INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Improved Image Classification Task Using Enhanced Visual Geometry Group of Convolution Neural Networks

Nurzarinah Zakaria [a], Yana Mazwin Mohmad Hassim [a,*]

[a] *Faculty of Computer Science & Information Technology, Universiti Tun Hussein Onn Malaysia, Batu Pahat, Johor, Malaysia*

*Corresponding author: *yana@uthm.edu.my*

***Abstract*— Convolutional Neural Networks (CNNs) have become essential to solving image classification tasks. One of the most frequent models of CNNs for image classification is the Visual Geometry Group (VGG). The VGG architecture is made up of multiple layers of convolution and pooling processes followed by fully connected layers. Among the various VGG models, the VGG16 architecture has gained great attention due to its remarkable performance and simplicity. However, the VGG16 architecture is still prone to have many parameters contributing to its complexity. Moreover, the complexity of VGG16 may cause a longer execution time. The complexity of VGG16 architecture is also more highly prone to overfitting and may affect the classification accuracy. This study proposes an enhancement of VGG16 architecture to overcome such drawbacks. The enhancement involved the reduction of the convolution blocks, implementing batch normalization (B.N.) layers, and integrating global average pooling (GAP) layers with the addition of dense and dropout layers in the architecture. The experiment was carried out with six benchmark datasets for image classification tasks. The results from the experiment show that the network parameters are 79% less complex than the standard VGG16. The proposed model also yields better classification accuracy and shorter execution time. Reducing the parameters in the proposed improved VGG architecture allows for more efficient computation and memory usage. Overall, the proposed improved VGG architecture offers a promising solution to the challenges of long execution times and excessive memory usage in VGG16 architecture.**

***Keywords*— Convolutional neural networks; computer vision; deep learning; image classification; Visual Geometry Group.**

Manuscript received 16 Apr. 2023; revised 24 Jul. 2023; accepted 14 Sep. 2023. Date of publication 31 Dec. 2023.

I. INTRODUCTION

Image classification is one of the applications in machine learning, aiming to develop algorithms that can recognize and classify the content of an image with near human-level accuracy. The recent years have witnessed significant improvements in the use of advanced machine learning techniques, namely deep learning, for image classification [1],[2]. Deep learning techniques, notably Convolutional Neural Networks (CNNs), have revolutionized the vision of a computer in the field of image classification [3], [4], [5], making it the most preferred and implemented machine learning method. Moreover, image classification has demonstrated outstanding performance employing CNNs to extract deep representations of training data where it produced results equal to or better than those produced by humans.

As a deep learning technique, CNNs take an input image and apply weights and biases to various objects to aid object recognition [6]. There have been many CNN models up until this point, including AlexNet [7], DenseNet [8], GoogleNet/InceptionNet [9], LeNet [10], ResNet [11], SqueezeNet [12], Visual Geometry Group (VGG) [13], Xception [14] and ZFNet [15]. Among the various CNNs, the VGG architecture stands out for its simplicity and efficacy in image classification tasks. The VGG16 network has demonstrated state-of-the-art results in various computer vision applications, including object recognition [16], image classification [17], and image retrieval [18].

The VGG16, one of the VGG networks, is a deep neural network architecture that concludes with 16 layers. It was introduced in 2014, and cutting-edge results in image classification tasks were obtained[13]. The network architecture comprises 13 convolution layers stacked with three fully connected layers. The convolution layers use small filters of size 3 x 3. Despite its success, the VGG architecture's deep and parameter-heavy design addresses challenges in practical implementations, especially for real-time applications and resource-constrained devices. A larger parameter size results in a longer execution time and produces more memory usage, leading to limited scalability [19]. To address this drawback, this study focuses on reducing the VGG parameter while maintaining its high accuracy. The proposed method involves downsizing the VGG architecture by reducing the parameters, aiming to make the model more compact and efficient. The ultimate objective is to strike a balance between execution time and accuracy, thereby enhancing the reliability and applicability of VGG in real-world scenarios.

By achieving a downsized VGG architecture, the proposed method can provide better image classification performance than other existing VGG architectures with faster execution times. The need to improve the reliability of VGG in practical applications motivates this research, where speed and efficiency are critical factors.

II. Materials and Method

This study aims to provide an improved VGG architecture for image classification by downsizing the network and training it on six image datasets. The proposed method contains two convolution blocks with a pooling and batch normalization (B.N.) layer in each block. To ensure linearity, the ReLu activation function is used during training [20]. Following that, the global average pooling (GAP) layer, three dense layers with SoftMax activation function and dropout of 0.2, 0.4, and 0.6, respectively, were used to analyze the images.

Furthermore, the B.N. layer has been put together to normalize the output of previous layers in order to address the issue of randomly changed minibatch samples that could result in gradients exploding or disappearing [21] during the feature extraction. Next, the GAP layer will be employed for the classifier rather than the flatten layer to minimize the number of parameters that cause a longer execution time and may lead to overfitting [21]. Moreover, instead of using two dense layers from the standard VGG16, three layers will be implemented because by adding extra layers to the dense part, the network's efficacy can be strengthened, and the accuracy can be increased [22].

Finally, dropout layers will be placed after each dense layer to reduce the complexity and avoid overfitting, which can reduce network performance [23]. Dropout generates independent activations by establishing independent random gates for neurons in a layer [24], which also cuts off the incoming and outgoing connections to the neurons when the neurons are turned off. This step is taken to enhance the learning process of the network. Fig. 1 illustrates the architecture of the proposed improved VGG.

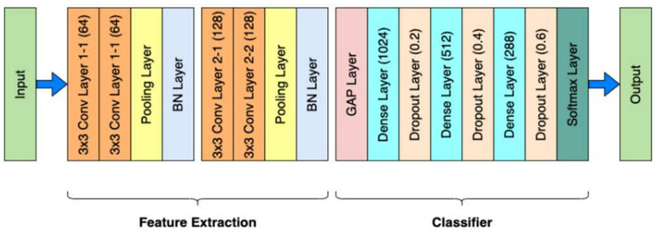

Fig. 1 The proposed improved VGG architecture

Several steps have been taken to accomplish the study's goal, including data collecting, architecture development, implementation, and performance analysis of the architectures. The method structure is graphically presented in Fig. 2. The next subsections describe the actions in more detail.

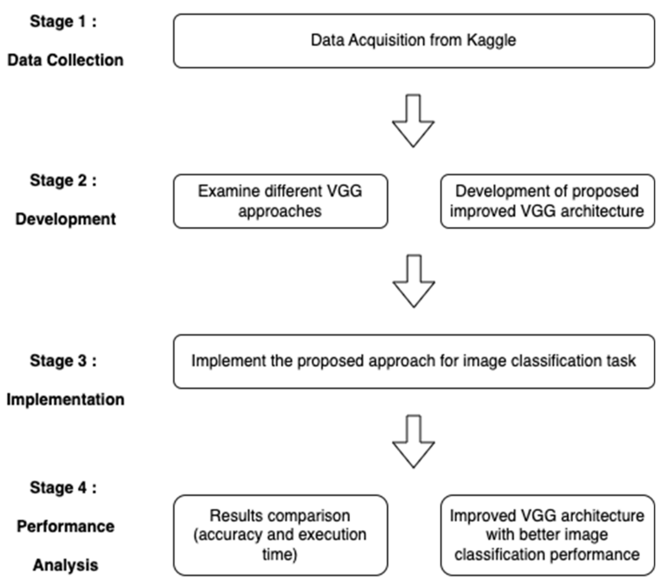
Stage 1 :
Data Collection
Data Acquisition from Kaggle
Stage 2 :
Development
Examine different VGG approaches
Development of proposed improved VGG architecture
Stage 3 :
Implementation
Implement the proposed approach for image classification task
Stage 4 :
Performance Analysis
Results comparison (accuracy and execution time)
Improved VGG architecture with better image classification performance

Fig. 2 The method structure

A. Data Collection

In this research, a comprehensive evaluation was conducted using six Kaggle image datasets. The datasets were selected to encompass various image categories, including brain tumor, cloth, forest, road, Rome weather, and room. Table 1 provides a detailed overview of each dataset, including the name, total number of images, and number of classes. To ensure unbiased training and testing of the architecture, the entire dataset was randomly split into two divisions: the training set and the testing set. The amount of 80% of the whole dataset was used for training purposes, while the other 20% remaining amount of the dataset has been utilized for testing the architecture.

TABLE I
Dataset Details

No	Image Datasets		
	Name	**Total Images**	**No. of Classes**
1	Brain Tumor	171	2
2	Cloth	398	2
3	Forest	166	2
4	Road	669	2
5	Rome Weather	250	5
6	Room	192	2

The selection of the brain tumor, cloth, forest, road, Rome weather, and room datasets in the experiment reflects various domains and classification tasks, spanning medical imaging, fashion, environmental monitoring, road infrastructure, weather, and room conditions. While some datasets have multiple classes and others have two classifications, the choice comprehensively evaluates the proposed architecture's performance across various real-world applications. Including different domains and classifications, complexities can impact the results, requiring specific considerations in model design and evaluation metrics. Overall, this diverse dataset selection enables insights into the architecture's effectiveness and its potential application in multiple domains.

B. Process of Developing the Proposed Method

The VGG model is employed for image classification in CNNs due to its high accuracy [25], [26]. However, the drawback of VGG model is its large number of parameters, which can cause slower execution times. Additionally, the abundance of parameters requires a longer training duration, and the broader model is more susceptible to overfitting, which can impact classification accuracy [27]. To evaluate the efficacy and reliability of the proposed method, we compared the evaluation results with other existing architectures, namely VGG16 [13], VGG16 2021 [22] and VGG 2020 [21]. This stage determines if the proposed architecture could outperform or match the performance of the existing architecture.

To facilitate the comparison, this study included a table containing the details of the compared architectures. Table 2 includes three columns: the key used to address the article reviewed, the reference source, and the name of the model developed by the authors. By comparing the proposed method's performance with existing architectures, this study aimed to demonstrate its superiority and ability to be applied in actual circumstances. This comparative analysis also provides insights into the strengths and weaknesses of each architecture, which could inform future research in the field.

TABLE II
THE DETAILS OF THE ARCHITECTURE

Key	Reference	Method
A1	Simonyan & Zisserman [13]	VGG16
A2	Islam et al., [22]	VGG16 2021
A3	Zhongqin Bi et al., [21]	VGG 2020

VGG16 is a type of architecture within CNNs, proposed in 2013 and refined based on the 2014 ImageNet Challenge. VGG16 differs from earlier models like AlexNet and ZFNet by using smaller 3x3 receptive fields with a stride of one pixel for the whole network, as opposed to larger fields used in previous models. VGG16 has five sets of convolutional layers and three fully connected layers and employs max pooling to reduce dimensionality [28]. It uses three fully connected layers after the convolutional layers, the first two with 4096 neurons and the last with 1000 neurons corresponding to ImageNet classes. The SoftMax activation layer is placed after the output layer for classification [29]. Next, VGG16 2021 has improved standard VGG16 by removal of several layers, particularly after the first 3 x 3 layer in the fifth convolutional block, as depicted in Fig. 3.

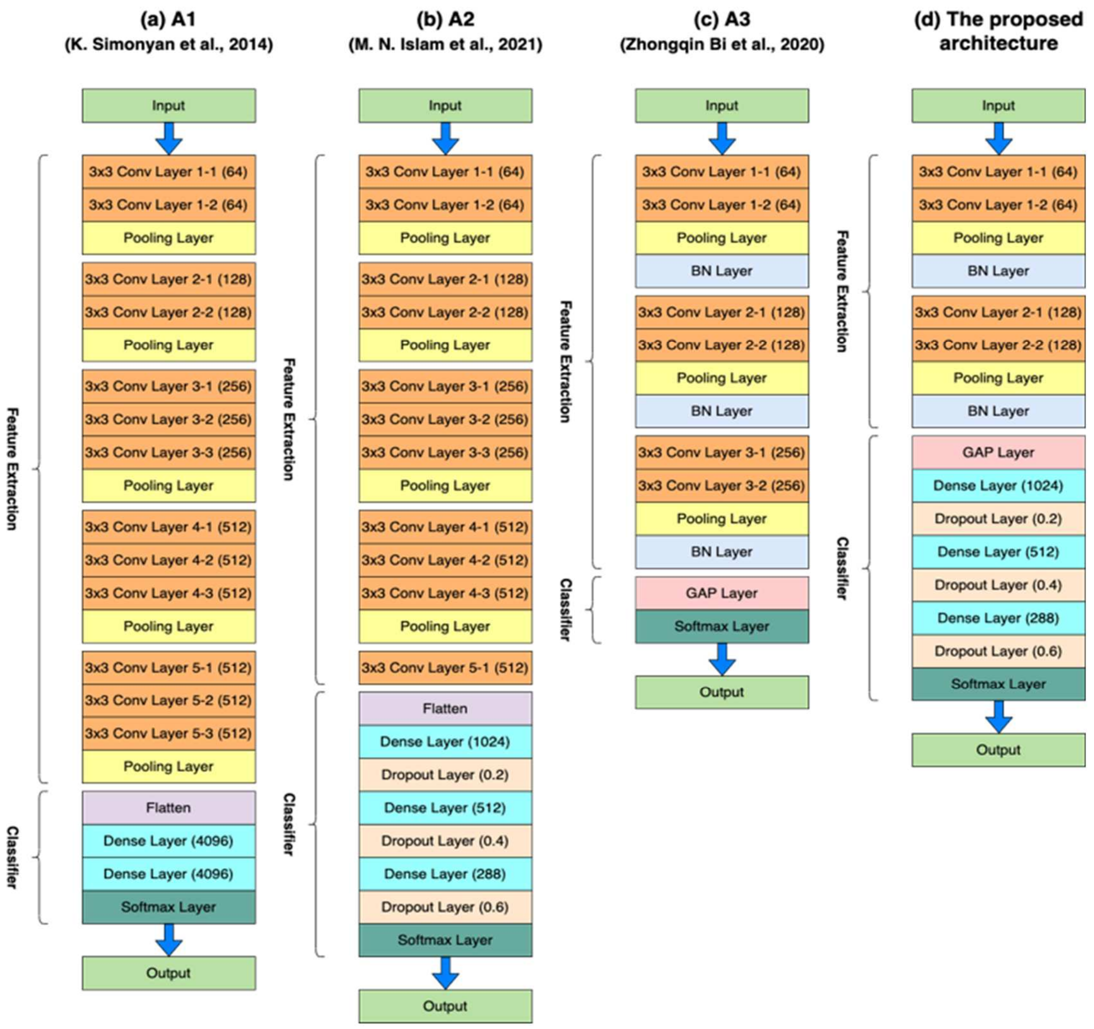

Fig. 3 The comparison of the architecture

To enhance robustness and classification accuracy, new dense layers were introduced in the fully connected block, consisting of three layers with units of 1,024, 512, and 288. Dropout layers were added after each dense layer to prevent overfitting, with dropout rates of 0.2, 0.4, and 0.6, respectively. The first layers until the fourth convolutional block were frozen during training. Furthermore, VGG16 2020 retains the first three blocks of VGG16 while removing a convolution layer from the third block. To counter gradient issues during training, a B.N. layer is introduced to normalize input maps for improved stability. The standard VGG16 includes two dense layers with 4096 units each, leading to complexity and slower execution. In contrast, the VGG16 2020 replaces these with a GAP layer, addressing parameter count and computation speed.

Finally, the proposed architecture combines VGG16 2021 and VGG16 2020 to simplify the complexity and computational requirements of the standard VGG16. In comparison to VGG16, VGG16 2021, and VGG16 2020 with 139 590 725, 134 281 029, 113 430 533 and 1 148 485 parameters respectively, the new architecture reduces parameters to 1 067 013. This results in faster execution due to a GAP layer replacing the flattened layer. Minimizing convolution blocks is crucial to prevent overfitting and maintain classification accuracy [27].

The standard VGG16, VGG16 2021, VGG16 2020, and the proposed architecture are varied in terms of parameters and computational efficiency. The standard VGG16 has larger parameters, potentially slowing down the execution time and demanding more resources [19], [21]. VGG16 2021 and VGG16 2020 introduce modifications to improve these aspects. Adopting the features of VGG16 2021 and VGG16 2020, the proposed architecture strives to reduce parameters, shorten the execution time, and enhance the accuracy of its applicability in real-world scenarios.

Fig. 3 is a visualization illustrating the network developed in VGG16, VGG16 2021, VGG16 2020, and the proposed method. It is likely a diagram that depicts the flow of data through the neural network, including the types of layers used, the number of filters used in each layer, and the connections between them. The input size of an architecture refers to the dimensions of the image that it can accept as input. In this case, all the architectures mentioned above have the same input size, which is 224 for height and 224 for width, with a depth of 3 representing the three-color channels (red, green, and blue) of the RGB image.

This research was conducted with specific hardware and software configurations. The training process was executed on a computer with M1 chip, 8-core GPU, 8-core CPU, and 16-core Neural Engine. The computer was equipped with 16 G.B. of RAM, and the programming language used for writing the code was Python. Jupyter Notebook was the platform for developing and executing the code, while TensorFlow was the deep learning framework employed for the experiment. The number of epochs set for the training process was 100.

C. *Implementation of the Proposed Architecture*

The third stage of the research involves the implementation of the proposed architecture. The expected outcome is to reduce the complexity of the standard VGG16 architecture. The comparison between the overall architecture of the standard VGG16 and the proposed improved VGG can be visualized side-by-side in Fig. 4. The proposed architecture is expected to show better performance in terms of accuracy and speed. The implementation will be carried out using the same experimental setup as described in the previous statement.

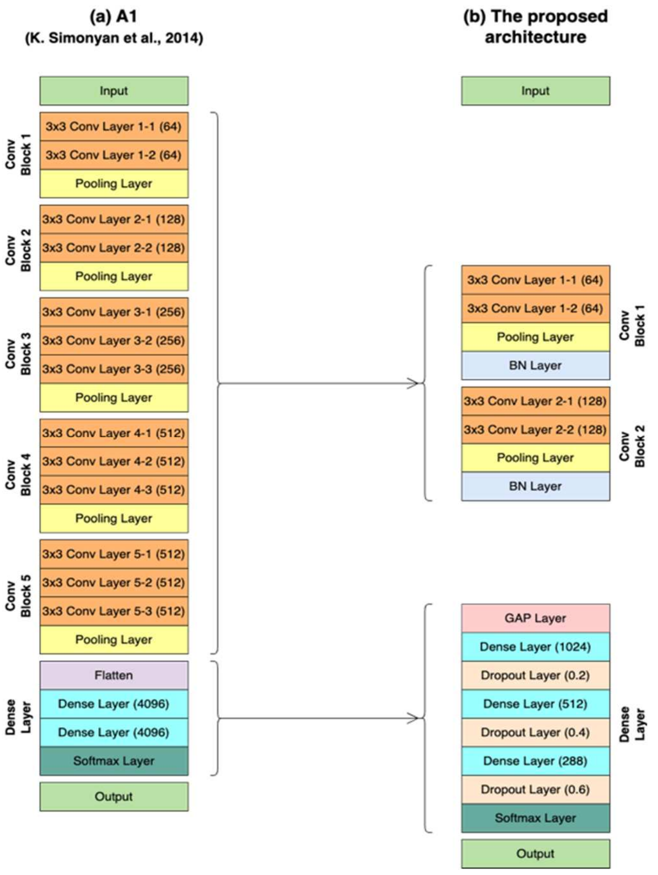

Fig. 4 The improvement of the VGG16 architecture

Implementing the proposed method involves including B.N. layers located after every pooling layer during the feature extraction phase, which is not present in the standard VGG16 architecture. This modification is expected to accelerate the training speeds of the architecture [30]. Including B.N. layers during each epoch helps normalize data distribution and leads to improved convergence speeds while reducing the tendency of parameter changes caused by varied data distributions [31]. Fig. 5 shows the network's improvement in the architecture's first convolution block.

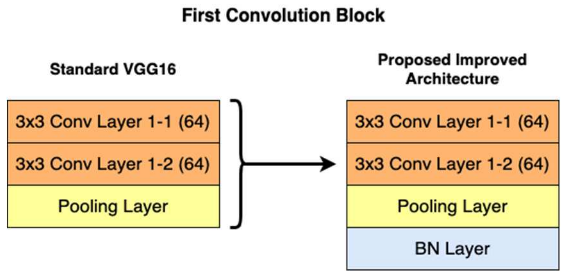

Fig. 5 The first convolution block of the proposed architecture

To address the issue of large memory usage in VGG16, the proposed architecture adopted a strategy of reducing the number of convolution blocks. Specifically, the first and second convolution blocks in the standard VGG16 were frozen, and the remaining blocks were removed. This led to a significant reduction in the number of total parameters from 134 268 738 in the standard VGG16 to 1 067 013 in the proposed architecture. This reduction in parameter count not only addresses the issue of limited storage capacity in several computers but also leads to faster execution times. In this study, the implementation focuses on smaller or simpler datasets, where a compact model can effectively learn the relevant features [32], [33]. The improvement of the second convolution block of the proposed architecture is illustrated in Fig. 6.

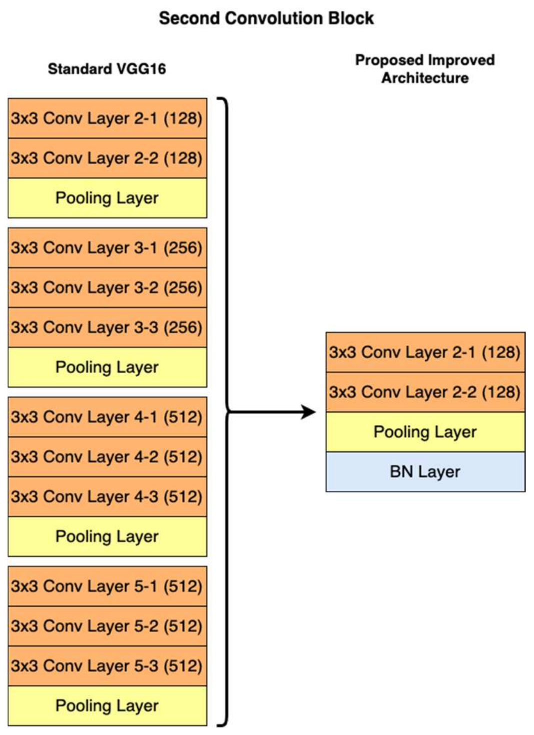

Fig. 6 The second convolution block of the proposed method

In the standard VGG16 architecture, the flattening layer has a large number of elements in the output of the last convolution layer after flattening, precisely a vector of size 25 088, which impacts the overall execution time and memory usage. In order to address this issue, this study utilizes the GAP layer instead of the flattened layer, as it is parameter-free, which can possibly significantly decrease the number of parameters in the architecture [34]. According to a previous study, utilizing GAP layer has been found to reduce the number of parameters and computational complexity [35]. Using the GAP layer decreases the vector's size in the layer to 128, which is obtained from the previous layer. Fig. 7 represents the classifier of the standard VGG16 and the proposed method. The proposed method uses the GAP layer instead of the flattened layer, which can minimize the number of parameters and improve the execution time.

Adding extra dense layers can help to analyze more complicated forms of the input data and increase the model's capacity to capture high-level features [22]. This can ultimately lead to better classification performance. Furthermore, every neuron in each dense layer is typically a hyperparameter that must be tuned based on the dataset and the complexity of the task. In this study, the proposed architecture uses three dense layers with 1024, 512, and 288 neurons to achieve better classification performance compared to the standard VGG16. Three dense layers give the model more ability to recognize and learn complex correlations and patterns in input data [36]. It is worth noting that adding more layers and increasing the neurons can also increase the tendency to overfit. Therefore, it is important to carefully choose the number of layers and neurons and apply regularization techniques, such as dropout, to prevent overfitting.

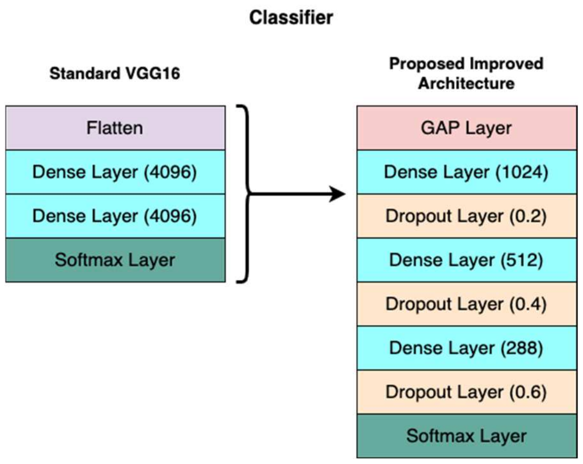

Fig. 7 The classifier of the proposed method

Finally, dropout layers are a regularization technique used to reduce overfitting in neural networks [20], [37]. Overfitting is a problem that happens when a model becomes overly complex to learn the data used for training [38], [39], [40], [41]. Hence, the overfitting must be reduced in order to lower the cost of misclassification in the architecture [42]. During training, the number of neurons in the network is randomly dropped out or ignored, which forces the network to analyze more robust and generalizable features. The dropout rate is a hyperparameter that needs to be tuned based on the specific problem and dataset. This research added dropout layers with rates of 0.2, 0.4, and 0.6 after each dense layer.

D. Performance Analysis

Several evaluation metrics have been used to measure and compare the proposed method's effectiveness with other architectures. These metrics include the execution time which measures the time taken for the model to make predictions, and the classification accuracy, which measures the ability of the model to classify images into their respective categories correctly. In order to ensure a fair evaluation of the proposed method's performance, it is important to evaluate its execution time on the same platform and in the same state. This is because varying machine specifications or tool versions could introduce bias in the feature extraction, and other tasks running on the CPU could impact the time taken for feature extraction. In order to ensure consistent evaluation, the execution time will be measured in hours (h).

The proposed architecture's accuracy will be assessed using the confusion matrix, a straightforward and uncomplicated metric for measuring correctness and accuracy. The confusion matrix is typically employed in classification problems where the output could belong to two or more classes. It comprises a two-dimensional table which is Actual and Predicted with sets of classes in both dimensions. A diagrammatic representation of the confusion matrix is displayed in Fig. 8.

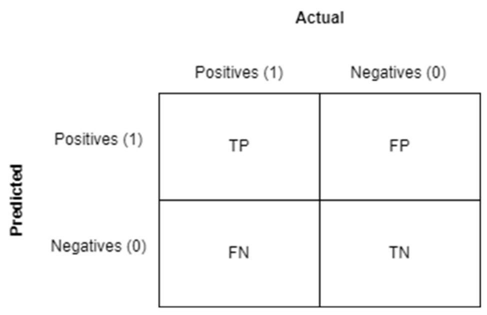

Fig. 8 Confusion matrix

In Fig. 8, a confusion matrix is presented to evaluate the performance of the classification model. In the context of classification problems, accuracy is a crucial metric that represents the proportion of correct predictions made by the model out of all the predictions made. Equation (1) is used to calculate accuracy, which considers the number of True Positives (T.P.), True Negatives (T.N.), False Positives (F.P.), and False Negatives (F.N.).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \quad (1)$$

III. Results and Discussion

The experimental results of the proposed improved VGG architecture are presented in this section, including the evaluation of execution time and classification accuracy on six different datasets trained on the same platform. These results are essential to assess the proposed architecture's effectiveness and determine its potential applications in real-world scenarios. The Adam optimizer was used during the training process due to its suitability for handling models with a large number of parameters [43]. The number of epochs was set to 100, and the implicit regularization of the model's greater depth and smaller convolution filter sizes, along with pre-initialization of certain layers, allowed the networks to converge in fewer epochs [13].

TABLE III
Performance Analysis Based on Execution Time

No.	Datasets	Execution Time (h)			
		A1	A2	A3	Proposed Method
1	Brain Tumor	0:17:47	0:17:05	0:09:35	0:06:13
2	Cloth	0:46:05	0:39:07	0:31:35	0:19:27
3	Forest	0:20:59	0:14:44	0:11:36	0:06:56
4	Road	1:21:49	1:11:01	0:43:17	0:26:26
5	Rome Weather	0:24:07	0:22:50	0:12:39	0:08:36
6	Room	0:25:05	0:21:42	0:12:34	0:09:05

Table 3 is a comparison table that showcases the execution time performance of the proposed improved VGG architecture in comparison to three other existing architectures. The table is used to evaluate the effectiveness and efficiency of the proposed architecture in terms of its execution time. Comparing the execution time of the proposed method with other architectures can help determine the proposed architecture's potential practical applications and scalability.

Fig. 9 demonstrates that the proposed architecture's execution time is shorter for each dataset compared to other architectures, and this concludes that the execution time is 35.7% faster than the standard VGG16. The reduction in the number of parameters used in the proposed architecture is the primary reason for its compact execution, making it faster to execute than other VGG architectures. A model's speed is important in real-world applications because it can affect how quickly the model can provide output. Therefore, the proposed architecture's faster execution time is a crucial advantage in practice.

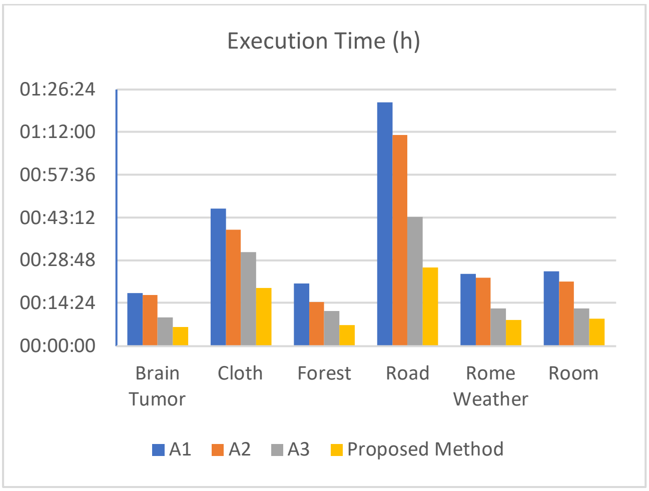

Fig. 9 The execution time (h) obtained based on datasets

Additionally, the compactness of the proposed architecture can also reduce memory usage, which is beneficial in low-memory devices. The performance comparison of execution time in this study aligns with previous works, highlighting the importance of model size and parameter reduction for efficient deep-learning models [44]. Next, to validate the proposed architecture, the accuracy of each dataset was evaluated as the second metric, a popular metric in multi-class classification computed from the confusion matrix [45]. The comparison of the classification accuracy of the proposed improved VGG architecture with three existing architectures is shown in Table 4. This evaluation method has been widely used in various studies to validate the accuracy of deep learning models [28], [46], [47].

TABLE IV
Performance Analysis Based on Accuracy

No.	Datasets	Accuracy (%)			
		A1	A2	A3	Proposed Method
1	Brain Tumor	43	43	46	77
2	Cloth	34	58	56	59
3	Forest	56	56	62	82
4	Road	55	55	91	98
5	Rome Weather	12	12	26	64
6	Room	44	44	54	67

The study reveals that the proposed architecture outperforms other VGG architectures in terms of accuracy. It is observed that the proposed architecture has a 41.9% accuracy improvement compared to the standard VGG16. This is demonstrated in Fig. 10, which shows the comparison of the proposed architecture with other architectures.

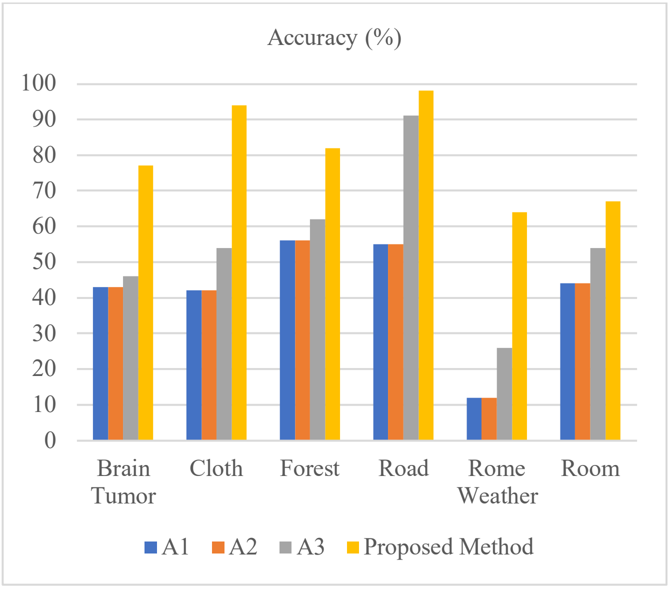

Fig. 10 The accuracy (%) obtained based on datasets.

The proposed method presented in this study reduces memory usage by decreasing the parameters to 1 066 146, compared to 134 268 738, 113 429 666, and 1 147 714 in architectures A1, A2, and A3, respectively. Despite the parameter reduction, this approach helps reduce network complexity and computational resources. By replacing the flattening layer with the GAP layer, the proposed architecture reduces the number of network parameters, which is 79% smaller in comparison with the standard VGG16, which contributes to shorter execution time. Throughout the training process, the proposed architecture shows better execution time results than A1, A2, and A3 architectures for all datasets. The proposed architecture outperforms the existing architectures for all six datasets for classification accuracy, achieving higher accuracy rates.

The reduction in the number of parameters in the proposed improved VGG architecture allows for more efficient computation and memory usage. The GAP layer replaces the flatten layer, which significantly impacts the number of parameters and execution time. These changes contribute to the proposed architecture's superior performance in terms of execution time and classification accuracy. Overall, the proposed improved VGG architecture offers a promising solution to the challenges of long execution times and excessive memory usage in VGG16 architecture. The results demonstrate that the proposed architecture outperforms existing architectures, achieving high accuracy rates with shorter execution times.

IV. Conclusion

The proposed method presented in this study offers a solution to the difficulties of VGG16, which is a very long execution time for image classification. The proposed architecture improves the network's efficiency and classification accuracy by reducing the number of layers and adding B.N. layers in feature extraction after every pooling layer. Moreover, using a GAP layer in the classifier followed by additional dense layers and dropout layers has reduced memory consumption, enhanced the network's robustness, and improved.

Based on the evaluation of six public datasets from Kaggle, the study's results demonstrate that the proposed architecture significantly reduces execution time while achieving better classification accuracy compared to three other existing architectures despite having fewer parameters. These findings suggest that the proposed architecture could be a promising solution for better image classification in practical applications.

Future research could expand on this study's findings by developing a web application that integrates the proposed model for classifying new images that are not included in the dataset. Such an application would provide researchers with a practical tool to improve image classification accuracy with less manual work, ultimately contributing to developing more efficient and effective image classification methods.

Furthermore, while the proposed architecture showed promising results in reducing parameters and improving classification performance, more investigation is needed to understand fully how these changes affect accuracy. Additionally, the study focused on image classification, and there's room to explore how the proposed architecture could be applied to other tasks like object detection or segmentation.

Acknowledgment

This research was supported by Ministry of Higher Education (MOHE) through Fundamental Research Grant Scheme (FRGS/1/2020/ICT02/UTHM/02/1).

References

[1] P. Patel and A. Thakkar, "The upsurge of deep learning for computer vision applications," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 1, 2020, doi: 10.11591/ijece.v10i1.pp538-548.

[2] K. Joshi, V. Tripathi, C. Bose, and C. Bhardwaj, "Robust Sports Image Classification Using InceptionV3 and Neural Networks," *Procedia Comput Sci*, vol. 167, no. Iccids 2019, pp. 2374–2381, 2020, doi: 10.1016/j.procs.2020.03.290.

[3] M. Mahdianpari, B. Salehi, M. Rezaee, F. Mohammadimanesh, and Y. Zhang, "Very deep convolutional neural networks for complex land cover mapping using multispectral remote sensing imagery," *Remote Sens (Basel)*, vol. 10, no. 7, 2018, doi: 10.3390/rs10071119.

[4] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4. 2018. doi: 10.1007/s13244-018-0639-9.

[5] Y. Wang and Z. Wang, "A survey of recent work on fine-grained image classification techniques," *J Vis Commun Image Represent*, vol. 59, 2019, doi: 10.1016/j.jvcir.2018.12.049.

[6] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif Intell Rev*, vol. 53, no. 8, 2020, doi: 10.1007/s10462-020-09825-6.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun ACM*, vol. 60, no. 6, 2017, doi: 10.1145/3065386.

[8] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017. doi: 10.1109/CVPR.2017.243.

[9] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015. doi: 10.1109/CVPR.2015.7298594.

[10] Y. Lecun, E. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," 1998.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, Dec. 2016, pp. 770–778. doi:10.1109/CVPR.2016.90.

[12] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," Feb. 2016, [Online]. Available: http://arxiv.org/abs/1602.07360

[13] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–14, Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.1556

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

[15] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014. doi: 10.1007/978-3-319-10590-1_53.

[16] Z. Song, L. Fu, J. Wu, Z. Liu, R. Li, and Y. Cui, "Kiwifruit detection in field images using Faster R-CNN with VGG16," in *IFAC-PapersOnLine*, Elsevier B.V., 2019, pp. 76–81. doi:10.1016/j.ifacol.2019.12.500.

[17] P. Hridayami, I. K. G. D. Putra, and K. S. Wibawa, "Fish species recognition using VGG16 deep convolutional neural network," *Journal of Computing Science and Engineering*, vol. 13, no. 3, pp. 124–130, 2019, doi: 10.5626/JCSE.2019.13.3.124.

[18] S. Theetchenya, S. Ramasubbareddy, S. Sankar, and S. M. Basha, "Hybrid approach for content-based image retrieval," *International Journal of Data Science*, vol. 6, no. 1. p. 45, 2021. doi:10.1504/ijds.2021.117467.

[19] S. S. Sawant *et al.*, "An optimal-score-based filter pruning for deep convolutional neural networks," *Applied Intelligence*, vol. 52, no. 15, 2022, doi: 10.1007/s10489-022-03229-5.

[20] I. Singh Walia, M. Srivastava, D. Kumar, M. Rani, P. Muthreja, and G. Mohadikar, "Pneumonia Detection using Depth-Wise Convolutional Neural Network (DW-CNN)," 2020, doi: 10.4108/.

[21] Zhongqin Bi, Ling Yu, Honghao Gao, Ping Zhou, and Hongyang Yao, "Improved VGG model-based efcient trafc sign recognition for safe driving in 5G scenarios.pdf." 2020.

[22] M. N. Islam *et al.*, "Diagnosis of hearing deficiency using EEG based AEP signals: CWT and improved-VGG16 pipeline," *PeerJ Comput Sci*, vol. 7, p. e638, 2021, doi: 10.7717/peerj-cs.638.

[23] A. Labach, H. Salehinejad, and S. Valaee, "Survey of Dropout Methods for Deep Neural Networks," Apr. 2019, [Online]. Available: http://arxiv.org/abs/1904.13310

[24] G. Chen, P. Chen, Y. Shi, C.-Y. Hsieh, B. Liao, and S. Zhang, "Rethinking the Usage of Batch Normalization and Dropout in the Training of Deep Neural Networks," May 2019, [Online]. Available: http://arxiv.org/abs/1905.05928

[25] D. Serdyuk, O. Braga, and O. Siohan, "Audio-Visual Speech Recognition is Worth $32\times 32\times 8$ Voxels," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2021 - Proceedings*, 2021. doi: 10.1109/ASRU51503.2021.9688191.

[26] Y. Zhou, H. Chang, Y. Lu, X. Lu, and R. Zhou, "Improving the Performance of VGG through Different Granularity Feature Combinations," *IEEE Access*, vol. 9, 2021, doi:10.1109/ACCESS.2020.3031908.

[27] F. Pasa, V. Golkov, F. Pfeiffer, D. Cremers, and D. Pfeiffer, "Efficient Deep Network Architectures for Fast Chest X-Ray Tuberculosis Screening and Visualization," *Sci Rep*, vol. 9, no. 1, Dec. 2019, doi:10.1038/s41598-019-42557-4.

[28] M. Mateen, J. Wen, Nasrullah, S. Song, and Z. Huang, "Fundus image classification using VGG-19 architecture with PCA and SVD," *Symmetry (Basel)*, vol. 11, no. 1, Jan. 2019, doi:10.3390/sym11010001.

[29] F. Zhao, B. Zhang, Z. Zhang, X. Zhang, and C. Wei, "Classification and detection method of Blood lancet based on VGG16 network," *2021 IEEE International Conference on Mechatronics and Automation, ICMA 2021*, pp. 849–853, 2021, doi:10.1109/ICMA52036.2021.9512686.

[30] H. Yang, J. Ni, J. Gao, Z. Han, and T. Luan, "A novel method for peanut variety identification and classification by Improved VGG16," *Sci Rep*, vol. 11, no. 1, 2021, doi: 10.1038/s41598-021-95240-y.

[31] Q. Yan, B. Yang, W. Wang, B. Wang, P. Chen, and J. Zhang, "Apple leaf diseases recognition based on an improved convolutional neural network," *Sensors (Switzerland)*, vol. 20, no. 12, pp. 1–14, 2020, doi:10.3390/s20123535.

[32] X. Li *et al.*, "Multi-Modal Multi-Instance Learning for Retinal Disease Recognition," in *MM 2021 - Proceedings of the 29th ACM International Conference on Multimedia*, 2021. doi:10.1145/3474085.3475418.

[33] B. Cui, X. M. Dong, Q. Zhan, J. Peng, and W. Sun, "LiteDepthwiseNet: A Lightweight Network for Hyperspectral Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, 2022, doi: 10.1109/TGRS.2021.3062372.

[34] E. Ovalle-Magallanes, N. G. Aldana-Murillo, J. G. Avina-Cervantes, J. Ruiz-Pinales, J. Cepeda-Negrete, and S. Ledesma, “Transfer Learning for Humanoid Robot Appearance-Based Localization in a Visual Map,” *IEEE Access*, vol. 9, pp. 6868–6877, 2021, doi:10.1109/ACCESS.2020.3048936.

[35] M. Lin, Q. Chen, and S. Yan, "Network In Network," Dec. 2013, [Online]. Available: http://arxiv.org/abs/1312.4400.

[36] P. P. Das, A. Acharjee, and Marium-E-Jannat, "Double coated VGG16 architecture: An enhanced approach for genre classification of spectrographic representation of musical pieces," in *2019 22nd International Conference on Computer and Information Technology, ICCIT 2019*, 2019. doi: 10.1109/ICCIT48885.2019.9038339.

[37] H. P. A. Tjahyaningtijas, A. K. Nugroho, C. V. Angkoso, I. K. E. Purnama, and M. H. Purnomo, "Automatic Segmentation on Glioblastoma Brain Tumor Magnetic Resonance Imaging Using Modified U-Net," *EMITTER International Journal of Engineering Technology*, vol. 8, no. 1, pp. 161–177, Jun. 2020, doi:10.24003/emitter.v8i1.505.

[38] M. M. Bejani and M. Ghatee, "A systematic review on overfitting control in shallow and deep neural networks," *Artif Intell Rev*, vol. 54, no. 8, 2021, doi: 10.1007/s10462-021-09975-1.

[39] T. D1Etterich, "Overfitting and Undercomputing in Machine Learning," *ACM Computing Surveys (CSUR)*, vol. 27, no. 3, 1995, doi:10.1145/212094.212114.

[40] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight sharing," in *The Mathematics Of Generalization*, 2018. doi:10.1162/neco.1992.4.4.473.

[41] D. M. Hawkins, "The Problem of Overfitting," *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 1. 2004. doi:10.1021/ci0342472.

[42] H. N. A. Pham and E. Triantaphyllou, "The impact of overfitting and overgeneralization on the classification accuracy in data mining," in *Soft Computing for Knowledge Discovery and Data Mining*, 2008. doi:10.1007/978-0-387-69935-6_16.

[43] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.6980

[44] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *36th International Conference on Machine Learning, ICML 2019*, 2019.

[45] M. Grandini, E. Bagli, and G. Visani, "Metrics for Multi-Class Classification: an Overview," Aug. 2020, [Online]. Available: http://arxiv.org/abs/2008.05756.

[46] M. Shu, "Deep Learning for Image Classification on Very Small Datasets Using Transfer Learning," 2019.

[47] K. S. Lee, S. K. Jung, J. J. Ryu, S. W. Shin, and J. Choi, "Evaluation of transfer learning with deep convolutional neural networks for screening osteoporosis in dental panoramic radiographs," *J Clin Med*, vol. 9, no. 2, Feb. 2020, doi: 10.3390/jcm9020392.