Journal of Theoretical and Applied Information Technology
15th January 2024. Vol.102. No 1
© 2023 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

THE USE OF CONVOLUTIONAL NEURAL NETWORK WITH EFFICIENTNET-B0 ARCHITECTURE IN BRAIN TUMOR CLASSIFICATION USING FLASK

[1] VINSENSIUS HARYO BHAKORO HADI [2]ACHMAD BENNY MUTIARA, [3]RINA REFIANTI
[1]Department of Informatics Engineering, Gunadarma University, Indonesia
[2,3]Faculty of Computer Science dan Information Technology, Gunadarma University, Indonesia
E-mail: [1]vincentius.haryo@gmail.com, [2,3]{amutiara,rina}@staff.gunadarma.ac.id

ABSTRACT

Brain tumors pose a significant health challenge, often evading early detection and resulting in fatal outcomes. Despite the assistance of imaging technologies like CT-Scan and MRI, achieving timely and accurate diagnoses remains a formidable task. This study advocates for the implementation of Convolutional Neural Networks (CNNs) to facilitate rapid and precise brain tumor detection. Specifically, we assess the performance of EfficientNet-B0, an advanced CNN architecture, in comparison to other CNN architectures for classifying brain tumors in MRI images. Our dataset comprises 3264 images across glioma, normal, pituitary, and meningioma classes. Testing involved various scenarios for epochs and optimizers, including Adam and RMSProp. Model testing results, analyzed through a confusion matrix, revealed an impressive average precision, recall, and F1-Score, all reaching 98%. The best-designed model accurately predicted glioma, normal, pituitary, and meningioma tumor types. Furthermore, the successful implementation of the classification model into a website using Python and the Flask framework signifies its potential for practical applications, enhancing accessibility and usability.

Keywords: *Convolutional Neural Network, EfficienNet-B0, Brain Tumor, Flask, Python.*

1. INTRODUCTION

Brain tumors are a highly lethal form of cancer that pose challenges in early detection. In Indonesia, brain tumors are prevalent and come in various types, including benign and malignant forms such as gliomas originating from brain cells [1]. Other types include meningiomas and pituitary tumors. Meningioma, the most common adult brain tumor, develops from the meninges the thin tissue layer covering the brain and spinal cord. This slow-growing tumor is typically benign but can rarely become malignant and spread to other body parts. Pituitary tumors, on the other hand, are non-cancerous growths occurring in the pituitary gland at the base of the brain they disrupt hormone production processes leading to imbalances.

Timely and accurate detection of brain tumors is imperative for effective treatment [2]. While conventional imaging technologies like CT-Scan and MRI contribute to detection, the challenges persist, mainly attributed to limitations in biopsy-based manual approaches concerning accuracy and efficiency. Previous approaches to brain tumor detection, often reliant on biopsy-based manual methods, have faced challenges in terms of accuracy and efficiency. The limitations of these methods underscore the need for innovative solutions. While some studies have explored the application of CNN architectures like VGGNet and ResNet, there remains a critical gap in understanding their performance in the context of diverse image sizes, a gap that the current research seeks to address by scrutinizing the capabilities of the EfficientNet-B0 architecture. The advent of Convolutional Neural Networks (CNNs) introduces a promising avenue for brain tumor image classification. Unlike traditional machine learning methods, CNNs employ deep learning techniques to automatically extract intricate features from images, thereby enhancing classification accuracy. Notable CNN architectures, including VGGNet, ResNet, and Inception, have demonstrated success in image processing and pattern recognition tasks.

An emerging method currently under development involves using Convolutional Neural Networks (CNNs) for classifying brain tumor images. CNNs leverage deep learning techniques to automatically extract complex features from images while improving classification accuracy compared to traditional machine learning methods. Various CNN architectures have demonstrated success in image processing and pattern recognition tasks these include VGGNet, ResNet, Inception among others.

However, despite these advancements, critical areas still demand attention. The existing literature highlights concerns regarding the adaptability and generalizability of CNN architectures to diverse medical imaging tasks and datasets. This research aims to address this gap by comparing the performance of the EfficientNet-B0 architecture with other CNN architectures in classifying MRI images of brain tumors. EfficientNet-B0 stands out as one of the latest architectures excelling at processing diverse image sizes while maintaining model training efficiency through its relatively low parameter count. The expected outcome is a deeper understanding of EfficientNet-B0's effectiveness relative to other CNN architectures when classifying MRI images of brain tumors. Additionally, the classification model utilizing EfficientNet-B0 will be integrated into a website using Python and Flask framework.

This implementation will enable users to upload MRI images for brain tumor detection, facilitating early diagnosis for both the general public and medical professionals alike. Such accessibility enhances information dissemination and raises awareness about brain tumors within the community.

2. RELATED WORKS

2.1 Artificial Intelligence

Artificial Intelligence is a combination of human and machine intelligence. In the world of Artificial Intelligence, algorithms have been designed to enable machines to complete the tasks assigned to them [3].

2.2 Machine Learning

Machine Learning is a collection of methods used to manage and forecast various data with the help of learning algorithms. In machine learning, computers can learn automatically from the given data. This allows the computer to improve its performance as more data becomes available [4].

2.3 Deep Learning

Deep learning is a type of machine learning that uses artificial neural networks to learn and recognize patterns in data. This technique allows machines to develop a deeper and more complex understanding of given data [5]. In this context, a neural network is a collection of nodes that mimic the structure of neurons in the brain of a living being. Each node in the network performs calculations by summing the weights of the input signals it receives. To prevent overfitting of the model, techniques such as DropOut are utilized. In the case of DropOut, the selection of deactivated units is done randomly [6].

2.4 Convolutional Neural Network

Convolutional Neural Network (CNN) is a fundamental structure within deep learning, consisting of multiple layers designed to deeply represent data. In the convolutional layer, crucial features are extracted from digital images, while the pooling layer reduces dimensionality and computation time. This architecture facilitates model regularization for enhanced performance. The extracted features are then directed into the softmax layer to initiate the classification process [7]. Convolutional Neural Network (CNN) consists of three layers: the input layer, output layer, and hidden layer sandwiched in between. The hidden layer typically encompasses convo- lutional layers, pooling layers, normalization layers, ReLU layers, fully connected layers, and loss layers [8]. Pooling layers maintain data size during convolution by downsampling, reducing the number of samples. Common pooling operations include max pooling, average pooling, and L2-norm pooling, each contributing to data size management [9].

2.5 EfficientNet

EfficientNet is a type of CNN architecture developed using depthwise and pointwise convolution techniques. In 2020, it was explained that EfficientNet-B0 is one of the architectural variants in the EfficientNet family that has the least number of parameters, around 5.3 million parameters [10].

3. RESEARCH METHODOLOGY

This research method employs the Software Development Life Cycle (SDLC), which includes stages of analysis, design, development, testing, and implementation. In the analysis stage, data processing is conducted to develop a Convolutional Neural Network (CNN) model with EfficientNet-B0 architecture using a dataset from Kaggle. The design stage encompasses the CNN model's design with various layers such as convolutional layer, activation layer, pooling layer, and fully connected layer. The development utilizes libraries such as NumPy, Pandas, Matplotlib, OpenCV, Keras, Tensorflow, and Flask as the web framework for deployment purposes. Subsequently, the model is tested with the same dataset and its performance is evaluated using evaluation methods like ac- curacy, presicion, and recall. Following that, the trained model will be implemented into a website using Python and Flask for brain tumor detection in MRI images.

4. RESULT & DISCUSSION

4.1 Data Collection

The dataset used in this study comprised a total of 3264 two-dimensional JPG images sourced from Kaggle. These images were categorized into four different classes: glioma tumors (826), normal tumors (395), meningioma tumors (822), and pituitary tumors (827) for the training set. A separate test- ing set consisted of additional samples from each class: glioma tumors (100), normal tumors (105), meningioma tumors (115), and pituitary tumors (74).

4.2 Preprocessing

After completing the data collection phase, the next step is preprocessing to facilitate feature extraction. In this study, the preprocessing process includes cropping and resizing. Cropping is used to remove noise in irrelevant image corners that are not relevant to the image label pattern, making it easier for extraction. Meanwhile, the resizing function adjusts the size of images to suit analysis needs both horizontally and vertically. The data splitting process divides the dataset into two parts: training data and testing data. A total of 3264 images were used in this study to develop a convolutional neural network model. Out of these, 2870 images were selected as samples for training the CNN model, while the remaining 394 images were used for performance evaluation. Data splitting was done randomly using functions from the Scikit-Learn library. This division aims to ensure the good generalization ability of the CNN model and its capability to handle problems in a generalized manner rather than making accurate predictions on specific samples only. Scaling the data involves resizing images from 150 x 150 pixels to 75 x 75 pixels. This is done using the OpenCV and NumPy libraries by reading image files with imread(), resizing them with resize(), and saving them as NumPy arrays. Data augmentation is performed on the training data to increase the sample size and prevent overfitting in the CNN model developed for brain tumor classification purposes. It involves steps like flipping, zooming, or rotation using the ImageDataGenerator library from Keras. It’s important during the scaling and augmentation processes to consider aspect ratios so that they don’t distort significantly, which can impact the CNN model’s performance.

4.3 Data Visualization

In Figure 1, the dataset of each class, namely glioma tumor, no tumor, meningioma tumor, and pituarity tumor. To display the images from the dataset, one sample from each class is taken randomly.

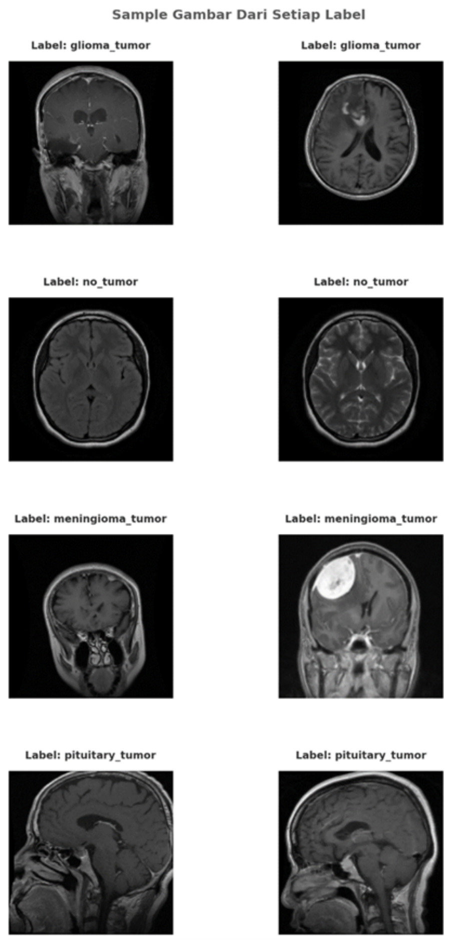


Figure 1: Data Visualization

4.4 Model Design

The input data consists of cropped and resized MRI brain tumor images with a size of 150 x 150 pixels. The initial stage involves a convolutional process using a 3 x 3 filter, resulting in 32 output channels, followed by batch normalization and ReLU activation. In this phase, depthwise convolution with a 3 x 3 filter generates 16 channels, followed by batch normalization, ReLU activation, global average pooling, reshaping, two convolutions with 3 x 3 filters and 16 channels, multiplication, and subsequent batch normalization. The subsequent stage performs convolutional operations using a 3 x 3 filter, yielding 24 output channels, followed by batch normalization, ReLU activation, and the MBConv1 process with a 3 x 3 filter. Continuing, a convolutional operation with a 5 x 5 filter produces 40 channels, followed by batch normalization, ReLU activation, another MBConv1 process with a 5 x 5 filter, and finally, dropout. In the next step, successive convolution operations are applied to the previous input. Using a 3 x 3 filter, 80 output channels are generated, followed by batch normalization and ReLU activation. An MBConv1 process with a 3 x 3 filter follows. Similarly, a sequence of convolutions is applied, utilizing a 5 x 5 filter, resulting in 112 channels. Batch normalization, ReLU activation, an MBConv1 process with a 5 x 5 filter, and dropout are subsequently applied. Moving forward, a convolutional operation with a 5 x 5 filter produces 192 output channels. Batch normalization, ReLU activation, and an MBConv1 process with a 5 x 5 filter are employed, followed by dropout. A 3 x 3 filter convolution is conducted, generating 320 output channels, with batch normalization and ReLU activation. Another MBConv1 process with a 3 x 3 filter follows. The final steps involve a 1 x 1 filter convolution, resulting in a one-dimensional output array or 'flatten'. Batch normalization and ReLU activation are applied during this step. The output from the previous step undergoes pooling to obtain a one-dimensional flattened array. Subsequently, the flattened result is fed into a neural network to obtain weight values. The ultimate stage involves classification using softmax activation on the output obtained from the preceding steps.

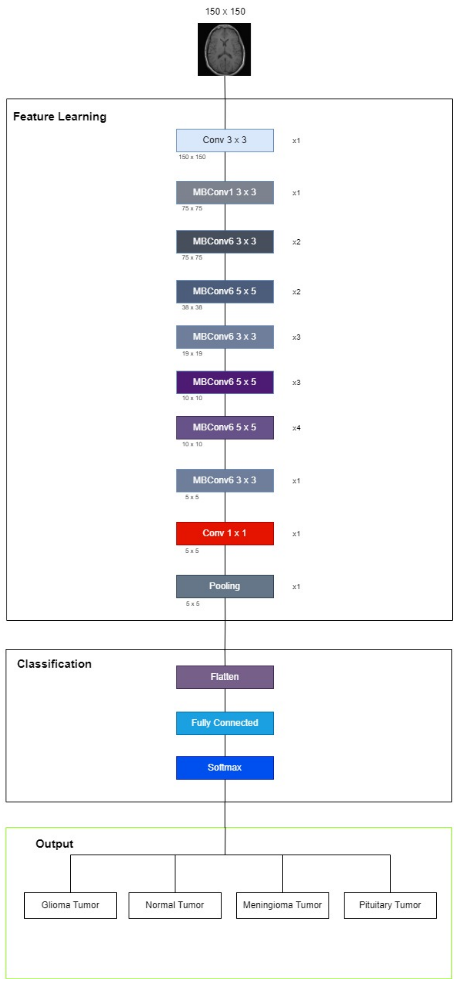
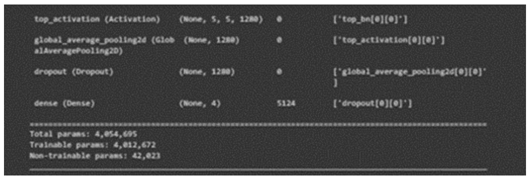


Figure 2: EfficientNet-B0 Architecture

4.5 Create Model

Imported the EfficientNet-B0 module from the keras.applications package. Created an effnet object using the EfficientNet-B0 architecture. Initialized the EfficientNet-B0 object using pre-trained from the ImageNet dataset and did not use a fully connected layer at the end of the model and determined the image dimensions to be used in model training. Defined the model variable as the output of the effnet object. Added the GlobalAveragePooling2D layer after the model output. Implemented a dropout layer by partially disabling the units by 0.5. Added an output layer with 4 units and used a softmax activation function to generate output class probabilities. Build a hard model using tf.hard.models. Models with inputs from effnet.input and outputs from model variables. Then the next model summary is done to see the total parameters can be seen in Figure 3.



```
top_activation (Activation)    (None, 5, 5, 1280)  0        ['top_bn[0][0]']

global_average_pooling2d (Glob  (None, 1280)       0        ['top_activation[0][0]']
alAveragePooling2D)

dropout (Dropout)              (None, 1280)        0        ['global_average_pooling2d[0][0]'
                                                            ]

dense (Dense)                  (None, 4)           5124     ['dropout[0][0]']

==================================================================================================
Total params: 4,054,695
Trainable params: 4,012,672
Non-trainable params: 42,023
__________________________________________________________________________________________________
```

Figure 3: Model Summary

4.6 Model Evaluation Results

Training results that have obtained accurate results based on training on validation data. This value is used to determine the level of classification success of the training model. Confusion matrix is a method that is usually used to calculate accuracy. The accuracy of a model is measured using several confusion matrix measurements which include measurement, precision, recall, accuracy, and F1. These mea- surements require the values of TFN, TFP, TNN, and TNP obtained from the prediction results with the actual class confusion matrix. The calculation results of the EfficientNet- B0 architecture that have been saved are then displayed in the form of a confusion matrix as shown in Figure 4.

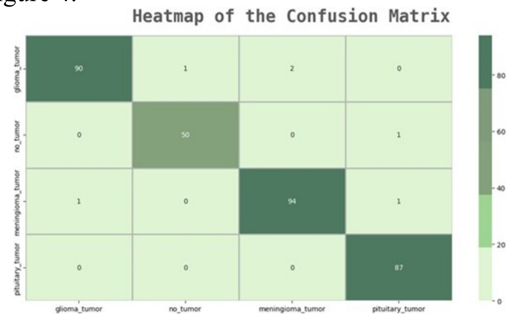
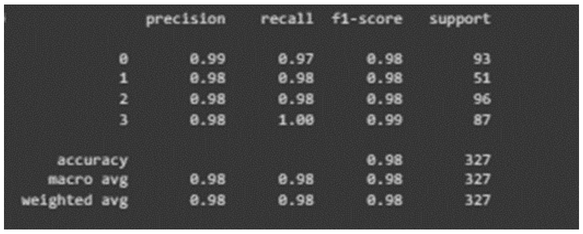


Figure 4: Confusion Matrix EfficientNet-B0



```
              precision    recall  f1-score   support

           0       0.99      0.97      0.98        93
           1       0.98      0.98      0.98        51
           2       0.98      0.98      0.98        96
           3       0.98      1.00      0.99        87

    accuracy                           0.98       327
   macro avg       0.98      0.98      0.98       327
weighted avg       0.98      0.98      0.98       327
```

Figure 5: EfficientNet-B0 Summary Results

Based on the summary results above, it is shown that the precision results of glioma are 99%, normal 98%, meningioma 98%, pituitary 98%. Recall glioma is 97%, normal 98%, meningioma 98%, pituitary 100% and f1 glioma is 98%, normal 98%, meningioma 98%, pituitary 99%. In this study based on the results of the architecture comparison can be seen in table 1.

Table 1: Architecture Comparison Results

No	Architecture	Precision	Recall	F1-Score
1	VGG16	91%	92%	91%
2	InceptionResnetV2	97%	98%	97%
3	EfficientNet-B0	98%	98%	98%

From the results of the comparison, the EfficientNet-B0 architecture is the architecture with the best results of the two architectures that have been compared. In the early stages of testing the optimizer. Tests carried out in the form of testing methods using Adam and RMSProp.

Table 2: Testing the Optimizer

Optimizer	Accuracy	Loss	Time
Adam	99.79%	0.6567	158ms/step
RMSProp	99.76%	0.5996	163ms/step

Based on the tests carried out to determine the optimizer, it was selected based on the highest accuracy value produced, the lowest number of losses, and the fastest time using Adam. Next, determine the optimum number of epochs. The epoch value is determined by conducting trials by entering epoch values of 15, 20, 25 and 30.

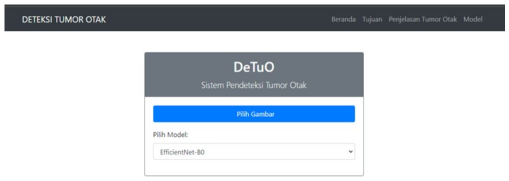
Table 3: Testing the Epoch

Optimizer	Epoch	Accuracy	Loss	Time
Adam	15	99.65%	1.1526	161ms/step
Adam	20	99.55%	0.7977	160ms/step
Adam	25	99.79%	0.6567	158ms/step
Adam	30	99.76%	0.7106	162ms/step

Based on the experiment, the best result of this research is using EfficientNet-B0 architecture using Adam optimizer with batch size value is 32 and the number of epochs is 25 which can produce the highest model accuracy of 99.79%.

4.7 Implementation

In the implementation process for brain tumor classifica- tion, which is based on a website that uses the flask framework. The resulting model will be used in the process of classifying the type of brain tumor on the website, there are 3 choices of architectural models. Users can later choose the architecture they want to use, the three architectures also have different levels of accuracy.



Pilih Model: EfficientNet-B0

Figure 6: Landing Page

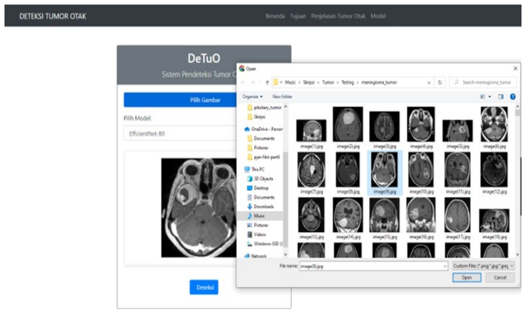
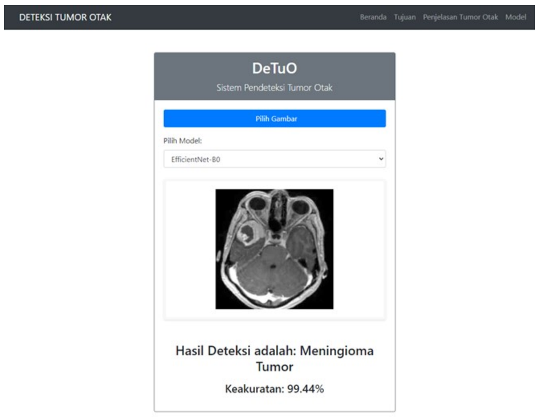


Figure 7: Choose File


DETEKSI TUMOR OTAK
Beranda Tujuan Penjelasan Tumor Otak Model
DeTuO
Sistem Pendeteksi Tumor Otak
Pilih Gambar
Pilih Model:
EfficientNet-B0

Hasil Deteksi adalah: Meningioma Tumor
Keakuratan: 99.44%

Figure 8: Image Detection Result

In Figure 6 is the initial display of the website on the home menu, on that display there is a select image to select an image of a brain tumor image and can also select the model. Figure 7 selects the image you want to detect and then a detection button will appear which will bring up the output results. Figure 8 after detection will appear the detection results and also the level of accuracy or confidence value in the form of a percentage.

5 CONCLUSION

Based on the results of the research that has been done, it can be concluded that the Convolutional Neural Network (CNN) method has been successful in predicting glioma, normal, pituitary, and meningioma tumor types on MRI images. The dataset used is 3264 data with details of 2870 training data, 394 test data, and in this training also takes validation data from training data by 10%. The model uses layers, namely convolutional layer, batch normalization layer, depthwise layer, activation layer, zero padding layer and ReLu activation function. For the training and testing process using a batch size of 32, epoch of 25, learning rate of 0.001, Adam optimizer, reduce learning rate and the results of all experimental scenar- ios with the confusion matrix method get an average precision value of 98%, average recall reaches 98%, and the average F1-Score value reaches 98%. Model training with EfficienNet- B0 architecture on brain tumor MRI that produces the highest accuracy value of the 2 architectures that have been compared. A website program for brain tumor type classification using Convolutional Neural Network (CNN) using Flask framework and EfficientNet-B0 architecture has been successfully created. This program can predict the type of brain tumor from the image with good accuracy.

REFRENCES

[1] Nugroho, D. (2020). Tumor Otak: Penyebab, Gejala, dan Pengobatan, Jakarta: Penerbit Gramedia.
[2] Green, R., Smith, J., Jones, D., & Miller, A. (2021). Survival and functional outcomes for patients with brain tumors detected at early stages: A systematic review and meta-analysis, Neuro Oncology, 23(1), 1-10.
[3] Jakhar, D., & Kaur, I. (2019). Artificial intelligence, machine learning & deep learning: Definitions and differences, Clinical and Experimental Dermatology, 45(1), DOI:10.1111/ced.14029.
[4] Danukusumo, K.P. (2017). Implementasi Deep Learning Menggunakan Convolutional Neural Network untuk Klasifikasi Citra Candi Berbasis GUI, Skripsi. Universitas Atma Jaya Yogyakarta.
[5] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning, Nature, 521(7553), 436-444.
[6] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2014). Dropout: A simple way to prevent neural networks from overfitting, Journal of machine learning research, 15(1), 1929-1958.
[7] You, Y., Liu, Z., & Ma, Y. (2017). A deep learning approach for text classification, IEEE Transactions on Knowledge and Data Engineering, 29(12), 2685-2696.
[8] Alom, M. M., Islam, M. M., Hasan, M. M., Hossain, M. A., & Ray, P. K. (2018). A novel convolutional neural network for image classification, Neural Computing and Applications, 30(11), 3375-3386.
[9] Zafar, A., Arshad, A., Nawi, N.M.M., Aamir, M. (2022). A Comparison of Pooling Methods for Convolutional Neural Networks, Applied Sciences,12(17):8643,DOI:10.3390/app12178643.
[10] Kizrak, M. A., Tahaoglu, S., & Yildirim, E. A. (2020). A Comparative Study of Deep Learning Algorithms on Image Classification Tasks, International Journal of Computer Theory and Engineering, Vol.12(4), pp. 22-26.