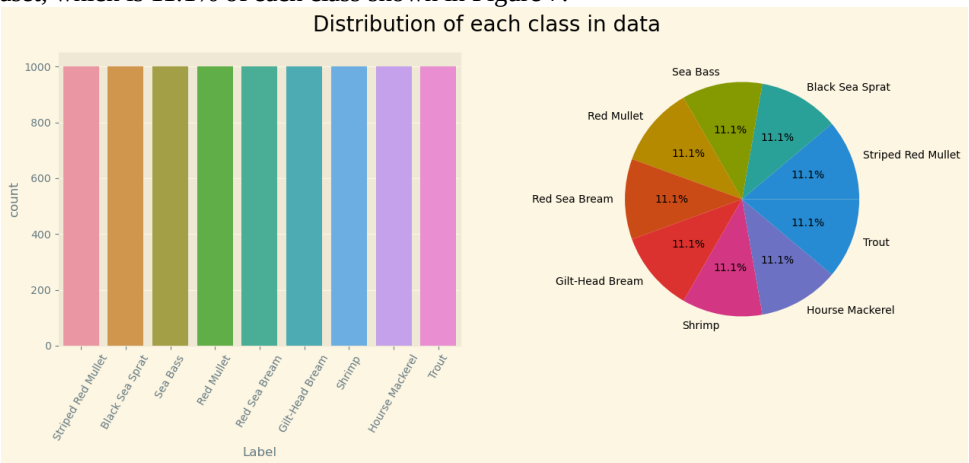

Figure 7. The distribution of each class in fish dataset

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 3, **Number** 2, November 2023
https://doi.org/10.47709/brilliance.v3i2.3270

Data Preprocessing

With the specification that 10% of the data should be used for testing and the remaining 90% for training, the dataframe created in the previous procedure is split into training and testing dataframes. It is best to mix the data before splitting it. To make sure that the training and testing datasets accurately reflect the distribution of data as a whole, shuffling is helpful. Subsequently, it furnishes a random number generator seed. Establishing a seed guarantees repeatability, i.e., you will obtain the same split if you execute the code more than once with the same seed. Following the split, training_df and testing_df are given the resultant data frames. There are 8100 samples in the training data frame (training_df) and 900 samples in the testing data frame (testing_df), with two features each, as shown in Figure 8. This attests to the fact that the data was correctly divided based on the predetermined parameters. A machine learning model is frequently trained on the training set, and its performance on unseen data is assessed on the testing set.

```
The dimension of training data : (8100, 2)
The dimension of testing data : (900, 2)
```

Figure 8. The distribution of each class in fish dataset

The results of using the flow_from_dataframe method with the image data generators for training, validation, and testing sets are shown in Figure 9, which is presented in three lines of statement. The first statement of the image data generator for the training set (training_images) found 6480 validated image filenames; these images belong to 9 classes, suggesting that the dataset has 9 different categories or labels. The same result for the second line indicates that the image data generator for the validation set (validation_images) found 1620 validated image filenames. Similar to the training set, these images also belong to nine classes. The last line indicates that the image data generator for the testing set (testing_images) found 900 validated image filenames. Like the training and validation sets, these images belong to nine classes. The statement "validated" in the result indicates that the generator has successfully processed and validated these images. These statements are informative and reassure that the data generators have successfully identified and processed the specified number of images for each dataset split (training, validation, and testing) and that the images belong to the expected number of classes. These generators are now ready to be used for training and evaluating a deep learning model.

```
Found 6480 validated image filenames belonging to 9 classes.
Found 1620 validated image filenames belonging to 9 classes.
Found 900 validated image filenames belonging to 9 classes.
```

Figure 9. The results of using image data generators for training, validation, and testing sets

VGG-16 Model

Using TensorFlow's Keras API, the VGG16 architecture is pre-trained and then set to non-trainable. The input photos' form is set to (224, 224, 3), a standard RGB image size. The VGG16 model's completely connected layers, or top layers, which handle categorization, ought to be disregarded. This is frequently done when more custom categorization layers are added later on, or when the model is utilized as a feature extractor. The ImageNet dataset's pre-trained weights should be used to initialize the model. A sizable dataset called ImageNet is employed in picture classification. The final convolutional layer should be applied before applying global average pooling. During the training of neural networks, "epochs" and "batch size" are two key parameters that influence how the model learns from the training data. The training of our neural network takes 7 epochs, meaning it will see the entire dataset seven times. In each epoch, the dataset will be divided into batches of 32 examples, and the model's weights will be updated after processing each batch. For building and training a neural network using the Keras API with a pretrained model as a feature extractor. The dense layers are fully connected layers with ReLU activation functions. Dropout is applied to the second dense layer, which helps prevent overfitting by randomly dropping a fraction of the input units during training. Batch normalization is applied to normalize the activations of the previous layer, which can help improve training stability.

The final layer is a dense layer with a softmax activation function. Assuming this is a classification task with nine classes, the model will output probabilities for each class. The Keras model is used to connect the input and output layers. During training, the model compiles, specifying the optimizer, loss function, and metrics. Figure 10 displays the results of a neural network's seven epochs of training. In general, the training procedure involves changing the model's weights to minimize the training loss while iterating over the dataset for a predetermined number of epochsIn this case, the learning rate did not change. For reference, the training time for each epoch is also shown, the detail information about the results of a neural network's seven epochs of training provided in Table 2.

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 3, **Number** 2, November 2023
https://doi.org/10.47709/brilliance.v3i2.3270

Table 2. Detailed information results of a neural network's seven epochs of training

Epoch	Training Metrics		Validation Metrics		Additional Information
	Loss	Accuracy	Loss	Accuracy	
1/7	0.8630	71.47%	0.5451	89.81%	• Training took 144 seconds. • A model checkpoint was triggered, and the best model was saved to ./bestmodel.h5. • The learning rate (lr) was 0.0010.
2/7	0.1803	94.63%	0.3129	96.23%	• Training took 86 seconds. • Another model checkpoint was triggered, and the best model was saved again. • The learning rate remained at 0.0010.
3/7	0.1254	96.22%	0.2952	96.73%	• Training took 86 seconds. • Another model checkpoint was triggered, and the best model was saved again. • The learning rate remained at 0.0010.
4/7	0.0717	97.72%	0.4012	97.10%	• Training took 89 seconds. • Another model checkpoint was triggered, and the best model was saved again. • The learning rate remained at 0.0010.
5/7	0.0550	98.16%	0.2511	98.33%	• Training took 85 seconds. • Another model checkpoint was triggered, and the best model was saved again. • The learning rate remained at 0.0010.
6/7	0.0470	98.32%	0.3192	97.10%	• Training took 85 seconds. • Another model checkpoint was triggered, and the best model was saved again. • The learning rate remained at 0.0010.
7/7	0.0359	98.89%	0.2330	97.72%	• Training took 86 seconds. • Another model checkpoint was triggered, and the best model was saved again. • The learning rate remained at 0.0010.

```
Epoch 1/7
203/203 [==============================] - ETA: 0s - loss: 0.8630 - accu
racy: 0.7147
Epoch 1: val_accuracy improved from -inf to 0.89815, saving model to ./b
estmodel.h5
203/203 [==============================] - 144s 663ms/step - loss: 0.863
0 - accuracy: 0.7147 - val_loss: 0.5451 - val_accuracy: 0.8981 - lr: 0.0
010
Epoch 2/7
203/203 [==============================] - ETA: 0s - loss: 0.1803 - accu
racy: 0.9463
Epoch 2: val_accuracy improved from 0.89815 to 0.96235, saving model to
./bestmodel.h5
203/203 [==============================] - 86s 421ms/step - loss: 0.1803
- accuracy: 0.9463 - val_loss: 0.3129 - val_accuracy: 0.9623 - lr: 0.001
0
Epoch 3/7
203/203 [==============================] - ETA: 0s - loss: 0.1254 - accu
racy: 0.9622
Epoch 3: val_accuracy improved from 0.96235 to 0.96728, saving model to
./bestmodel.h5
203/203 [==============================] - 86s 422ms/step - loss: 0.1254
- accuracy: 0.9622 - val_loss: 0.2952 - val_accuracy: 0.9673 - lr: 0.001
0
Epoch 4/7
203/203 [==============================] - ETA: 0s - loss: 0.0717 - accu
racy: 0.9772
Epoch 4: val_accuracy improved from 0.96728 to 0.97099, saving model to
./bestmodel.h5
203/203 [==============================] - 89s 440ms/step - loss: 0.0717
- accuracy: 0.9772 - val_loss: 0.4012 - val_accuracy: 0.9710 - lr: 0.001
0
Epoch 5/7
203/203 [==============================] - ETA: 0s - loss: 0.0550 - accu
racy: 0.9816
Epoch 5: val_accuracy improved from 0.97099 to 0.98333, saving model to
./bestmodel.h5
203/203 [==============================] - 85s 420ms/step - loss: 0.0550
- accuracy: 0.9816 - val_loss: 0.2511 - val_accuracy: 0.9833 - lr: 0.001
0
Epoch 6/7
203/203 [==============================] - ETA: 0s - loss: 0.0470 - accu
racy: 0.9832
Epoch 6: val_accuracy did not improve from 0.98333
203/203 [==============================] - 85s 421ms/step - loss: 0.0470
- accuracy: 0.9832 - val_loss: 0.3192 - val_accuracy: 0.9710 - lr: 0.001
0
Epoch 7/7
203/203 [==============================] - ETA: 0s - loss: 0.0359 - accu
racy: 0.9889
Epoch 7: val_accuracy did not improve from 0.98333
203/203 [==============================] - 86s 422ms/step - loss: 0.0359
- accuracy: 0.9889 - val_loss: 0.2330 - val_accuracy: 0.9772 - lr: 0.001
0
```

Figure 10. The results of a neural network's seven epochs of training

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 3, **Number** 2, November 2023
https://doi.org/10.47709/brilliance.v3i2.3270

The output architecture of a neural network model appears to be using a pretrained convolutional neural network (CNN) as a feature extractor, followed by additional dense layers for classification as shown in Figure 11. Global average pooling is applied to reduce the spatial dimensions before the dense layers. Dropout and batch normalization are used for regularization and normalization, respectively. The final dense layer with softmax activation is likely for classification output. The model has a total of over 14 million parameters, but only around 91,000 of them are trainable, indicating the use of transfer learning with a pretrained model.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 global_average_pooling2d (G (None, 512)               0
 lobalAveragePooling2D)

 dense (Dense)               (None, 128)               65664

 dense_1 (Dense)             (None, 128)               16512

 dropout (Dropout)           (None, 128)               0

 batch_normalization (BatchN (None, 128)               512
 ormalization)

 dense_2 (Dense)             (None, 64)                8256

 dense_3 (Dense)             (None, 9)                 585

=================================================================
Total params: 14,806,217
Trainable params: 91,273
Non-trainable params: 14,714,944
_________________________________________________________________
```

Figure 11. A summary output of the architecture of a neural network model

Model Evaluation

A machine learning model's accuracy and validation accuracy in training epochs is shown on the graph in Figure 12. The accuracy or validation accuracy is displayed on the y-axis, while the epoch number is displayed on the x-axis. We can observe in Figure 12 that as training progresses, both validation accuracy and accuracy increase. This comes as a result of the model learning from additional data to provide predictions that are more accurate. But the accuracy always exceeds the validation accuracy. Compared to validation data, the model is more likely to overfit to the training set. It is also seen in Figure 12 that after epoch 3, the validation accuracy begins to plateau. This may indicate that the training data is being overfitted by the model. It's crucial to stop training the model and assess it on a test set if it's overfitting. Overall, the graph indicates that as training progresses, the model is improving its prediction skills. To prevent overfitting, it's crucial to keep an eye on the validation accuracy.

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 3, **Number** 2, November 2023
https://doi.org/10.47709/brilliance.v3i2.3270

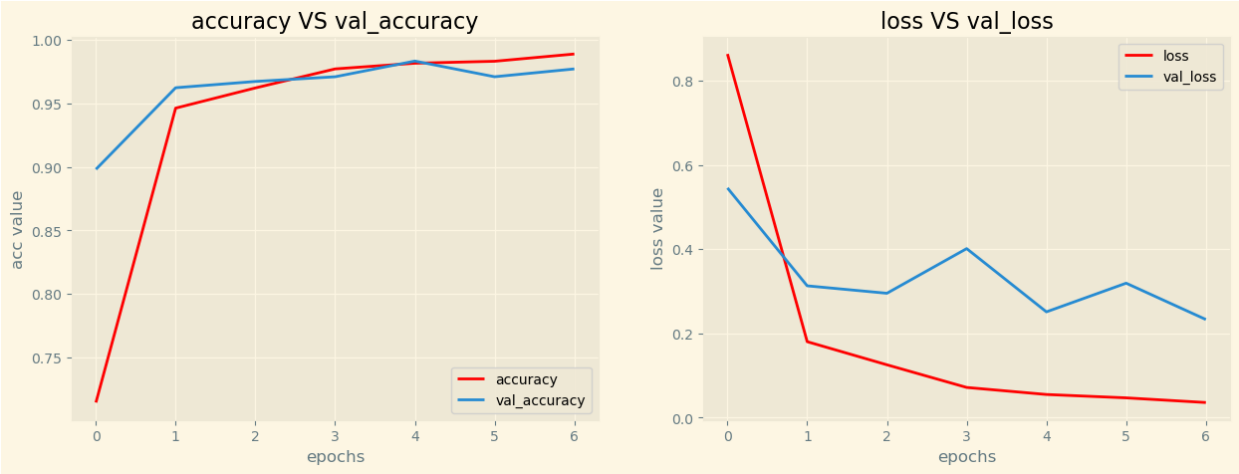

Fig. 12. Accuracy and l**oss** graph of the VGG 16

DISCUSSION

Figure 13 shows the precision, recall, and F1-score of a fishing rod for detecting and classifying different types of fish. For every species of fish, the figure demonstrates that the fishing rod has extremely excellent memory, accuracy, and F1-scores. Every fish kind has at least 0.97 accuracy and 0.96 recall. Every kind of fish has an F1-score of at least 0.98. All things considered, the figure demonstrates how well the fishing rod can identify and categorize various fish species. The fishing pole works especially well for identifying and locating trout, shrimp, and striped red mullet. These fish species have F1-scores of 1.00, recall, and accuracy. Red mullet is harder to find and categorize using a fishing line. Compared to the other fish species, red mullet has a somewhat lower accuracy of 0.896. Red Mullet's recall and F1 scores are still quite high, though.

	precision	recall	f1-score	support
Black Sea Sprat	0.97	0.99	0.98	94
Gilt-Head Bream	0.98	0.98	0.98	113
Hourse Mackerel	0.99	0.98	0.99	111
Red Mullet	0.96	1.00	0.98	105
Red Sea Bream	0.98	1.00	0.99	95
Sea Bass	1.00	0.96	0.98	102
Shrimp	1.00	0.98	0.99	87
Striped Red Mullet	0.99	0.98	0.98	99
Trout	1.00	1.00	1.00	94
accuracy			0.99	900
macro avg	0.99	0.99	0.99	900
weighted avg	0.99	0.99	0.99	900

Figure 13. Performance metrics from a classification model evaluation

The VGG16 model outperformed usual objectives in fish classification with remarkable accuracy. This precision translates into an enhanced capacity to distinguish between different fish species, which is essential for managing aquaculture, sustainable fishing methods, and ecological monitoring. The robustness and generalizability of the model are demonstrated by its consistent performance across all fish species. This implies that the model can evaluate different fish populations and is not prone to overfitting on certain species. Because VGG16 is a proven and computationally efficient design, it may be used in resource-constrained real-world applications.

CONCLUSION

With the VGG16 model, fish picture categorization achieved a high overall accuracy of 99%, which is undoubtedly a good result. This shows that for the great majority of the dataset's cases, the model accurately predicted the class. To get a complete picture of the model's performance, it is necessary to take into account additional assessment metrics in addition to accuracy. The overall accuracy is impressive, but it's beneficial to examine class-specific metrics such as precision, recall, and F1-score. This can highlight how well the model performs on individual classes. While a

E-ISSN : 2807-9035
Volume 3, **Number** 2, November 2023
https://doi.org/10.47709/brilliance.v3i2.3270

99% overall accuracy is promising, it's recommended to delve deeper into class-specific metrics and potential imbalances to ensure that the model performs well across all classes in the fish image classification task. Contextual understanding of the application domain and a thorough analysis of evaluation metrics will contribute to a more comprehensive assessment of the model's effectiveness.

REFERENCES

Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaría, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. (2021). Review of Deep Learning: Concepts, Cnn Architectures, Challenges, Applications, Future Directions. Journal of Big Data Vol. 8 Number 53.

Belaid, O.N.; Loudini, M., (2020). Classication of Brain Tumor by Combination of Pre-Trained Vgg1 6 CNN. Journal of Information Technology Management2020, 12, 1 3-25.

Bhatt, D.; Patel, C.; Talsania, H.; Patel, J.; Vaghela, R.; Pandya, S.; Modi, K.; Ghayvat, H. (2021) CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope. Electronics, 10, 2470.

Chen, M., Lai, T., Chen, Y., Chou, T., & Ning, F. (2023). A Robust Fish Species Classification Framework: FRCNN-VGG16-SPPNET. https://doi.org/10.21203/rs.3.rs-2825927/v1

Guo, P. and Gao Q., (2017). A multi-organ plant identification method using convolutional neural networks, in Proceedings of 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2017, pp. 371-376.

Hasan, Mohammed & Fatemi, Mohammed & Khan, Mohammad & Kaur, Manjit & Zaguia, Atef. (2021). Comparative Analysis of Skin Cancer (Benign vs. Malignant) Detection Using Convolutional Neural Networks. Journal of Healthcare Engineering. 2021. 1-17. 10.1155/2021/5895156.

He, K., Zhang, X., Ren, S. and Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770-778. https://doi.org/10.1109/CVPR.2016.90

Hu, J., Li, D., Duan, Q., Han, Y., Chen, G., Si, X., (2012). Fish species classification by color, texture and multi-class support vector machine using computer vision. Comput. Electron. Agric. 88, 133–140. https://doi.org/10.1016/j.compag.2012.07.008.

Khalifa, N. E. M., Taha, M. H. N. and Hassanien A. E., (2019). Aquarium Family Fish Species Identification System Using Deep Neural Networks, Advances in Intelligent Systems and Computing Proceedings of the International Conference on Advanced Intelligent.

Koushik, J. (2016). Understanding Convolutional Neural Networks. May 2016. Available online: http://arxiv.org/abs/1605.09081

Kratzert, F. and Mader, H., (2018) Fish species classification in underwater video monitoring using Convolutional Neural Networks, doi: 10.31223/osf.io/dxwtz.

Labao, A.B., Naval, P.C., (2019). Cascaded Deep Network Systems With Linked Ensemble Components For Underwater Fish Detection In The Wild. Ecol. Inf. 52, 103–121. https://doi.org/10.1016/j.ecoinf.2019.05.004.

Liu, S. and Deng, W., (2015). Very deep convolutional neural network based image classification using small training sample size, 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur, pp.730-734.

Mingwang, L., (2017). Fish Image Recognition and Separation Based on Convolutional Neural Network. Image and Multimedia Technology, Vol. 3, pp. 82-83.

Pardede, J.; Sitohang, B.; Akbar, S.; Khodra, M.L., (2021) Implementation of transfer learning using VGG1 6 on fruit ripeness detection. Int. J. Intell. Syst. Appl 2021, 13, 52-61.

Prakash, K.B.; Kannan, R.; Alexander, S.A.; Kanagachidambaresan, G.R. (2021). Advanced Deep Learning for Engineers and Scientists: A Practical Approach; Springer: Berlin/Heidelberg, Germany.

Prasetyo, E., Suciati, N., Fatichah, C., (2022). Multi-level residual network VGGNet for fish species classification, Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 8, Part A, Pages 5286-5295, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2021.05.015.

Qin, H., Li, X., Liang, J., Peng, Y., Zhang, C., (2016). DeepFish: Accurate underwater live fish recognition with a deep architecture. Neurocomputing 187, 49–58. https://doi.org/10.1016/J.NEUCOM.2015.10.122.

Rathi, D., Jain, S. and Indu, S., Underwater Fish Species Classification using Convolutional Neural Network and Deep Learning, (2017). Ninth International Conference on Advances in Pattern Recognition (ICAPR)Bangalore, pp.1-6. doi: 10.1109/ICAPR.2017.8593044.

Santos, A. A. D. and Gonçalves, W. N., (2019). Improving Pantanal Fish Species Recognition Through Taxonomic Ranks in Convolutional Neural Networks, Ecological Informatics, vol. 53, p. 100977, doi: 10.1016/j.ecoinf.2019.100977

Simonyan, K. and Zisserman A., (2014). Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556.

Brilliance
Research of Artificial Intelligence

E-ISSN : 2807-9035
Volume 3, **Number** 2, November 2023
https://doi.org/10.47709/brilliance.v3i2.3270

Standford University, (2018). Introduction to Convolutional Neural Networks, https://web.stanford.edu/class/cs231a/lectures/intro_cnn.pdf.

Taye MM. (2023). Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions. Computation. 11(3):52. https://doi.org/10.3390/computation11030052.

Younis, A.; Qiang, L.; Nyatega, C.O.; Adamu, M.J.; Kawuwa, H.B. (2022). Brain Tumor Analysis Using Deep Learning and Vgg-1 6 Ensembling Learning Approaches. Applied Sciences 2022, 12, 7282.