Journal of King Saud University – Computer and Information Sciences 34 (2022) 5286–5295

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Multi-level residual network VGGNet for fish species classification

Eko Prasetyo [a,b], Nanik Suciati [a,*], Chastine Fatichah [a]

[a] Department of Informatics, Faculty of Intelligent Electrical and Informatics Technology, Institut Teknologi Sepuluh Nopember, Jl. Raya ITS, Surabaya 60111, Indonesia
[b] Department of Informatics, Faculty of Engineering, Universitas Bhayangkara Surabaya, Jl. Ahmad Yani 114, Surabaya 60231, Indonesia

ARTICLE INFO

Article history:
Received 31 December 2020
Revised 4 May 2021
Accepted 29 May 2021
Available online 5 June 2021

Keywords:
Multi-level residual
Low level feature
Convolutional neural network
Asymmetric convolution
Fish species classification
VGGNet

ABSTRACT

The development of an image-based fish classification system using Convolutional Neural Network (CNN) has the advantages of no longer directly conducting features extraction and several features analysis. These steps has been involved by cascading convolution from initial to final block, where the initial, middle, and final blocks produce low-, middle-, and high-level features, respectively. Due to cascading convolution, CNN produces only high-level features. However, fish classification requires not only high-level features but also low-level features such as points, lines, and textures for representing edge spines, gill covers, fins, and skin textures in order to achieve higher performance; furthermore, CNN generally has not yet incorporated low-level features in the last block. In this paper, we proposed Multi-Level Residual (MLR) as a new residual network strategy by combining low-level features of the initial block with high-level features of the last block by applying Depthwise Separable Convolution. We also proposed MLR-VGGNet as a new CNN architecture inherited from VGGNet and strengthened it using Asymmetric Convolution, MLR, Batch Normalization, and Residual features. Our experimental results show that MLR-VGGNet achieved an accuracy of 99.69%, outperformed original VGGNet relative up to 10.33% and other CNN models relative up to 5.24% on Fish-gres and Fish4-Knowledge dataset.

1. Introduction

Fish species manual classification is challenging, time-consuming, and requires experience, especially when encountering similar fish species. We can recognize the differences by more closely observing body shape, color, and skin contours. Developing an automatic fish classification system on an image with various backgrounds and lighting conditions is challenging. Hence, several earlier works have been done in developing an image-based application for fish species classification. For example, a system to recognize fish species living in an open underwater environment (Qin et al., 2016), fish detection and counting on a video captured from the sea (Labao and Naval, 2019), a system to detect and classify fish species to monitor changes in the relative abundance of fish populations at sea (Jalal et al., 2020). In the automatic image-based fish species classification system using a hand-crafted features-based approach (Bermejo, 2007; Hu et al., 2012; Qin et al., 2016; Tharwat et al., 2018), it achieves optimal performance by combining low-level features (presence of minor objects), such as points, edges, textures, for representing edge spines, gill covers, fins, and skin textures; and high-level features (presence of the major object), such as heads, tails and other body parts. Therefore, the development of a classification system by combining low- and high-level features promises higher performance. Nevertheless, a hand-crafted features-based approach requires high effort related to data preprocessing such as anomaly detection, normalization, incorrect data treatment, and feature selection.

Another one is non-hand-crafted features-based approach using Convolutional Neural Network (CNN), where it has the advantages of no longer explicitly conducting feature extraction and several features analysis. These steps has been included in the cascading convolution from initial block to final block using particular convolution strategies. The initial, middle, and final blocks produce low, middle, and high-level features, respectively. The research related to this approach, for example, Deepfish framework to recognize fish from underwater video captured in the ocean observation network (Qin et al., 2016); the system was trained on Fish4-Knowledge dataset and achieved an accuracy of 98.64%. A fish detection system by combining Region-based Convolutional

* Corresponding author.
E-mail addresses: eko@ubhara.ac.id (E. Prasetyo), nanik@if.its.ac.id (N. Suciati), chastine@if.its.ac.id (C. Fatichah).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2021.05.015
1319-1578/

Neural Networks and Long Short-Term Memory to detect and localize fish from a dataset of 18 videos taken in the wild and achieved detection performance (Labao and Naval, 2019). CNN is also used to solve common problems such as soft drink type classification (Hafiz et al., 2020), human action recognition (Jaouedi et al., 2020), and aging classification (Boussaad and Boucetta, 2020). CNN has also effectively solved various problems using a transfer learning technique, such as the classification of materials using a crystal graph convolutional neural network (CGCNN) to enhance classification performance (Lee and Asahi, 2021), classification on magnetic resonance images using LeNet-like model (Aderghal et al., 2020), geochemical anomaly detection (Li et al., 2020), and also. crack detection of civil infrastructure (Yang et al., 2020).

Combining CNN with other approaches such as heuristics can also strengthen systems to solve problems, including the combination of CNN and Genetic Algorithm (GA) to find the best architectural combination for brain tumor classification (Kabir Anaraki et al., 2019), using GA to analyze the microscopy images dataset for training data of CNN (Połap, 2020), using GA to determine the best-combined weight of the three CNN models (Ayan et al., 2020), and also GA for initializing CNN weights (Ijjina and Chalavadi, 2016). On the other hand, these hybrid approaches increase performance but using the more intricate model and expensive computation, which is not ideal for applications on devices with limited storage space, such as mobile devices. We require certain models that comparable results to other models while using smaller model sizes and computations.

Several researchers have developed many CNN architectures, for example, VGGNet. (Simonyan and Zisserman, 2015), ResNet (He et al., 2016), Inception V3 (Szegedy et al., 2016), DenseNet (Huang et al., 2017), Xception (Chollet, 2017), and MobileNet (Howard et al., 2017). Previous studies on CNN established convolutional concepts aimed to achieve better performance and/or simplifying architecture while retaining performance. VGGNet simplifies convolutional architecture by using smaller kernels (Simonyan and Zisserman, 2015), ResNet uses the residual concept (skip connection) to maintain low-level features that may be lost due to higher-level convolution (He et al., 2016), Densenet also enhances the concept of residue using concatenation and convolution (Huang et al., 2017). However, the residues sent, both ResNet and Densenet, hold only within the same block, so the low-level features of the initial block have not been retained until the last block. This means that the features produced at the end of the convolution are typically high-level features as well. In addition, VGGNet has succeeded in simplifying the AlexNet architecture by using smaller convolutional kernel sizes, but VGGNet still uses a vast number of parameters and its performance has been outperformed by newer models. Reducing the number of parameters by removing the fifth block is beneficial since newer CNN models usually use a more significant number of parameters to produce better performance.

As discussed earlier, CNN conducted cascading convolution; consequently, it produces only high-level features in the final block and leaves low- and middle-features from the earlier block. However, fish classification requires both a low-level and a high-level feature map to achieve optimal classification performance. On the other hand, the current CNN has not yet combined low- and high-level features. Hence, we intentionally assemble low- and high-level features to overcome this issue by proposing Multi-Level Residual VGGNet (MLR-VGGNet) with the following details:

- Proposing Multi-Level Residual (MLR) as a new residual network strategy

MLR combines low-level features of the initial block with high-level features of the end block by applying DSC computation. At the end of each block where the feature map will be retained (low-level features), a branch is added using MLR to forward the low-level features to the upper convolutional layer. DSC will project the low-level feature map using one convolution to adjust the feature map size with the upper feature map size.

- Using pretrained VGGNet as backbone with parameter reduction

As indicated earlier, VGGNet has a severe problem that it has a vast number of parameters. We use VGGNet as a backbone in our proposed CNN architecture without a fifth block and replace it with asymmetric convolution (AC). AC is a convolution consisting of two sequence layers with a 3x1 and 1x3 filters for decreasing number of parameter (Szegedy et al., 2016). Usually, the number of parameters in the final block is vaster than the initial block, so it is possible to reduce the number of parameters by replacing the fifth block with AC. We use pre-trained VGGNet as the backbone, so we no longer need to train CNN using imagenet anymore. We retrain the additional new convolution layers and freeze the remaining layers to speed up the training process and achieve optimal performance.

- Strengthening the asymmetric convolution (AC) using batch normalization (BN) and residual mechanism

BN can obtain a stable distribution of the activation value and shorten the epoch training (Amin et al., 2020; Wang et al., 2019) by calculating normalization with scaling and shifting (Ioffe and Szegedy, 2015), while Residual Network (ResNet) conveys the concept of residual features (skip connection) to keep the lower layer map features (He et al., 2016). We maintain each AC layer's features map by sending it to the next layer and add BN to get stable output distribution. This residual embedding does not require additional parameters to the CNN model so that it does not increase the model size.

We compared our proposed architecture's performance with state-of-the-art such as original VGG16, original VGG19, ResNet50, Inception V3, and Xception using Fish-gres (3248 images and eight fish species) and Fish4-Knowledge (F4K) dataset (27,320 images and 23 fish species). The experimental results show that our proposal achieves better performance compared to state-of-the-art using pre-trained CNN models. Hence, our model resolves the case of fish species classification.

The rest of this paper is organized as follows. Section 2 discusses related work to fish species classification and VGGNet. Section 3 describes our proposal and some other components. Section 4 explains our experiment result and discussion of performance comparison with several CNN models. Finally, section 5 summarizes the conclusion of this research.

2. Related work

2.1. Fish species classification

Research in fish species classification is still overgrowing both hand-crafted features-based approach and Convolutional Neural Network (CNN) features-based approach. In hand-crafted features-based approach, a fish species classification system was developed using color and texture and a multi-class support vector machine (Hu et al., 2012); by assembling feature extraction, feature reduction, and classification (Tharwat et al., 2018); and also fish age classification based on the combination of morphological features and its biometrics (Bermejo, 2007). In the CNN features-based approach, Deepfish framework was developed to recognize

fish from underwater video (Qin et al., 2016). And also a fish detection system by combining Region-based Convolutional Neural Networks and Long Short-Term Memory to detect and fish localization (Labao and Naval, 2019).

Most studies in the field of fish species classification have only focused on applying or combining certain features and classifiers to achieve optimal performance. Neither the conventional approach nor the CNN-based approach, no research uses low-level and high-level features as the basis for classifying fish species classification. The use of these two features promises optimal performance.

2.2. VGGNet

VGGNet was introduced in the challenge of the 2014 ILSVRC (ImageNet Large Scale Visual Recognition Competition) in the classification task (Simonyan and Zisserman, 2015) and had significant improvements over ZFNet and AlexNet (Krizhevsky et al., 2012). Since then, VGGNet has been applied in many classification cases (Rangarajan et al., 2018; Shallu and Mehra, 2018). VGGNet was also compared with several fine-tuned pre-trained CNN architectures, such as Inception-V3, VGG16, and ResNet50; the VGG16 achieved the highest accuracy (Rodrigues et al., 2019). The other researcher also found the best average performance was achieved by Inception V3, ResNet50, GoogleNet, VGG16, AlexNet, and VGG19 (Lumini and Nanni, 2019).

Recent research has suggested that different architectural models provide different performance results in different cases. This performance is also obtained from the architectural parameters of each CNN. Especially for VGGNet, it has a severe problem since it has a vast number of parameters. We propose an architecture that inherits VGGNet with fewer parameters and higher performance.

3. Proposed methods

3.1. VGGNet as the backbone of the architecture

The VGGNet consists of five blocks, in which each block consists of a convolutional layer followed by max-pooling. The most popular architectures of VGGNet are VGG16 (13 convolutional layers and three fully connected layers) and VGG19 (16 convolution layers and three fully connected layers). For example, the convolution part of the VGG16 architecture is shown in Fig. 1. The convolution part of VGG16 consists of five blocks, where the number of layers in the first until the fifth block are two, two, three, three, and three layers, respectively. The feature maps for each block are 64, 128, 256, 512, and 512 feature maps, respectively. In addition, the number of parameters for each block is 38.72 thousand, 221.44 thousand, 1.48 million, 5.9 million, and 7.08 million, respectively. The total of parameters is 14.71 million. VGG19 has a similar architecture, consisting of 16 convolutional layers and three fully connected layers with 20.02 million parameters.

After winning the ILSVRC, VGGNet has become a prevalent model and has been developed by numerous researchers for various purposes, including classification and object detection. However, VGGNet performance was already outperforming other newer models, therefore in this research, we improved VGGNet with fewer parameters and achieved superior performance than newer models. Reducing the number of parameters is beneficial since newer CNN models usually use a more significant number of parameters to produce better results. Hence, to diminish the number of parameters, the best approach is eliminating the final convolution block (the fifth block) since this block generates high-level features while keeping the previous block. We also use pre-trained VGGNet as initial weights to speed up training computation and inherit low-level features that have been created during training with the imagenet dataset. Meanwhile, we freeze the four blocks during training to keep the weights and preserve the layers' generalization.

3.2. Asymmetric convolution

Asymmetric Convolution (AC), one method of Factorizing Convolution, aims to reduce the number of connections and/or parameters of CNN without reducing recognition performance (Szegedy et al., 2016, 2015). In AC, a mask filter $n \times n$ of a convolutional layer is split into two layers with two sequence asymmetric filters, one filter of $n \times 1$ and $1 \times n$. The mathematical expression as follows:

$$y = \mathcal{F}(\{W_i, F\}) = \sum_{i=-M}^{M} \sum_{j=-N}^{N} W(i,j)F(x-i, y-j)$$
$$= \sum_{i=-M}^{M} W_x(i) \left[\sum_{j=-N}^{N} W_y(j)F(x-i, y-j) \right] \quad (1)$$

where W is a 2D-kernel filter, M and N are the row and column of W, F is 2D-image, W_x is a 1D-kernel along x-dimension, and W_y is a 1D-kernel along y-dimension. The number of parameters for a convolution layer is calculated as follows:

$$P_k = (M.N.f_{k-1} + 1).f_k \quad (2)$$

where P_k is the number of parameters for k-layer, f_{k-1} is the number of feature maps of k-1 layer, and f_k is the number of feature maps of k layer. For example, a convolutional layer with kernel size 3×3, input 32 feature map, and output 32 feature map, then the number

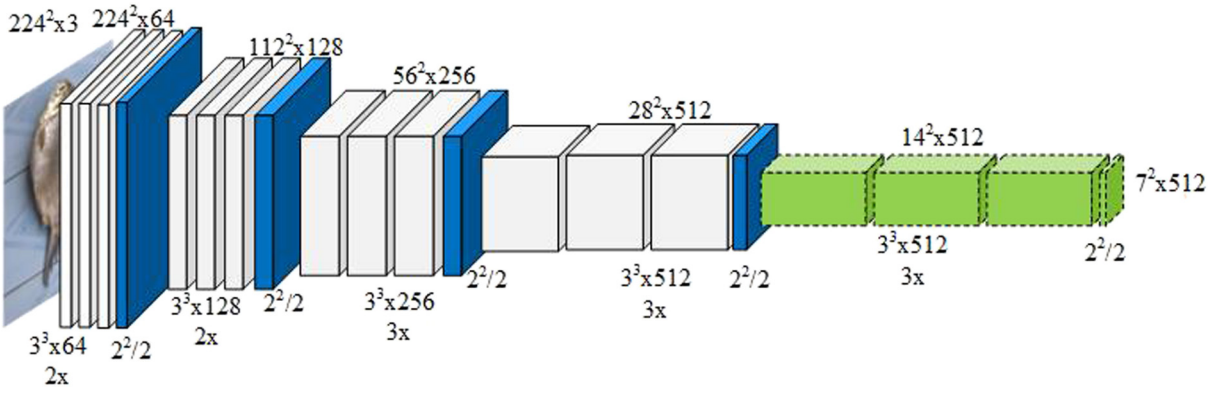

Fig. 1. Architecture of VGG16.

network connection or parameters are $(3 \times 3 \times 32 + 1) \times 32 = 9.248$ parameters. The AC reduces the number of parameters by filter 3×1 with $(3 \times 1 \times 32 + 1) \times 32 = 3.104$ parameters followed by filter 1x3 with $(1 \times 3 \times 32 + 1) \times 32 = 3.104$ parameters. The total number of parameters would be 6,208 parameters, and the reduction is 33%. This theoretical calculation example proves that AC reduces the parameters of the CNN.

The difference between standard convolution and asymmetric convolution is presented in Fig. 2. The standard convolution with filter size 3×3 is conducted by multiplying each pixel and surrounding neighbor with w_i filter, as presented in Fig. 2 (a). The AC consists of two sequence convolutions using a pair of asymmetric filters, where the convolutional process is conducted by operating the 3×1 filter followed by 1×3 filter, as presented in Fig. 2 (b)-(c).

3.3. Residual block and batch normalization

As discussed earlier, the skip connection in our proposed architecture was implemented to support higher classification results. We put a skip connection on each AC layer so that the residue from the previous layer was added at the end of the AC convolution. Residual block is defined using the equation as follows:

$$y = \mathcal{F}(F, \{W_i\}) + F \tag{3}$$

where F and y are the input and output layers viewed in the residual block. Function $\mathcal{F}(F, \{W_i\})$ states the residual mapping trained in the model. In our proposal, we use one layer, then $\mathcal{F} = W_i F$. There are three ACs in our architecture; then the residual blocks will be sent out three times.

The use of residual blocks is motivated by the fact that the identity mapping from one layer to the next is asymptotically equivalent to using several nonlinear layers to estimate a complicated function (He et al., 2016). Therefore, integrating residual blocks into our proposed model would boost the system's ability to handle complex functions and generate high-level features. As stated by the formula above, there is an identity mapping in the next layer after feature map F.

The BN plays an essential role in stabilizing the distribution of activation values and shortening the epoch (Ioffe and Szegedy, 2015). Therefore, we applied BN at each end of the AC layer to get the distribution of ReLU activation results at the end of the AC layer. To calculate the normalization of y in the k-dimension as a result of activation of the previous layer, we used this formula:

$$BN_{\gamma,\beta}(y) = \frac{y - E[y]}{\sqrt{Var[v]}} . \gamma + \beta \tag{4}$$

where $E[y] = \frac{1}{N}\sum_{i=1}^{N} F_i$ is the average of all y data in the k-dimension, whereas $Var[y] = \frac{1}{N}\sum_{i=1}^{N}(y - y_i)^2$ is the variance. Whereas γ and β are scale and shift parameter.

The use of three times AC in the new block can induce an internal covariate shift; as the training data and layers increase, the gradient generated by the activation function gets closer to zero (Ioffe and Szegedy, 2015). As a result, the training carried out also took longer. We added BN at the end of each 3×1 and 1×3 kernel pair convolution to prevent internal covariate shifts and speed up the training process.

3.4. Depthwise separable convolution

Depthwise separable convolutions (DSC) is a convolution method that breaks the standard convolution method into two convolution layers, consisting of a depthwise convolution and a pointwise convolution (Howard et al., 2017). Depthwise convolution used a single filter for each input channel, while pointwise convolution used a 1×1 filter size to combine the outputs of depthwise convolution. The difference between standard convolution and DSC is shown in Fig. 3.

Standard convolution obtains the output feature map using the following formula:

$$y_{k,l,n} = \sum_{i,j,m} \mathcal{F}_{i,j,m,n} . F_{k+i-1,l+j-1,m} \tag{5}$$

Depthwise convolution using individual filter per input channel (depth), we used the following formula:

$$y_{k,l,m} = \sum_{i,j} \widehat{\mathcal{F}}_{i,j,m} . F_{k+i-1,l+j-1,m} \tag{6}$$

where $\mathcal{F}_{i,j,m,n}$ is standard convolution, $\widehat{\mathcal{F}}_{i,j,m}$ is depthwise convolution with size of $D_k \times D_k$. Then the DSC can be formulated into:

$$y = DSC(F, \{W_i\}) = \mathcal{F}\left(\widehat{\mathcal{F}}(F, \{W_i\})\right) \tag{7}$$

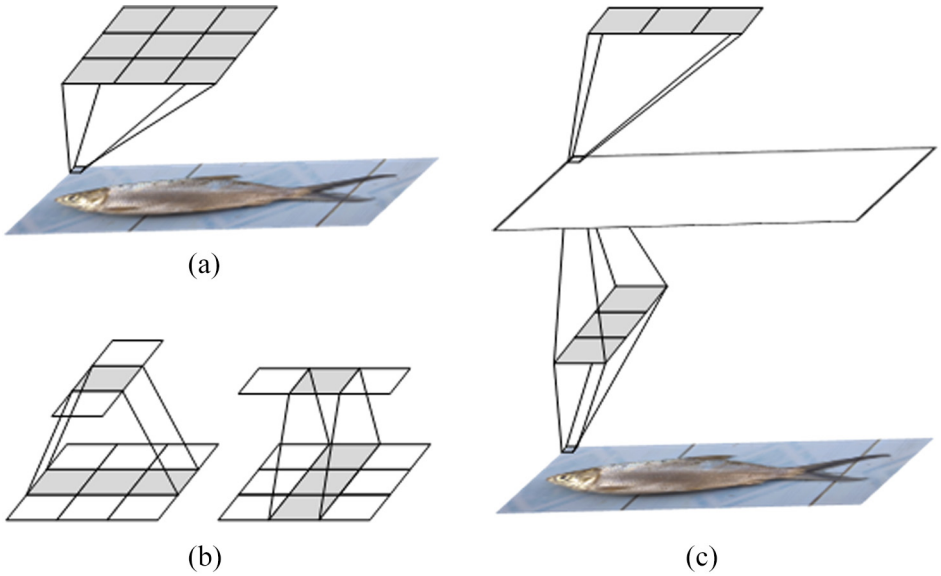

Fig. 2. Asymmetric Convolution, a. Convolution with 3×3 filter, b. Asymmetric filters, c. Asymmetric Convolution with 3×1 filter and 1×3 filter.

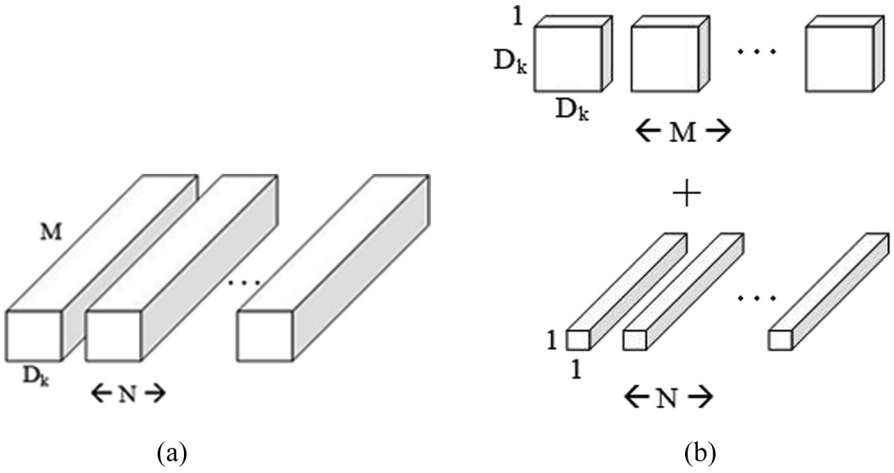

Fig. 3. Convolution, (a) Standard convolution, (b) Depthwise separable convolution.

where $\hat{\mathcal{F}}()$ uses kernel with size of $D_k \times D_k \times 1$ with amount M, whereas $\mathcal{F}()$ uses size of $1 \times 1 \times N$. F is the input feature map, and W is the kernel size of depthwise convolution.

In MobileNet, DSC is the key component of the convolution to achieve tiny models with comparable performance to standard convolution (Howard et al., 2017). It consumes less computational resources due to its small size. DSC is an excellent choice for adjusting the lower and upper layers' feature maps with minimal computation costs to realize MLR in our proposal. To support the MLR-VGGNet model's development, we need only one time DSC.

3.5. Multi-level residual network

Generally, CNN conducts a cascading convolution from the initial layer (producing low-level features) to the final layer (producing high-level features). The low-level features (presence of minor objects) such as points, lines, and textures are produced by the initial convolution layer for representing edge spines, gill covers, fins, and skin textures in the fish classification problem; while the high-level features (presence of major objects) such as heads, tails, and other body parts are produced by the final convolution layer. Due to cascading convolution, the final layer only produces high-level features. Although ResNet also proposes a residue (skip connection) to maintain low-level features that might be lost due to higher-level convolutions, the residual delivery does not reach final layers (He et al., 2016). Densenet also adds residues with concatenation and pointwise convolution, but the delivery of residues ends in the same block, meaning that the CNN has not maintained the initial block's low-level features until the final block (Huang et al., 2017). Therefore, we incorporate the low-level features at the end of the VGGNet convolution block using skip connection through one-time DSC for joining the low- and high-level features.

The main problem in sending low-level features to the final layer is that the feature map sizes are not the same. For example, in VGG16, the feature map size at the end of block one is 112^2x64, while block four is $14^2 \times 512$. We cannot add a skip connection from block one to the end of block four. Moreover, the feature map of block two and block three is $56^2 \times 128$ and $28^2 \times 256$, respectively. Therefore, we proposed a Multi-Level Residue (MLR) to resolve residues with different feature map sizes by modifying the feature map source using a depthwise separable convolution (DSC), as presented in Fig. 4. The source feature map is sent through the skip connection branch, then convoluted using DSC to adjust the feature map size in the destination layer. We did not use zero paddings as a resizing method because there will be a mismatch in the feature's spatial location. The use of one-time DSC is for the following reasons: (1) No significant changes to the features map; (2) Adjusting the size to the destination feature map; (3) Using fewer parameters/computations.

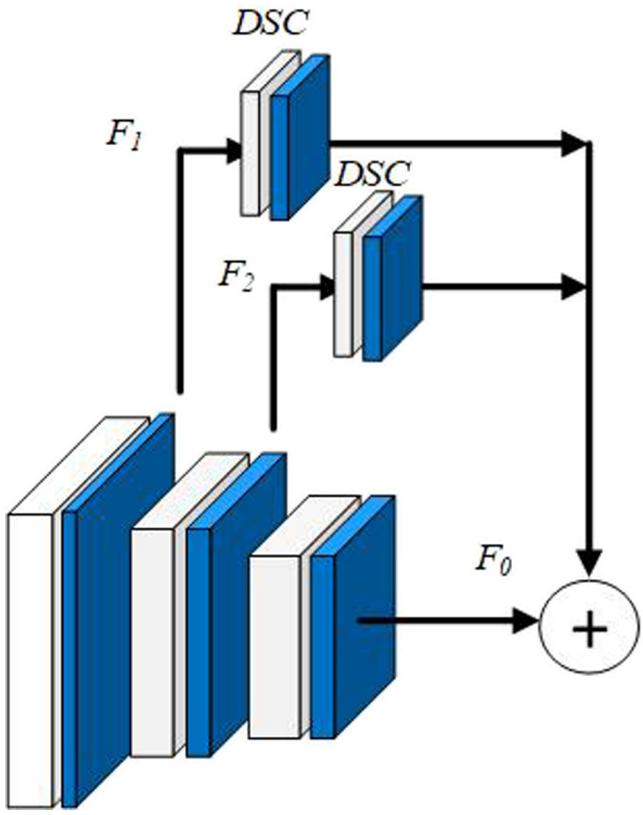

Fig. 4. Multi-Level Residual Network.

Suppose F_i is the low-level features map of the lower layer, W_i is the kernel of depthwise convolution according to Equation (7). The $DSC()$ projects all N low-level features to be the same size as F_0, where F_0 has size of $h_0 \times w_0 \times d_0$ (height × width × feature). The number of d_i features on the F_i layer is usually not the same as the d_0 in F_0. Hence, we modify the DSC formula in Equation (7) to b:

$$\grave{i}\acute{i}\ F_i = DSC(F_i, \{W_i, d_0\}) = \mathcal{F}\left(\hat{\mathcal{F}}(F_i, \{W_i, d_0\})\right) \quad (8)$$

Therefore that $\overset{\text{i¿}}{F}_i$ will be the same size as F_0.

At this point, the d dimension is the same for all F, but not for h and w. We needed to project the two dimensions to be h_0 and w_0; we used max-pooling with size j = 2^{N+1-i} and stride the projection s. We found that the projections of h_i and w_i into h_0 and w_0 are achieved with a stride size s = 2^{N+1-i}. Therefore, max-pooling uses the following Equation:

$$a_j^{(l+1)} = MaxPool\left(a_1^{(l)}, \cdots, a_i^{(l)}, \cdots, a_n^{(l)}\right)_s, i \in R_j^{(l)} \tag{9}$$

where $R_j^{(l)}$ is pooling region j at layer l, $n = \left|R_j^{(l)}\right|$ is the number element in $R_j^{(l)}$. We formulated a stride s = 2^{N+1-i} if the reduction in the image's spatial size is also based on 2 (VGGNet). Thus, we modified Equation (8) to implement max-pooling as follows:

$$\overset{\text{i¿}}{F}_i = MaxPool(DSC(F_i, \{W_i, d_0\}))_n \tag{10}$$

Consequently, we used Equation (10) to project $h_i \times w_i \times d_i$ to be equal in size to $h_0 \times w_0 \times d_0$. Hence, we combined all the feature levels using the following formula:

$$F_y = \sum_{i=1}^{N} \overset{\text{i¿}}{F}_i + F_0 \tag{11}$$

where $\overset{\text{i¿}}{F}_i$ is a projected features map on layer i (low-level features), meanwhile F_0 is a high-level features map on the layer where we combined all the projected featured maps and F_0 into F_y as a result of MLR operation.

3.6. Multi-level residual VGGNet

In this research, we proposed a CNN architecture that uses part of the VGGNet architecture as the backbone (non-green block as in Fig. 1) and adds several components, including Asymmetric Convolution (AC), Multi-Level Residual (MLR), Residual Block (RB), and Batch Normalization (BN). We used an AC block instead of the fifth block of VGGNet, where AC doubled the number of layers but with a smaller kernel size, namely two pairs of asymmetric filters (3×1 and 1×3 filters). In each convolution result using AC, we normalized using BN to avoid internal covariate shift (ICS) and shorten the epoch training. We also strengthened this block with a residual block by adding a skip connection to each AC so that each layer's convolution results can be retained until the last layer. From the design of MLR-VGGNet architecture presented in Fig. 5, we can see that we retain the low-level features of the three blocks by combining them at the end of the fourth block. We used MLR using N = 3 by combining the high-level features with the low-level features from the end of first block, the end of second block, and the end of third block, at the end of fourth block. The first, second, and third blocks present features size $112^2 \times 64$, $56^2 \times 128$, $28^2 \times 256$, respectively. We projected them using DSC + max-pooling so that everything is $14^2 \times 512$. Next, we combined all the feature maps using the addition operator, followed by three times AC, BN, and residual. In the end, we downsampled again using max-pooling size 2^2 with stride 2 so that the feature map became $7^2 \times 512$. The architecture was then connected with a fully-connected layer to perform classification.

The pseudo-code of fish species classification is presented below; we use the entire image as input for all CNN models. The model generates a features map $\overset{\text{i¿}}{F}$, then processes it using the fully-connected classifier to obtain a class label y. During the training, the system conducts augmentation to the training image and then makes it as input for the model.

Input: labeled training data as $\overset{\text{i¿}}{X} = \left\{X^{(1)}, X^{(2)}, \cdots, X^{(K)}\right\}$, K is the total of classes.

Output: $y = f\left(\overset{\text{i¿}}{X}\right)$; % the class label of fish species, $f()$ is CNN model for classifying fish species

$X \leftarrow Augment(X)$; % the augmented image from training data

$\overset{\text{i¿}}{F} \leftarrow MLR\_VGGNet\left(X\right)$; % the extracted feature map from MLR-VGGNet

$y \leftarrow FC\left(\overset{\text{i¿}}{F}\right)$; % $FC()$ is the fully-connected layer for classifying

3.7. Testing

We used accuracy as the metric of performance, both training, validation, and testing result. The accuracy is a comparison of the right result classification with all classification conducted, using the following formula:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{12}$$

where TP is true positive, TN is true negative, FP is false positive, and FN is false negative of all data used in training, validation, or testing session.

4. Results and discussions

4.1. Dataset

In this study, we used the Fish-gres and Fish4-knowledge (F4k) dataset to compare our proposed model and state-of-the-art performance using a proportion of 60:20:20 for training, validation, and test data, respectively—then we presented performance with training, validation, and testing accuracy.

4.1.1. Fish-gres dataset

In this research, a dataset, called Fish-gres dataset, was collected (https://data.mendeley.com/datasets/76cr3wfhff/1), in which this dataset consists of 8 fish species, where the number of images on each species varies from 240 images to 577 images (Prasetyo et al., 2020). The example of image data is presented in Fig. 6 (a)-(h), where the species are Chanos Chanos (500 images), Johnius Trachycephalus (240 images), Nibea Albiflora (252 images), Rastrelliger Faughni (544 images), Upeneus Moluccensis (577 images), Eleutheronema Tetradactylum (240 images), Oreochromis Mossambicus (331 images), and Oreochromis Niloticus (564 images), respectively.

4.1.2. Dataset Fish4-Knowledge

The Fish4-Knowledge (F4K) dataset is an underwater live fish dataset captured from a video dataset in the open sea, used to evaluate a CNN-like architecture (Boom et al., 2012). The number of images is 27,320 images divided into 23 species; each species' image varies from 16 to 12,112. The unbalanced number of images can also result in lacking performance.

4.2. Data augmentation

We conducted data augmentation to obtain more variations in image and oversampling. Data augmentation is a strategy that allows us to add diversity of the data based on the available data without having to add new data. In this research, we used scale augmentation to provide a variety of large and small sizes, rotation

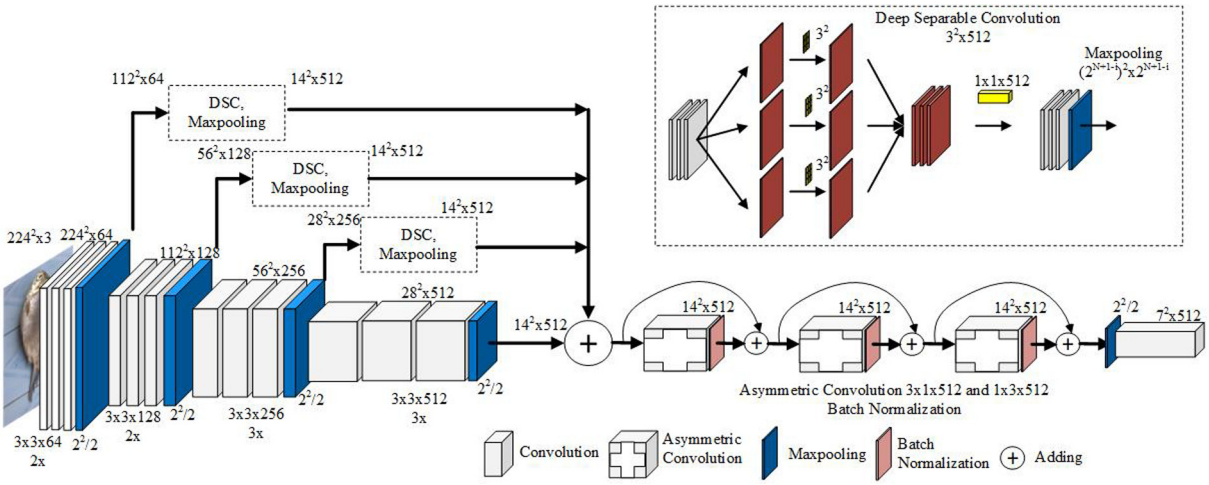

Fig. 5. Multi-Level Residual VGG16.

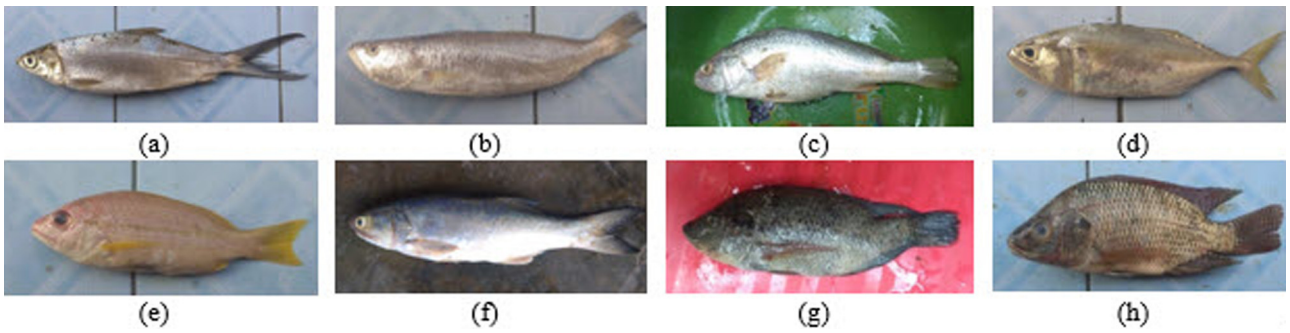

Fig. 6. Fish-gres dataset, (a) Chanos Chanos, (b) Johnius Trachycephalus, (c) Nibea Albiflora, (d) Rastrelliger Faughni, (e) Upeneus Moluccensis, (f) Eleutheronema Tetradactylum, (g) Oreochromis Mossambicus, (h) Oreochromis Niloticus.

to provide various directions for taking images, translation to provide various positions of objects in the image, shearing to provide a variety of shapes, and flipping to provide a variety between left and right.

4.3. Software tools for experimental

We implemented our proposal by experimenting using Python in Google Colaboratory with Tensorflow 2.3.0 and Keras 2.4.3; during training, we used a Tesla T4 GPU with 16 GB of memory provided by Colab. We also use pre-trained CNN comparisons from Keras Application, including VGGNet, Inception V3, ResNet50, and Xception.

4.4. Design experiments

The performance of the proposed MLR-VGGNet was proven by a comparison on various CNN models, such as the original VGG16, VGG19, ResNet50, Inception V3, and Xception. In all comparison models, a transfer learning from pre-trained weights available was also used. The training parameters on all models is given, such as batch size = 20, epoch = 60, optimizer = RMSProp, learning rate = 1e-5, loss function = categorical cross-entropy.

During training, the four blocks of VGGNet were frozen; therefore, the weights did not change to retain the lower layers generalized to generate low-level features. Additional components such as AC, BN, and residuals are fully trained to achieve weight with high-level feature generation that corresponds to the fish species classification.

4.5. The comparison result using Fish-gres dataset

We conducted performance comparison with training, validation, and testing accuracy. The comparison between the proposed architecture and the original VGGNet indicates that the proposed model is superior to the original VGGNet, both VGG16, and VGG19. Our proposed architecture achieved better performance on the data test with 98.46% and 97.84% accuracy for VGG16 and VGG19 backbone, compared to the original VGG16 and VGG19 with 89.83% and 87.51% accuracy, respectively. There is a relative increase of up to 8.63% and 10.33%, respectively. These results show that the proposed architecture is superior to the original VGGNet.

As presented in Table 1, the comparison with state-of-the-art shows that our proposed model is also superior to all state-of-the-art except ResNet50. Almost all methods gained high performance; for example, ResNet50 achieved testing accuracy of 97.84%, while our proposed model achieved higher performance of 98.46% accuracy. Our models also achieved the highest performance during training and validation, where accuracy was up to 97.03% and 99.69%, respectively. Only ResNet50 has a competitive performance upon our model, where the performance is close to our proposal. Our proposal achieves a relative improvement of up to 5.24% on the test data at the Inception V3 model.

The Fig. 7 shows examples of the results of our proposed classification and other models. In general, our proposals and other models classify similarly, as seen in Fig. 7 (a), where our proposals and other models classify correctly. In some instances, our proposed model is more robust than other models, such as Fig. 7 (b), MLR-VGG16, MLR-VGG19, and original VGG16, which classify cor-

Table 1
The performance (%) of models using Fish-gres and F4K dataset.

Model	Fish-gres			F4K		
	Training	Val.	Test	Training	Val.	Test
VGG16	82.30	91.36	89.83	80.52	83.85	82.44
VGG19	76.30	89.93	87.51	86.21	90.14	88.07
ResNet50	95.20	98.42	97.84	82.80	90.40	88.12
Inception V3	82.85	91.61	93.22	77.55	87.75	85.66
Xception	89.95	95.73	93.53	80.20	86.75	83.70
MLR-VGG16	96.97	**99.69**	**98.46**	**97.11**	**96.66**	**96.25**
MLR-VGG19	**97.03**	98.92	97.84	**98.09**	**97.84**	**97.09**

	(a)	(b)	(c)	(d)	(e)
Ground truth	0	6	2	4	5
MLR-VGG16	0	6	2	4	1
MLR-VGG19	0	6	2	4	1
VGG16	0	6	3	1	5
VGG19	0	7	3	1	5
ResNet50	0	7	7	4	5
Inception V3	0	7	0	4	5
Xception	0	7	0	4	5

Fig. 7. Examples of image classification results in the Fish-gres dataset, *0 : Chanos chanos; 1 : Johnius Trachycephalus; 2 : Nibea Albiflora; 3 Rastrelliger Faughni; 4 : Upeneus Moluccensis; 5 : Eleutheronema Tetradactylum; 6 : Oreochromis Mossambicus; 7 : Oreochromis Niloticus.*

Table 2
Comparison between our proposed and several CNN architectures in term the number of parameters, inference time and model size.

Model	Parameter (millions)	Decreased Parameters (%) Compared to MLR-VGGNet		Inference time/image (ms)	Model size (MB)
		MLR-VGG16	MLR-VGG19		
VGG16	14.71	**14.34**	−5.71	509	**252**
VGG19	20.02	37.06	**22.32**	643	**273**
ResNet50	23.59	**46.59**	34.08	173	874
Inception V3	21.80	42.20	28.67	**117**	484
Xception	20.86	39.60	25.46	227	864
MLR-VGG16	**12.60**	–	–	556	281
MLR-VGG19	**15.55**	–	–	679	302

rectly while other models are incorrect. Moreover, in Fig. 7 (c) - (d), our proposal and some other models classify correctly while the rest are wrong. Our proposal is also robust in classifying images with multiple fish, such as Fig. 7 (c), while the other models classify the images incorrectly. However, in some instances, our proposal does not classify correctly, as shown in Fig. 7 (e), where the single fish is not correctly classified. In general, our proposal classifies fish species similarly to other models, but it is more robust when dealing with images that include several fish. This result demonstrates that our proposed models correctly classified more images, with an accuracy of up to 99.69%. The other models, such as ResNet50, achieved the highest accuracy of 98.42%. In conclusion, our proposed architecture model outperformed all state-of-the-art with very satisfactory performance in Fish-gres dataset.

4.6. The comparison result using F4K dataset

In comparison using the F4K dataset, as presented in Table 1, comparison with original VGG16 and VGG19, our proposal outperformed both. Our proposed model also achieved the highest performance and outperformed all state-of-the-art training, validation, and testing accuracy of 97.11%, 96.66%, and 96.25%, respectively for VGG16, and accuracy of 98.09%, 97.84%, and 97.09%, respectively for VGG19. This experiment proves that our proposed MLR-VGGNet architecture has a promising performance.

The interesting point is that all state-of-the-art cannot work optimally with performance below 90% due to an imbalance dataset. However, we also conducted augmentation in this experiment but could not help improve performance. Performance using this dataset shows that our proposed model can solve the imbalance problem and achieve the best performance superior to all state-of-the-art.

4.7. Analysis of parameters Number, inference time and model size

The number of parameters in the features learning part of several architectures are presented in Table 2. These parameters were not included the parameters in the classifier part of CNN. We showed that our proposed model used a few features; MLR-VGG16 and MLR-VGG19 use 12.60 and 15.55 million parameters, respectively. These parameters are decreased by 14% and 22% compared to the original VGG16 (14.71 million) and VGG19 (20.02 million) parameters. MLR-VGGNet also has fewer parameters than state-of-the-art such as ResNet50, Inception V3, and Xception, which is less up to 46.59% on MLR-VGG16 vs ResNet50. Hence, our proposed model used the lowest number of parameters.

The result presented in Table 2 shows that our proposal's inference needs slightly longer than the original VGGNet, while the MLR-VGG16 and original VGG16 needed 556 and 509 ms, respectively, 9% longer. In MLR-VGG19, the execution time is 6% longer than the original VGG19. Meanwhile, other models are considerably shorter than our proposal, even though other models use a more significant number of parameters. Other models, such as Resnet50 and Inception V3, require faster execution times because they employ a more specific microarchitecture in computational complexity. Although other models have faster execution times, the required size models are also huge compared to MLR-VGGNet. For example, ResNet50 and Xception require 874 MB and 864 MB of space, respectively, to store architecture and weights; and this is certainly not suitable for application on devices with limited storage capacity such as mobile devices. The MLR-VGG16 and MLR-VGG19, on the other hand, require 281 MB and 302 MB, respectively, to store architecture and weights. This number is also slightly higher than the original version by 12% and 11%, respectively, but these numbers are also within reasonable limits.

5. Conclusion

Automatic fish classification using Convolutional Neural Network provides the advantage by renouncing several steps related to data or features analysis through several cascading CNN convolution. Nevertheless, cascading convolution on images produces only high-level features in the final block and leaves low- and middle-level features in earlier blocks. To keep going low- and middle-level features at the final block, we propose Multi-Level Residual (MLR) as a new residual network strategy by combining low-level features of the initial block with high-level features of the last block using depthwise separable convolution (DSC). We use VGGNet as the backbone of the new CNN architecture with improvisation by removing the fifth block and replacing it with some components such as MLR, asymmetric convolution (AC), batch normalization (BN), and residual features. We called them as MLR-VGGNet to achieve higher fish classification performance. Our experimental results show that MLR-VGGNet achieved better performance than the other pre-trained CNN models, which achieved an accuracy of up to 99.69% on the Fish-gres and F4K datasets. The number of parameters has also been reduced by 37%. With excellent performance, our proposal is appropriate for implementation in the automatic fish species classification system.

MLR-VGGNet uses conventional convolution, especially in the first four blocks with a high number of parameters. This number of parameters is still possible to be reduced through particular convolutional strategies and keep going the high performance; this also requires further analysis in terms of initial weight considering that MLR-VGGNet uses the initial weight of pre-trained VGGNet.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We thank the Deputy in Strengthening Research and Development, Ministry of Research and Technology / National Research and Innovation Agency, Indonesia for supporting the research of authors with contract number 829/PKS//ITS/2021, on March 10, 2021.

References

Aderghal, K., Afdel, K., Benois-Pineau, J., Catheline, G., 2020. Improving Alzheimer's stage categorization with Convolutional Neural Network using transfer learning and different magnetic resonance imaging modalities. Heliyon 6 (12), e05652. https://doi.org/10.1016/j.heliyon.2020.e05652.

Amin, J., Sharif, M., Anjum, M.A., Raza, M., Bukhari, S.A.C., 2020. Convolutional neural network with batch normalization for glioma and stroke lesion detection using MRI. Cognit. Syst. Res. 59, 304–311. https://doi.org/10.1016/j.cogsys.2019.10.002.

Ayan, E., Erbay, H., Varçın, F., 2020. Crop pest classification with a genetic algorithm-based weighted ensemble of deep convolutional neural networks. Comput. Electron. Agric. 179, 105809. https://doi.org/10.1016/j.compag.2020.105809.

Bermejo, S., 2007. Fish age classification based on length, weight, sex and otolith morphological features. Fish. Res. 84 (2), 270–274. https://doi.org/10.1016/j.fishres.2006.12.007.

Boom, B.J., Huang, P.X., He, J., Fisher, R.B., 2012. Supporting ground-truth annotation of image datasets using clustering. Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012). IEEE, Tsukuba, Japan.

Boussaad, L., Boucetta, A., 2020. Deep-learning based descriptors in application to aging problem in face recognition. J. King Saud Univ. Comput. Inform. Sci. https://doi.org/10.1016/j.jksuci.2020.10.002.

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017. https://doi.org/10.1109/CVPR.2017.195

Hafiz, R., Haque, M.R., Rakshit, A., Uddin, M.S., 2020. Image-based soft drink type classification and dietary assessment system using deep convolutional neural network with transfer learning. J. King Saud Univ. – Comput. Inform. Sci. https://doi.org/10.1016/j.jksuci.2020.08.015.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR.2016.90.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.

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.

Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q., 2017. Densely connected convolutional networks, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017. https://doi.org/10.1109/CVPR.2017.243

Ijjina, E.P., Chalavadi, K.M., 2016. Human action recognition using genetic algorithms and convolutional neural networks. Pattern Recogn. 59, 199–212. https://doi.org/10.1016/j.patcog.2016.01.012.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: 32nd International Conference on Machine Learning, ICML 2015.

Jalal, A., Salman, A., Mian, A., Shortis, M., Shafait, F., 2020. Fish detection and species classification in underwater environments using deep learning with temporal information. Ecol. Inf. 57, 101088. https://doi.org/10.1016/j.ecoinf.2020.101088.

Jaouedi, N., Boujnah, N., Bouhlel, M.S., 2020. A new hybrid deep learning model for human action recognition. J. King Saud Univ. – Comput. Inform. Sci. 32 (4), 447–453. https://doi.org/10.1016/j.jksuci.2019.09.004.

Kabir Anaraki, A., Ayati, M., Kazemi, F., 2019. Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms. Biocyber. Biomed. Eng. 39 (1), 63–74. https://doi.org/10.1016/j.bbe.2018.10.004.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems.

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.

Lee, J., Asahi, R., 2021. Transfer learning for materials informatics using crystal graph convolutional neural network. Comput. Mater. Sci. 190, 110314. https://doi.org/10.1016/j.commatsci.2021.110314.

Li, H.e., Li, X., Yuan, F., Jowitt, S.M., Zhang, M., Zhou, J., Zhou, T., Li, X., Ge, C., Wu, B., 2020. Convolutional neural network and transfer learning based mineral prospectivity modeling for geochemical exploration of Au mineralization within the Guandian-Zhangbaling area, Anhui Province, China. Appl. Geochem. 122, 104747. https://doi.org/10.1016/j.apgeochem.2020.104747.

Lumini, A., Nanni, L., 2019. Deep learning and transfer learning features for plankton classification. Ecol. Inf. 51, 33–43.

Połap, D., 2020. An adaptive genetic algorithm as a supporting mechanism for microscopy image analysis in a cascade of convolution neural networks. Applied Soft Computing Journal 97, 106824. https://doi.org/10.1016/j.asoc.2020.106824.

Prasetyo, E., Suciati, N., Fatichah, C., 2020. Fish-gres Dataset for Fish Species Classification. https://doi.org/http://dx.doi.org/10.17632/76cr3wfhff.1

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.

Rangarajan, A.K., Purushothaman, R., Ramesh, A., 2018. Tomato crop disease classification using pre-trained deep learning algorithm, in: Procedia Computer Science. Elsevier, pp. 1040–1047. https://doi.org/10.1016/J.PROCS.2018.07.070

Rodrigues, L.F., Naldi, M.C., Mari, J.F., 2019. Comparing convolutional neural networks and preprocessing techniques for HEp-2 cell classification in immunofluorescence images. Comput. Biol. Med. 116, 103542. https://doi.org/10.1016/j.compbiomed.2019.103542.

Shallu, Mehra, R., 2018. Breast cancer histology images classification: Training from scratch or transfer learning? ICT Express 4, 247–254. https://doi.org/10.1016/J.ICTE.2018.10.007

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition, in: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR.2015.7298594.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR.2016.308.

Tharwat, A., Hemedan, A.A., Hassanien, A.E., Gabel, T., 2018. A biometric-based model for fish species classification. Fish. Res. 204, 324–336. https://doi.org/10.1016/J.FISHRES.2018.03.008.

Wang, J., Li, S., An, Z., Jiang, X., Qian, W., Ji, S., 2019. Batch-normalized deep neural networks for achieving fast intelligent fault diagnosis of machines. Neurocomputing 329, 53–65. https://doi.org/10.1016/j.neucom.2018.10.049.

Yang, Q., Shi, W., Chen, J., Lin, W., 2020. Deep convolution neural network-based transfer learning method for civil infrastructure crack detection. Autom. Constr. 116, 103199. https://doi.org/10.1016/j.autcon.2020.103199.