EKSAKTA
Journal of Sciences and Data Analysis

E-ISSN: 2720-9326
P-ISSN: 2716-0459

DOI : 10.20885/EKSAKTA.vol6.iss2.art6

# Hyperparameter Optimization of an Image Classification Model for Beef and Pork Using Convolutional Neural Networks

**Salsabila [1,*], Anwar Fitrianto [2], Bagus Sartono[3]**

[1] Digital business department Universitas Negeri Makassar
[2,3] Statistics and Data Science Department IPB University

[*] *Corresponding author:salsabila@radenintan.ac.id*

**Abstract:** Deep learning classification network in one case, has different classification capabilities than the network in another. The classification method of deep learning using CNN has specific hyperparameters that can be adjusted to have good performance. These hyperparameters include the number of convolutional layers, the number of neurons in the convolutional and fully connected layers, kernel size, and activation functions. Deep Learning uses experimental principles in finding the best hyperparameter in various cases. The model architecture can be determined by choosing a different design. This research uses pork and beef images as the data for classification using CNN. The abstract textures of beef and pork may make it difficult for the CNN classification model to distinguish between them. Hence, 32 combinations of five hyperparameters were compared. It was found that these hyperparameters affect the model's performance. The best model has obtained 98,7% accuracy that uses 20 neurons both layers of the convolution was, kernel size of 5 × 5, ReLU activation function, and two fully connected layers with dropout 0.7 as a method of overfitting prevention. A significant difference also occurs in the application of the activation function, in which ReLU has a better performance than *tanh* function to increase the model's prediction.

**Keywords:** Classification, Deep Learning, CNN, Model, Hyperparameter.

## Introduction

The application of deep learning varies across different cases, each influenced by several critical factors that affect model performance. One of the key factors is the volume of data used for training—generally, a larger dataset enhances the learning capability of the model. However, model performance is not solely determined by data size; the configuration of network components, known as hyperparameters, also plays a crucial role in determining the effectiveness of deep learning models [1].

Convolutional Neural Network (CNN) is a widely adopted deep learning method for image classification tasks due to its customizable and hierarchical architecture. CNNs rely on various hyperparameters, such as the number of convolutional layers, the number of neurons in each layer, kernel size, and activation functions. These hyperparameters must be carefully tuned to minimize the loss function and improve learning efficiency. An appropriate hyperparameter configuration significantly influences model complexity, learning capacity, and convergence rate, which ultimately determines classification performance [2].

Since its development, CNN has continuously evolved through the integration of new architectural components and computer vision techniques [3]. A CNN architecture can be built using diverse design strategies, making it challenging to determine the most effective configuration [4]. Consequently, empirical comparisons among different models and hyperparameter settings are often used to identify optimal designs.

E-ISSN: 2720-9326
P-ISSN: 2716-0459
DOI :

There is no universally optimal hyperparameter setting for all learning problems. Instead, deep learning requires empirical experimentation to find the most suitable configuration for each specific task. Previous studies have investigated the effect of regularization techniques, such as Dropout, on baseline models. In the specific case of binary image classification—distinguishing pork from beef images—a CNN model using dropout with a rate of 0.7 achieved a test accuracy of approximately 97.6%, with AUC and F1-Score values of 0.996 and 0.975, respectively, and demonstrated the lowest error rate among compared models [5]. More recently, a dropout rate of 0.5 was applied to an Inception-ResNet-V2 architecture, achieving 96.5% accuracy for pork-beef classification with optimal learning rates [6]. Additionally, strategic placement of dropout layers in ResNet-50 architectures was shown to improve generalization and reduce overfitting in complex visual classification tasks [7]. Building upon these empirical findings, the present study aims to explore diverse hyperparameter categories to derive an optimized CNN model for pork and beef image classification.

## Materials and Methods

### Materials

This study uses image data of beef and pork collected by photographing the meat with a smartphone camera. Images of both meat types were obtained from the Surya Kencana traditional market in Bogor. Approximately 90 % of the dataset consists of shots of 250 grams portions of fresh pork and beef tenderloin, which were cut into several pieces and photographed repeatedly from different angles under ambient light. The remaining 10 % comprises images of rib and thigh cuts that were photographed intact, also from multiple angles. Altogether, 3000 images were collected. Of these, 15 % (450 images) were designated as the test set, while the remaining 2550 images served as the training set.

### Hyperparameter Optimization in CNN

CNN implements a more special Artificial Neural Network (ANN) and is considered the best method for image recognition cases. CNN architecture consists of three layers: the convolution layer, the pooling layer, and the fully connected layer [8]. **Figure 1** shows an architecture of CNN that uses two convolutions (C1, C2) and pooling (P1, P2) layers, and two fully connected (FC) layers.

There are two main components of the CNN method, feature extraction and classifier. The feature extraction is at the convolution and pooling layers, while the classifier is at the fully connected layer. The image processing techniques for feature extraction which concentrate on extracting colour, texture and shape features from input pre-processed images [9], while the classifier is used for model learning and find the corresponding label for every test image [10].

E-ISSN: 2720-9326
P-ISSN: 2716-0459
DOI :

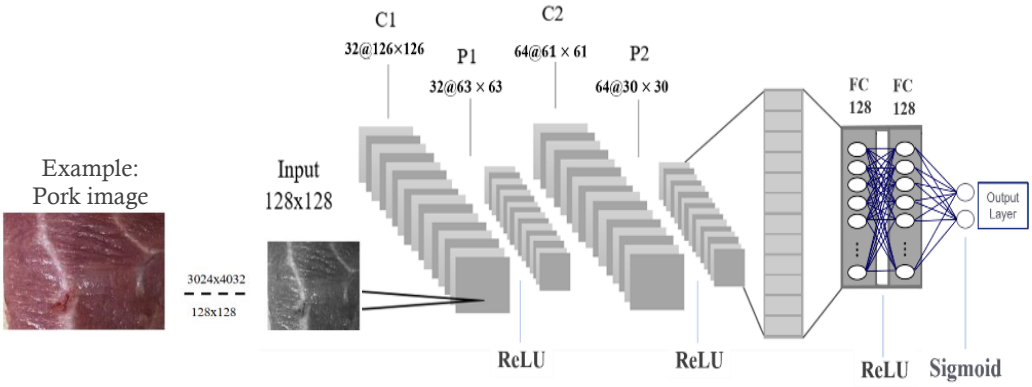



**Figure 1.** CNN architecture

Several studies have been carried out for the model optimization process by applying hyperparameter settings to the CNN algorithm. Research in comparing eight hyperparameter groups in the Convolutional Neural Network algorithm [4]. Then, the research used the Natural Language Processing (NLP) algorithm to compare 12 hyperparameter categories [11]. This research considers the number of convolutional layers, the number of neurons, the activation function, the size of the kernel, and the number of fully connected layers.

1. The number of convolution layers

In CNN architecture, a feature extraction process aims to identify patterns or image features that belong to the image. The process is in a convolutional layer. The convolutional layer consists of the processes of convolution, activation, and unification. A CNN network is typically constructed with one or more convolution layers and pooling layers [12]. However, the many layers of convolution cannot guarantee that the model has the potential to work properly.

2. The number of hidden layers

There are no specific rules in selecting the number of convolutional layers or the number of hidden layers. However, finding a CNN network that uses more than two hidden layers [2]. Although increasing the number of hidden layers can improve the model's performance, it will increase the computing power to run the network and thus take more time to train the network.

3. The number of neurons

The selection of the number of neurons greatly affects the network from the occurrence of overfitting and underfitting. If too many neurons are used in the hidden layer, this will result in overfitting. Conversely, if too few neurons are used, the possibility of underfitting is greater [2].

4. Activation function

In neurons, the activation function is used to calculate the number of weighted inputs and bias. It is responsible for determining whether the neuron is active or not [13]. In other words, the activation function can act as a threshold function in neurons.

Several activation functions commonly used in deep learning are sigmoid, *tanh*, and ReLU functions. However, the ReLU function is often used by many researchers in deep learning. Deep learning networks that use ULT are easier to optimize than networks with sigmoid or tanh, this is because the positive value generated makes the gradient more stable [14].

5. Kernel size

E-ISSN: 2720-9326
P-ISSN: 2716-0459
DOI :

In the convolution layer, the convolution process is understood by moving the kernels $f$ $m \times n$ in an $X$ image sized $i \times j$, then adding the multiplication of values in the image and kernel [1]. Each movement of the kernel is called a stride. The kernel and stride size affects the size of the feature map from the convolution process. The larger the kernel size and the number of steps, the smaller the feature map size. However, the small kernel size does not necessarily result in a feature map representing patterns in the original image. Therefore, several kernel sizes can be compared to get the best classification model performance.

**Model Evaluation**

During the training process using training data, the performance of the process is evaluated by test data. If the results obtained are not as expected, then an improvement is needed either by using different hyperparameters, different data partitions or by changing the CNN structure using both data until an acceptable result is obtained. If the test data turns out to be unacceptable, repairs can be made again using training data [1].

1. Confusion Matrix

Confusion Matrix is a metric that can accurately evaluate classification models [1].

**Table 1**. Confusion matrix

| | Actual Labels | |
|---|---|---|
| | 0 | 1 |
| Predicted Labels | *True Positive* (TP) | *False Positive* (FP) |
| | *False Negative* (FN) | *True Negative* (TN) |

Based on the value of the confusion matrix, several important values are used to measure the model's goodness. Some of them are the values for accuracy, sensitivity, specificity, precision, and F1.

- Accuracy

  Accuracy in a confusion matrix is the probability for classification class values to be classified correctly with all the values from that class being classified correctly or not.

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

- Recall and Precision

  Accuracy and recall are two very important quantitative measures in a classification model.

$$\text{Precision (P)} = \frac{\text{TP}}{\text{TP+FP}} \qquad (2)$$

$$\text{Recall (R)} = \frac{\text{TP}}{\text{TP+FN}} \qquad (3)$$

The precision is for the calculations on the prediction of the class is positive, whereas the calculation of the actual class is positive. In general, precision describes any accuracy of the prediction result when the class prediction is positive. Meanwhile, considering the probability value for the acquisition of a positive class in the actual sample is accurately predicted as a positive class.

2. Area Under the Curve (AUC) Score

The Area Under the Curve (AUC) serves as a standard metric for assessing the effectiveness of a classification model, particularly in terms of its ability to differentiate between distinct classes. This metric corresponds to the area beneath the Receiver Operating Characteristic (ROC) curve, which illustrates the relationship between the true positive rate (sensitivity) and the false positive rate (1-specificity) across various threshold values. AUC values can fluctuate between 0 and 1, with

E-ISSN: 2720-9326
P-ISSN: 2716-0459
DOI :

values approaching 1 signifying superior model performance. The computation of AUC can be performed using the trapezoidal rule as:

$$\int_0^1 TP(FP)d(FP) \qquad (4)$$

An AUC of 0.5 indicates that the model's performance is equivalent to random guessing, while an AUC of 1.0 indicates perfect classification [15].

3. F1-Score

One quantitative measure that can combine the precision and recall or their harmonization is the F1 Score:

$$\text{F1-Score} = \frac{2\,\text{PR}}{\text{P+R}} \qquad (5)$$

or

$$\text{F1-Score} = \frac{2\,\text{TP}}{2\text{TP+FP+FN}} \qquad (6)$$

The F1-score is in the range 0 and 1, if the F1-score is equal to 1, the classification obtained is perfect. In practice, the evaluation of the classification model idea using the F1 score is carried out on the test data and can be corrected until a satisfactory F1 score is achieved [1].

## Result and Discussion

### Image Classification Model Performance Beef and Pork Based on Hyperparameter Comparison

In identifying the optimal CNN architecture, the researcher systematically compared a range of hyper-parameter configurations—such as learning rate, batch size, and the number of convolutional filters—while classifying pork and beef images. As shown in Table 2, this comparison highlighted three top-performing models (the 3rd, 7th, and 12th), which posted the lowest loss values and the highest accuracy, AUC, and F1-scores among all models evaluated.

E-ISSN: 2720-9326
P-ISSN: 2716-0459
DOI :

**Table 2**. Summary of model performance based on hyperparameter comparison

| Model | Convolution Layer | Number of Neuron | Kernel Size | Activation Function | Fully Connected Layer | Accuracy | Loss | AUC | F1-Score |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 *CL* | V1 | 3 × 3 | Tanh | 2 *FCL* | 0.947 | 0.556 | 0.976 | 0.969 |
| 2 | 2 *CL* | V2 | 3 × 3 | Tanh | 3 *FCL* | 0.942 | 0.506 | 0.970 | 0.927 |
| 3 | 2 *CL* | V1 | 3 × 3 | ReLU | 2 *FCL* | 0.971 | 0.085 | 0.996 | 0.973 |
| 4 | 2 *CL* | V2 | 3 × 3 | ReLU | 3 *FCL* | 0.976 | 0.101 | 0.995 | 0.986 |
| 5 | 2 *CL* | V1 | 5 × 5 | Tanh | 2 *FCL* | 0.951 | 0.456 | 0.980 | 0.932 |
| 6 | 2 *CL* | V2 | 5 × 5 | Tanh | 3 *FCL* | 0.956 | 0.328 | 0.992 | 0.929 |
| 7 | 2 *CL* | V1 | 5 × 5 | ReLU | 2 *FCL* | 0.987 | 0.097 | 0.996 | 0.982 |
| 8 | 2 *CL* | V2 | 5 × 5 | ReLU | 3 *FCL* | 0.971 | 0.158 | 0.993 | 0.961 |
| 9 | 2 *CL* | V3 | 3 × 3 | Tanh | 2 *FCL* | 0.949 | 0.394 | 0.983 | 0.939 |
| 10 | 2 *CL* | V4 | 3 × 3 | Tanh | 3 *FCL* | 0.958 | 0.368 | 0.982 | 0.948 |
| 11 | 2 *CL* | V3 | 3 × 3 | ReLU | 2 *FCL* | 0.971 | 0.137 | 0.995 | 0.969 |
| 12 | 2 *CL* | V4 | 3 × 3 | ReLU | 3 *FCL* | 0.978 | 0.097 | 0.995 | 0.978 |
| 13 | 2 CL | V3 | 5 × 5 | Tanh | 2 *FCL* | 0.964 | 0.319 | 0.985 | 0.949 |
| 14 | 2 *CL* | V4 | 5 × 5 | Tanh | 3 *FCL* | 0.949 | 0.442 | 0.989 | 0.932 |
| 15 | 2 *CL* | V3 | 5 × 5 | ReLU | 2 *FCL* | 0.951 | 0.239 | 0.987 | 0.943 |
| 16 | 2 *CL* | V4 | 5 × 5 | ReLU | 3 *FCL* | 0.980 | 0.149 | 0.994 | 0.995 |
| 17 | 3 *CL* | V5 | 3 × 3 | Tanh | 2 *FCL* | 0.942 | 0.293 | 0.989 | 0.927 |
| 18 | 3 *CL* | V6 | 3 × 3 | Tanh | 3 *FCL* | 0.958 | 0.290 | 0.981 | 0.952 |
| 19 | 3 *CL* | V5 | 3 × 3 | ReLU | 2 *FCL* | 0.962 | 0.200 | 0.989 | 0.948 |
| 20 | 3 *CL* | V6 | 3 × 3 | ReLU | 3 *FCL* | 0.980 | 0.142 | 0.994 | 0.978 |
| 21 | 3 *CL* | V5 | 5 × 5 | Tanh | 2 *FCL* | 0.962 | 0.374 | 0.979 | 0.964 |
| 22 | 3 *CL* | V6 | 5 × 5 | Tanh | 3 *FCL* | 0.953 | 0.425 | 0.985 | 0.940 |
| 23 | 3 *CL* | V5 | 5 × 5 | ReLU | 2 *FCL* | 0.973 | 0.278 | 0.985 | 0.986 |
| 24 | 3 *CL* | V6 | 5 × 5 | ReLU | 3 *FCL* | 0.971 | 0.280 | 0.988 | 0.961 |
| 25 | 3 *CL* | V7 | 3 × 3 | Tanh | 2 *FCL* | 0.958 | 0.231 | 0.994 | 0.948 |
| 26 | 3 *CL* | V8 | 3 × 3 | Tanh | 3 *FCL* | 0.962 | 0.336 | 0.988 | 0.956 |
| 27 | 3 *CL* | V7 | 3 × 3 | ReLU | 2 *FCL* | 0.973 | 0.203 | 0.994 | 0.969 |
| 28 | 3 *CL* | V8 | 3 × 3 | ReLU | 3 *FCL* | 0.973 | 0.160 | 0.995 | 0.969 |
| 29 | 3 *CL* | V7 | 5 × 5 | Tanh | 2 *FCL* | 0.958 | 0.282 | 0.992 | 0.940 |
| 30 | 3 *CL* | V8 | 5 × 5 | Tanh | 3 *FCL* | 0.949 | 0.419 | 0.988 | 0.932 |
| 31 | 3 *CL* | V7 | 5 × 5 | ReLU | 2 *FCL* | 0.967 | 0.332 | 0.989 | 0.965 |
| 32 | 3 *CL* | V8 | 5 × 5 | ReLU | 3 *FCL* | 0.964 | 0.250 | 0.990 | 0.945 |

Notes:

CL : Convolution Layer
FCL: Fully Connected Layer
V1 : 20-20-128-128
V2 : 20-20-128-128-128
V3 :32-64-128-128
V4 : 32-64-128-128-128
V5 : 20-20-20-128-128
V6 : 20-20-20-128-128-128
V7 : 32-64-64-128-128
V8 : 32-64-64-128-128-128

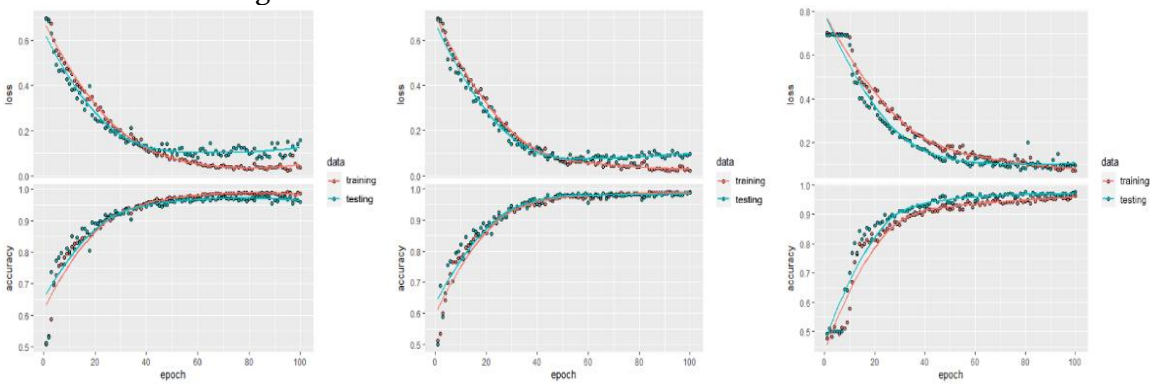
This table lists the accuracy, loss, AUC, and F1-score for 32 CNN models with varying convolutional depths, kernel sizes, activation functions, and fully connected layers used to classify beef versus pork images.

## Model Evaluation

Overfitting is general issue in machine learning, which cannot be completely avoided [16], but it can be reduced. **Figure 2** shows that the 3rd and 7th models appear to have a plot that tends

E-ISSN: 2720-9326
P-ISSN: 2716-0459
DOI :

to be more stable than the 12th model. However. the 7th model is known to have a small gap between the training and test data loss than the 3rd model. This is very important because it relates to the effect of overfitting.




(a) (b) (c)

**Figure 2.** Classification Metric Plot, (a) 3rd model, (b) 7th model, (c) 12th model

Overall performance clusters tightly, but the 7th model—the two-layer network using V1 filters, 5 × 5 kernels, ReLU activation, and two fully connected layers—stands out with the top accuracy of 0.987, a minimal loss of 0.097, an AUC of 0.996, and an F1-score of 0.982. Accordingly, this configuration is selected as the best model for classifying beef and pork images.

There is one hyperparameter that significantly affects model performance, namely the activation function. Models that apply the Tanh activation function have poor model performance. Almost all models that use the Tanh activation function have low accuracy, AUC, and F1-Score and high loss. The following is a plot to see the distribution of loss values in the model based on the activation function.

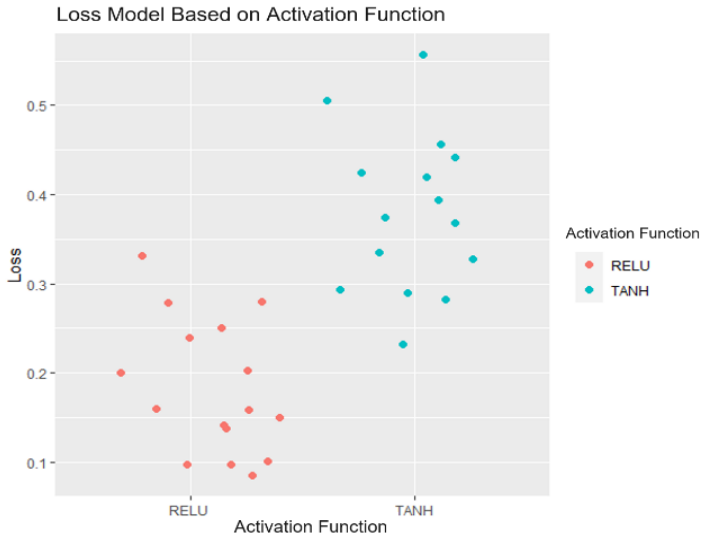



**Figure 3.** Distribution of loss in the model based on activation function

**Figure 3** shows that there is a significant difference in the distribution of loss models based on the type of activation function. The loss distribution in models that use the ReLU activation function tends to be lower than in models that use the Tanh activation function.

E-ISSN: 2720-9326
P-ISSN: 2716-0459
DOI :

The 7th model is a model consisting of a hyperparameter with two convolutional layers. 20 neurons in both convolution layers and 128 neurons in both fully connected layers, a 5 × 5 kernel size, ReLU activation function, and two fully connected layers. Based on this model, here is a confusion matrix that shows the number of classes that are classified correctly and incorrectly.

**Table 3.** Image classification results of beef and pork using the 7th model

| Prediction | Reference | |
|---|---|---|
| | Beef | Pork |
| Beef | 223 | 4 |
| Pork | 2 | 221 |

**Table 3** shows that from 450 test data, 223 beef images were classified correctly and 221 images of pork were classified correctly.

## Conclusion

Based on the comparison of several hyperparameter configurations, this study confirms that hyperparameter tuning significantly influences the performance of CNN models for pork and beef image classification. The best-performing model was achieved using 20 neurons in both convolutional layers, 128 neurons in each of the fully connected layers, a 5×5 kernel size, and the ReLU activation function. This configuration successfully classified 98.7% of the test data correctly. The selection of the best hyperparameters still relies on experimental techniques. A suggestion for future research is to utilize or develop an automated hyperparameter optimization method that can effectively determine the optimal hyperparameters for building a CNN model.

These findings also highlight the importance of optimizing architectural parameters in developing efficient food classification systems. The results have practical implications for food authentication, quality control, and halal verification processes. Future research may expand on this work by incorporating more diverse datasets, exploring transfer learning strategies, or evaluating model robustness under real-world conditions such as variable lighting and image noise.

## Acknowledgment

The author thanks Mr. Dr. Anwar Fitrianto, M.Sc. and Dr. Bagus Sartono, M.Sc. and all parties who have helped so that this research can be completed.

## References

[1] H. H. Aghdam, E. J Heravi, Convolutional Neural Network. A Guide to Convolutional Neural Network (a practical application to traffic-sign detection and classification), Spain: Springer, 2017, pp.106-118.
[2] SH. Chon, “Hyper-parameter optimization of a convolutional neural network,” M.S. thesis, Department of Operations Research, Air Force Institute of Technology, Nigeria, 2019.
[3] S. Neha, M. Vibhor, Anju, An analysis of convolutional neural networks for image classification, Procedia Comput Science, 2018, pp.337–384.

E-ISSN: 2720-9326
P-ISSN: 2716-0459
DOI :

[4] NM. Aszemi, D. Durai, Hyperparameter optimization in convolutional neural network using genetic algorithms, International Journal of Advanced Computer Science and Applications, vol.10, no.7, pp. 269–278, 2019.
[5] Dewi RS, Rohman F, Agustin R, Arifianto MS. Comparative Analysis of CNN Model to Classify Pork and Beef using Regularization. *J RESTI*. 2021;5(3):436–43. doi:10.29207/resti.v5i3.2969
[6] Harahap AA, Damanik W, Harahap B. Classification of Pork and Beef using Inception ResNet V2 and CNN. *Jurnal Mantik*. 2024;8(2):549–56.
[7] Wijaya Y, Sutrisno E, Nugroho AS. Optimization of Dropout Placement in ResNet-50 Architecture for Improved Image Classification. *TELKOMNIKA Telecommunication Computing Electronics and Control*. 2024;22(1):45–52.
[8] M.S. Junayed, A.J. Afsana, T.A. Syeda, N. Nafis, K. Asif, A. Sami, and S. Bharanidharan, AcneNet - A Deep CNN Based Classification Approach for Acne Classes, 12th International Conference on Information & Communication Technology, and System, 2019, pp. 203-208.
[9] K. Nanditha, K. Nagamani, Understanding and Visualization of Different Feature Extraction Processes in Glaucoma Detection. Journal of Phys.: Conf. Ser. 2327 012023, 2022.
[10] M. Alaslani, L. Elrefaei, Convolutional neural network based feature extraction for iris recognition, International Journal of Computer Science & Information Technology, Vol., No.2, 2018, pp. 65-78.
[11] A. Aghaebrahimian and M. Cieliebak, Hyperparameter tuning for deep learning in natural language processing, Proceedings of the 4th edition of the Swiss Text Analytics Conference, Vol. 2458, 2019.
[12] M. Sadeghi et al., PERSIANN-CNN: Precipitation estimation from remotely sensed information using artificial neural networks–convolutional neural networks, Journal Hydrometeorol., vol. 20, no. 12, pp. 2273–2289, 2019.
[13] C. Nwankpa, W. Ijomah, A. Gachagan, S. Marshall, Activation functions: comparison of trends in practice and research for deep learning, https://arxiv.org/pdf/2010.07359.pdf, (Accessed Jan. 11, 2021).
[14] P. Ramachandran, B. Zoph, Q.V. Le, Searching for activation functions. arXiv preprint, https://arxiv.org/pdf/1710.05941.pdf, (Accessed Jan. 11, 2021).
[15] T. Fawcett, An introduction to ROC analysis, *Pattern Recognition Letters*, 27(8), 861–874, 2006.
[16] Ying, An overview of overfitting and its solutions. Journal of Physics. 1168(2). 022022, 2019.