Jurnal Ilmiah KURSOR
Menuju Solusi Teknologi Informasi
Vol. 12, No. 4, Desember 2024

ISSN 0216 – 0544
e-ISSN 2301– 6914

SKIN RASH CLASSIFICATION SYSTEM USING MODIFIED DENSENET201 THROUGH RANDOM SEARCH FOR HYPERPARAMETER TUNING

[a]Fayza Nayla Riyana Putri, [b]R.Rizal Isnanto, [c]Aris Sugiharto

[a]Master of Information Systems Program, School of Postgraduate Studies, Diponegoro University, Semarang 50275, Indonesia
[b]Departement of Computer Engineering, Diponegoro University, Semarang 50275, Indonesia
[c]Department of Informatics, Diponegoro University, Semarang 50275, Indonesia
E-mail: fayzanaylaa@gmail.com

Abstract

Skin rashes caused by various diseases, such as monkeypox, cowpox, chickenpox, measles, and HFMD, often present similar symptoms, making accurate diagnosis challenging. This study aims to improve the classification of skin diseases through the application of a modified DenseNet-201 architecture combined with hyperparameter optimization using Random Search. The base DenseNet-201 model, with pre-trained weights, was first tested, achieving an accuracy of 63%, with the highest performance in the Healthy and HFMD classes. The proposed modified model, optimized using Random Search, improved overall accuracy to 80%, with enhanced precision, recall, and F1-score across most classes. The model's performance was particularly notable in the HFMD and normal skin classes, although further improvements are needed for challenging classes like Cowpox and Measles. The findings highlight the potential of Random Search for hyperparameter tuning to enhance the performance of deep convolutional neural networks in the medical image classification domain, offering a promising tool for efficient and accurate skin disease detection.

Key words: Classification, DenseNet-201, Hyperparameter, Random Search, Skin Rashes.

INTRODUCTION

Skin rashes are a common medical condition caused by various factors, including viral infections, allergies, and autoimmune diseases. Symptoms of skin rashes often resemble those of different diseases, making accurate diagnosis challenging based on clinical observation alone [1]. For example, according to Mande et al. (2022), several diseases diseases like monkeypox, chickenpox, cowpox, Hand, Foot, and Mouth Disease (HFMD), and measles often present with similar rashes, complicating the differentiation among them [2].

Currently, diagnosing diseases through skin rashes involves clinical examination by medical professionals, followed by laboratory confirmation using diagnostic techniques like Polymerase Chain Reaction (PCR) [3]. While this approach ensures accurate diagnosis, it can be time-consuming and depends on access to laboratory facilities, which may not be available in all regions. Delays in detection can lead to delays in treatment, potentially increasing the risk of infection spread within the community [3].

An innovative solution to address these challenges is the application of Deep

Convolutional Neural Networks (Deep CNNs). This technology offers the ability to classify diseases by automatically analyzing images of skin rashes. Unlike regular images, lesion images represent abnormal skin rashes indicative of diseases such as monkeypox, whereas normal images lack such abnormal visual cues [3]. Deep Convolutional Neural Networks (Deep CNNs) are characterized by a greater number of convolutional layers [4]. Similar to standard CNNs, Deep CNNs also incorporate pooling layers to reduce data dimensions and activation layers (such as ReLU) to enhance network efficiency [5]. Deep CNN, with its deep convolutional layers, can extract complex features from images and recognize patterns that might be difficult to capture with traditional methods [6]. This not only enhances detection speed but alsFo can expand access to early diagnosis, especially in areas with limited medical facilities.

The implementation of Deep CNN in classifying images of skin rashes has already begun in the healthcare field, including previous research conducted by Bala et al. (2023), which demonstrated success in applying Deep CNN for multiclass skin dataset classification, achieving the highest accuracy rates of 93.91% on the original dataset and 98.91% on the augmented dataset in detecting four classification classes: normal skin and skin with diseases such as measles, chickenpox, and monkeypox [7].

Although previous research results show promising accuracy rates in classifying skin diseases, the hyperparameter optimization process in that research employed a trial-and-error approach, which often faces several limitations. While sometimes effective, this approach often requires a long time and does not always guarantee the optimal combination of hyperparameters. This drawback emphasizes the need for a more systematic and efficient approach to hyperparameter optimization.

Therefore, this research aims to build upon and refine existing efforts by focusing on the application of more sophisticated hyperparameter optimization methods, namely Random Search. Random search is an approach for hyperparameter tuning that selects samples randomly from the search space, resulting in significant computational cost savings [8].

This research not only focuses on optimizing the Deep CNN model in classifying types of skin rashes for monkeypox but also for chickenpox, measles, and normal skin. This study will also compare the effectiveness of these two methods in optimizing model performance.

Random Search offers flexibility in exploring a vast hyperparameter space without getting stuck in local solutions [9]. Therefore, this methods is chosen for this study.

This study builds upon and refines existing efforts, focusing on implementing a more advanced hyperparameter optimization method, namely Random Search. The research not only aims to optimize the Deep CNN model for classifying monkeypox, chickenpox, cowpox, HFMD, and measles but also to compare the effectiveness of hyperparameter tuning methods in enhancing model performance. In this study, hyperparameters will be optimized within the DenseNet architecture, a variant of Deep CNN known for its ability to retain information through dense connections between layers. DenseNet improves the feed-forward characteristics of the network and strengthens the flow of information through CNN layers [7].

DenseNet-201 is chosen for this study due to its dense connectivity structure, where each layer is connected to every other layer, which significantly improves information flow and gradient propagation throughout the network. This architecture helps in mitigating the vanishing gradient problem and allows for better feature reuse, making it an ideal choice for complex image classification tasks, such as the detection of monkeypox and other skin conditions.

However, despite the strengths of DenseNet-201, further modification and hyperparameter tuning are necessary to enhance its performance for this specific task. By utilizing Random Search for hyperparameter optimization, the model's learning rate, dropout rate, and other parameters can be fine-tuned to avoid overfitting, improve convergence speed, and boost overall accuracy. Given the variability in skin conditions and the need for precise classification, fine-tuning these parameters can help the model adapt better to the complexities and nuances of medical images. Additionally, optimizing these hyperparameters helps in balancing the trade-off between model complexity and computational efficiency, which is crucial in real-time medical applications where both accuracy and speed are vital.

MATERIAL AND METHODS

This study utilizes input data comprising images of normal skin and various skin diseases, including chickenpox, measles, cowpox, HFMD, and monkeypox. The input data is sourced from Kaggle and undergoes a structured series of processing stages. The first step involves data collection, where the dataset is divided into two main parts: Training Data and Testing Data, with an 80:20 split. Subsequently, the training data is further partitioned using stratified k-fold cross-validation to ensure balanced class distribution across all folds.

Following data partitioning, a preprocessing stage is conducted to prepare the data for model training. This preprocessing includes normalization and data transformation to enhance the quality of inputs for the model. Additionally, data augmentation techniques are applied to expand and enrich the training dataset by introducing variations in image position and color, thereby improving the model's generalization ability.

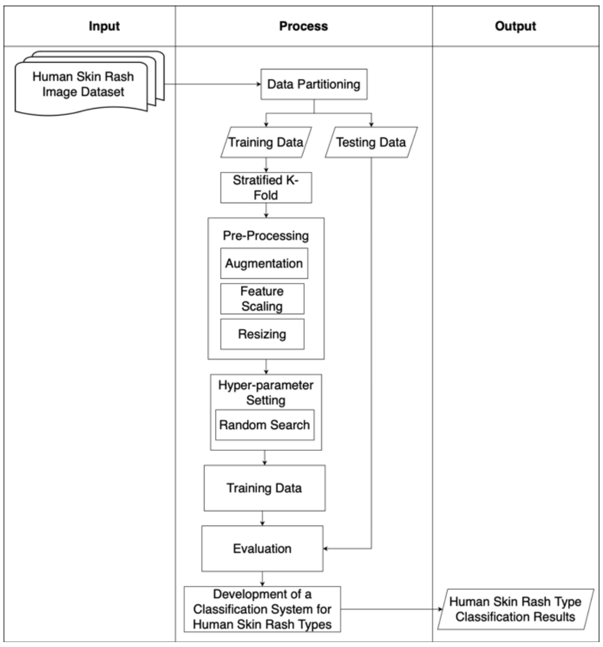


Fig 1. Research procedure

After preprocessing, the study focuses on hyperparameter tuning to identify the optimal configuration for the model. This process is carried out using the Random Search method to efficiently enhance model performance. The selected model architecture is DenseNet, a type of Deep Convolutional Neural Network (Deep CNN) known for its dense connections, which help maintain the flow of information between layers. With its ability to automatically extract hierarchical features, the Deep CNN model is expected to perform accurate classification of human skin diseases.

Once the model is trained, the final stage involves evaluation. The model's performance is assessed using the testing data based on metrics such as accuracy, precision, recall, and F1-score. This systematic approach aims to develop a skin disease classification model that is not only accurate but also reliable for application in various medical contexts.

The research procedure encompasses all the steps used in a study to achieve research objectives. The research procedure can be represented as shown in Figure 1.

Data Collection

The data in this study is sourced from a research data repository, namely Mendeley Data. The dataset used is called the Mpox Skin Lesion Dataset Version 2.0 (MSLD v2.0), which includes 75 images of chickenpox, 55 images of measles, 284 images of monkeypox, 66 images of cowpox, 161 images of HFMD, and 114 images of normal skin. This dataset was developed by Ali et al. (2023). All data are pre-labeled as Normal Skin Data, Chickenpox Data, Monkeypox Data, Cowpox Data, HFMD Data, and Measles Data, facilitating analysis and processing. These labels serve as identifiers for each image to be analyzed in this study. The data will be processed using Deep Learning methods, particularly Deep CNN, with the DenseNet architecture as the base model.

Data Partitioning

After completing the data collection stage, the next step involves splitting the dataset into two main groups: training data and testing data. In this study, the dataset is divided with a proportion of 80% for training data and 20% for testing data. This step ensures that the model receives sufficient data for training while reserving a separate set of completely unseen data to evaluate the overall performance of the model.

The split is performed prior to implementing the stratified k-fold cross-validation method.

The training data derived from this initial partition is subsequently used in the validation process through stratified k-fold, which aims to iteratively train the model while maintaining a balanced class distribution within each fold. This approach allows the model to learn more effectively from the training data without compromising the balance of information across classes.

Meanwhile, the testing data is separated at the beginning and remains unused during both the training and validation processes. This data is reserved exclusively as unseen data for the final stage of the study. Its primary role is to serve as the main evaluation tool for the model's actual performance, ensuring that the assessment reflects the model's ability to recognize patterns in completely unfamiliar data. By adopting this approach, the model is expected to demonstrate strong performance and a high degree of generalization when applied to real-world cases, such as in skin disease classification.

Stratified K-Fold Cross Validation

After dividing the dataset into training and testing data, the training data undergoes further processing using the Stratified K-Fold Cross-Validation method. In this study, the method is implemented with a 5-fold partition. Each fold is divided into two subsets: training data and validation data. The validation data is proportionally sampled from the training data to ensure that class distributions remain balanced in each fold.

The Stratified K-Fold Cross-Validation process involves splitting the training data into five equal parts. In each iteration, one part is used as validation data, while the remaining four parts serve as training data. This process is repeated five times, ensuring that each portion of the training data acts as validation data exactly once. With this approach, all training data contribute equally to the training process while also being used for intermediate evaluation on the validation set.

The primary role of validation data in this method is to provide an interim assessment of the model's performance during training. By utilizing a small portion of the training data that is excluded from the main training iterations, this study can measure and analyze the model's performance in greater detail. These evaluations help detect potential issues such as overfitting or underfitting early on, enabling adjustments to optimize the model's performance before conducting final testing on the unseen testing data. This strategy is expected to produce a more accurate model with improved generalization capabilities for classifying skin diseases

Data Preprocessing

After completing the stratified k-fold cross-validation stage, the preprocessing phase is carried out through several key steps, including data augmentation, feature scaling, and resizing. This process aims to prepare the data for training and testing the model. The complete workflow of these preprocessing steps is illustrated in Figure 2.

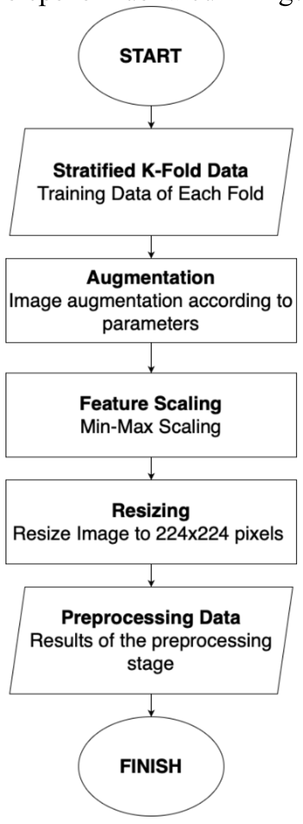

START
Stratified K-Fold Data
Training Data of Each Fold
Augmentation
Image augmentation according to parameters
Feature Scaling
Min-Max Scaling
Resizing
Resize Image to 224x224 pixels
Preprocessing Data
Results of the preprocessing stage
FINISH

Fig 2. Preprocessing data

The next step is data augmentation. In training deep learning models, having a large dataset is crucial to prevent overfitting and enhance accuracy [11].

Augmentation generates new variations from the existing data by changing attributes such as position and color, thereby increasing the diversity of the dataset. As seen in the research conducted by Bala et al. (2023) [7], data augmentation parameters can be applied as shown in Table 1.

Table 1. Augmentation Parameters

Parameters	Definition
Rotation Range	Input data is created by rotating from -45 to 45 degrees
Horizontal Flip	Randomly flip the input data horizontally
Vertical Flip	Randomly flip the input data vertically
Zoom Range	Perform zoom in or out from the center by a factor of 0.8 to 1.25 and randomly rotate the image by 45 degrees
Brightness Variation Range	Randomly adjust the brightness of the image by a factor of 0.1 to 2
Contrast Range	Randomly adjust the contrast of the image by a factor of 0.5 to 2
Jitter Hue	Randomly adjust the hue of the image by a factor of 0.5
Jitter Saturation	Randomly adjust the brightness of the image by a factor of 0.2 to 3
Fill Mode	Fill the gaps with black pixel values

In this study, there is an imbalance in the initial amount of data in each class. Through the data augmentation process, it is expected to create a balance between the amount of data in each class, allowing the resulting model to be more effective and able to address the existing data imbalances.

Hyperparameter Settings

After the data partitioning stage, the focus shifts to the Hyper-parameters Setting of the model. This setting aims to find the optimal combination of hyperparameter values that can enhance the model's performance. Several hyperparameters to be adjusted include dropout rate, learning rate, filters, optimizer, batch size, number of epochs, as well as parameters related to the model architecture, such as the number of layers and filters in the convolutional layers. For instance, the learning rate will be set to determine the extent to which the model responds to changes during training, while the batch size will influence how many samples are used in each training iteration. The number of epochs, on the other hand, will indicate how many times the entire dataset will be processed by the model. The dropout rate will be configured to reduce overfitting by randomly ignoring a number of units in the network layers during training, making the model more robust to new data.

The Hyper-parameters Setting process involves exploring various combinations of values to find the best configuration that can enhance the model's performance on the validation data. In this study, a comparison will be made across different trials in the random search for hyperparameter tuning. The validation accuracy from each hyperparameter combination in each trial will be compared, and the combination with the highest validation accuracy will be selected as the hyperparameters for modifying DenseNet-201. There will be five trials of the random search implementation as hyperparameter tuning in this research.

DenseNet-201 Model

After obtaining the best parameter values in the previous step, the next phase is to implement the Deep Convolutional Neural Network (CNN) Model using DenseNet-201 architecture as the foundation. DenseNet-201 offers an advantage through its dense connectivity structure, where each layer is connected directly to all subsequent layers. This dense connection facilitates better information and gradient flow throughout the network, leading to improved feature extraction from input images.

The DenseNet-201 architecture is shown in Figure 3. It begins with a convolutional layer followed by a pooling layer. The architecture consists of four dense blocks, each separated by transition layers. Each dense block contains several convolutional layers (6, 12, 24, and 16 layers, respectively), and these blocks perform a series of convolutions using 1x1 and 3x3 filters to extract various features from the input images. The transition layers between the dense blocks include batch normalization, a 1x1 convolution, and a 2x2 average pooling operation, which help reduce the spatial dimensions while maintaining the depth of the feature maps. The architecture concludes with a global average pooling layer, followed by a fully connected layer with a softmax activation for classification.

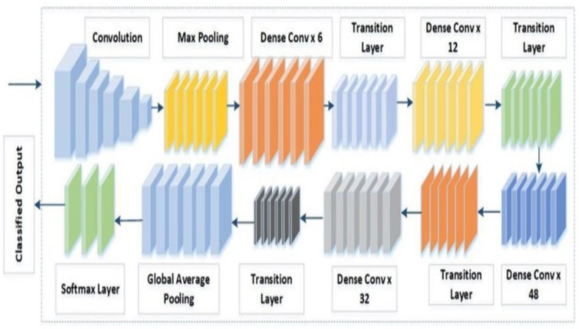


Fig 3. DenseNet-201 architecture

In the model implementation process, feature extraction focuses on two main characteristics: color and texture of skin rash images, particularly for chickenpox, monkeypox, measles, and normal skin.

The model implementation process involves training the model using the previously prepared training data. During training, validation data is used to evaluate how well the model performs and to avoid overfitting or underfitting issues. Once the model achieves good convergence, testing can be conducted on the testing data to assess the model's accuracy in making predictions on unseen data [12]. In subsequent steps, the model will be modified based on the best-found hyperparameters by using Random Search, optimizing its performance and refining its classification accuracy.

Model Evaluation

After implementing the model, its performance is evaluated using a Multiclass Confusion Matrix. A Confusion Matrix is a table that records the performance of a classification model [13].

The Multiclass Confusion Matrix provides values regarding how well the model can correctly classify positive cases (skin diseases) and negative cases (normal skin). From the results of the Multiclass Confusion Matrix, various metrics such as accuracy, precision, recall, and F1 score can be calculated to present a holistic picture of the model's performance. This evaluation helps in understanding how well the model can recognize skin diseases and guides necessary improvement steps.

Human Skin Rash Classification System

After completing the model evaluation, the next step is to design a simple system capable of classifying new images, providing important information for diagnosis and further medical intervention.

The aim of this system is to facilitate testing the model. This is realized through a visually simple user interface, allowing the classification results to be displayed clearly and understandably. This visual interface is expected to assist users in evaluating the model's performance and viewing the classification results of human skin rashes. With this approach, the system not only provides accurate classification results but also ensures practicality and usability in the model testing and evaluation process.

RESULT AND DISCUSSION

Based on the data collected, representative samples were taken from each class of the dataset. These samples included images from six classes: monkeypox, chickenpox, measles, cowpox, HFMD and normal skin. Then sample of image is shown in Figure 4.

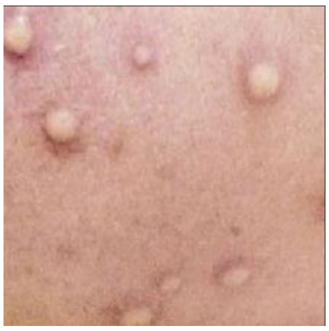


Fig 4. Sample of image

After the data collection phase is complete, the next step is to perform data partitioning. In this study, the data is divided with a ratio of 80:20, where 80% of the total data is used for training and the remaining 20% is used as test data.This partitioning is aimed at ensuring that the model has sufficient training data to learn from, while also having test data as unseen data to evaluate the model's performance after the training is completed. Table 2 provides a detailed breakdown of the data distribution for each class, illustrating how the training and testing data are allocated for each category.

Table 2. Data Partitioning

Label Data	Training Data	Testing Data
Chickenpox	60	15
Cowpox	53	13
Monkeypox	227	57
Measles	44	11
HFMD	129	32
Normal	91	23
Total Data	604	151

Table 3. Stratified K-Fold

Fold	Label Data	Training Data	Validation Data
1	Chickenpox	48	12
	Cowpox	43	10
	Monkeypox	181	46
	Measles	35	9
	HFMD	103	26
	Normal	73	18
2	Total Data	48	12
	Chickenpox	42	11
	Cowpox	182	45
	Monkeypox	35	9
	Measles	103	26
	HFMD	73	18
3	Normal	48	12
	Total Data	42	11
	Chickenpox	182	45
	Cowpox	35	9
	Monkeypox	103	26
	Measles	73	18
4	HFMD	48	12
	Normal	42	11
	Total Data	182	45
	Chickenpox	36	8
	Cowpox	103	26
	Monkeypox	72	19
5	Measles	48	12
	HFMD	43	10
	Normal	181	46
	Total Data	35	9
	Chickenpox	104	25
	Cowpox	73	18

After the training and testing data partitioning phase, the training data is processed using the Stratified K-Fold Cross-Validation method to ensure that the model is trained and evaluated optimally. In this study, the training data is divided into 5 equal folds. Each fold consists of two parts: training data and validation data, which are taken proportionally, ensuring that the class distribution remains balanced in each fold. Table 3 provides a detailed breakdown of the data for each fold, offering a deeper understanding of the distribution of training and validation data used in each iteration of Stratified K-Fold.

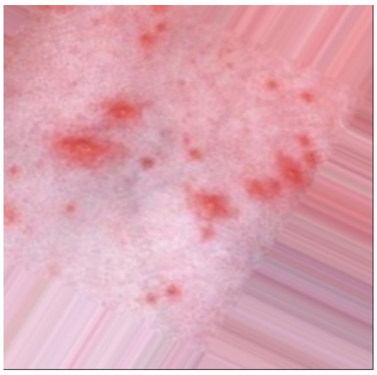


Fig 5. Sample of the augmented image

During training, each data batch is automatically processed to produce different images each time it is handled by the data generator. Table 4.4 shows the distribution of the initial data for each class, prior to dynamic augmentation. With this technique, augmentation not only adds variety but also maintains a balance in the number of samples for each class during the training process. Examples of the augmented images can be seen in Figure 5, demonstrating how these transformations enhance variation without altering the semantic representation of the data.

This augmentation process is integrated with the data preprocessing stage. Additionally, feature scaling was applied to normalize the pixel values of the images, ensuring that the model's learning process would not be influenced by varying scales in the input data. Min-Max scaling was used to transform each pixel value into the range of 0 to 1, making the data uniform for the model [14]. Subsequently, each image is resized to 224x224 pixels, aligning with the input dimensions required by the model's architecture.

After the preprocessing stage, the next step is to build a model using the basic DenseNet-201 architecture without hyperparameter tuning. This model utilizes default settings without any further adjustments to its parameters. Testing was performed to evaluate the model's performance with this standard configuration.

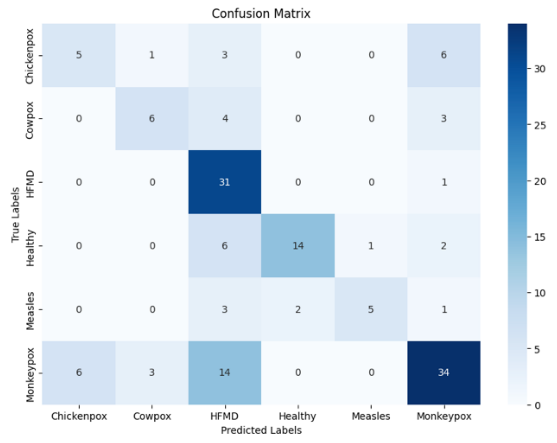
The results of the testing across five folds show varying accuracy for each fold. Table 4 presents the validation accuracy obtained from each fold.

Table 4. Validation Accuracy of Each Fold

Fold	Accuracy
1	0.67
2	0.70
3	0.67
4	0.67
5	0.71

Among the five folds, fold 5 achieved the highest accuracy, with a value of 71%. Therefore, the model trained with fold 5 was selected to proceed to the testing phase.

The testing results shown in Figure 6 reveal that the basic DenseNet-201 model achieved an overall accuracy of 63%, with an average precision value of 67%, recall of 57%, and F1-score of 59%. The best performance was achieved in the Healthy class, with the highest precision of 88% and an F1-score of 72%. The HFMD class also showed relatively good results, with a recall of 97%, although its F1-score was still considered low at 67%. On the other hand, performance in the Chickenpox and Measles classes was relatively poorer, especially in recall and F1-score.



```
Classification Report:
              precision    recall  f1-score   support

  Chickenpox       0.45      0.33      0.38        15
      Cowpox       0.60      0.46      0.52        13
        HFMD       0.51      0.97      0.67        32
     Healthy       0.88      0.61      0.72        23
     Measles       0.83      0.45      0.59        11
   Monkeypox       0.72      0.60      0.65        57

    accuracy                           0.63       151
   macro avg       0.67      0.57      0.59       151
weighted avg       0.67      0.63      0.62       151
```

Fig 6. Evaluation of basic DenseNet-201

Overall, while this model showed decent results for certain classes, its performance still needs to be improved in order to enhance accuracy and other evaluation metrics, especially for the more challenging classes. One potential approach to improve performance is hyperparameter tuning. In this study, the Random Search method was employed for hyperparameter tuning.

At this stage, the Random Search method is used to find the optimal hyperparameter combination for the model. This process involves evaluating different hyperparameter combinations randomly within the predefined search space. The hyperparameter search space is presented in Table 5, which includes parameters such as learning rate, dropout rate, batch size, number of filters, optimizer type, and number of epochs.

Table 5. Hyperparameter Initialization

Hyperparameter	Value Range
Learning rate	[0.0001, 0.001]
Dropout rate	[0.4, 0.5, 0.6, 0.7]
Batch size	[32, 64]
Filters	[256, 512, 1024]
Optimizer	[256, 512, 1024]
Epochs	[10]

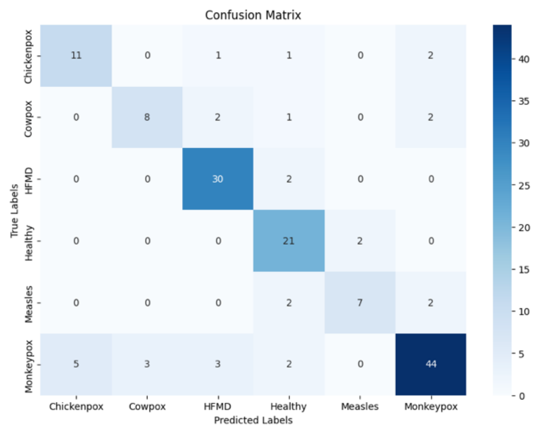
In this experiment, 10 trials are conducted for each fold during the cross-validation process. Therefore, each fold evaluates ten different hyperparameter combinations, providing ample opportunity to explore the search space. This method helps identify the best configurations that deliver optimal performance on the validation data.

Once all the folds are completed, Table 6 presents the best results obtained from each fold, including the best hyperparameter combinations discovered. This process ensures that the model receives hyperparameter settings that are not only suited for the training data but also improve generalization on the validation data.

From the trials that have been conducted, it is evident that in the trial on fold 4, the combination of hyperparameters used achieved the highest validation accuracy, reaching 90%. Therefore, this combination of hyperparameters was selected for use in the Deep CNN model with the DenseNet-201 architecture. This selection was based on the best performance shown in the evaluation, ensuring that the model has optimal generalization capability on unseen data.

Table 6. Best Trial Each Fold

Fold	Hyperparameter	Value	Accuracy
1	Learning rate	0.0001	0.88
	Dropout rate	0.7	
	Batch size	32	
	Filters	512	
	Optimizer	rmsprop	
	Epochs	10	
2	Learning rate	0.0001	0.81
	Dropout rate	0.7	
	Batch size	32	
	Filters	256	
	Optimizer	rmsprop	
	Epochs	10	
3	Learning rate	0.0001	0.82
	Dropout rate	0.4	
	Batch size	32	
	Filters	512	
	Optimizer	rmsprop	
	Epochs	10	
4	Learning rate	0.0001	0.90
	Dropout rate	0.4	
	Batch size	32	
	Filters	512	
	Optimizer	rmsprop	
	Epochs	10	
5	Learning rate	0.0001	0.82
	Dropout rate	0.4	
	Batch size	64	
	Filters	256	
	Optimizer	adam	
	Epochs	10	



```
Classification Report:
              precision    recall  f1-score   support

  Chickenpox       0.69      0.73      0.71        15
      Cowpox       0.73      0.62      0.67        13
        HFMD       0.83      0.94      0.88        32
     Healthy       0.72      0.91      0.81        23
     Measles       0.78      0.64      0.70        11
   Monkeypox       0.88      0.77      0.82        57

    accuracy                           0.80       151
   macro avg       0.77      0.77      0.76       151
weighted avg       0.81      0.80      0.80       151
```

Fig 7. Evaluation of modified DenseNet-201

Figure 7 shows the Confusion Matrix from the random search hyperparameter tuning. Based on the testing results, the model built with the best hyperparameter combination achieved an overall accuracy of 80%. The average evaluation metrics also indicate good performance, with precision of 81%, recall of 80%, and F1-score of 80%. The best performance was observed in the HFMD class, with the highest F1-score of 88%, indicating that the model was able to identify this class effectively. The normal skin class also showed good performance with an F1-score of 81%. However, performance on the Cowpox and Measles classes was relatively lower, particularly in terms of recall. Overall, the model demonstrated reasonably good capabilities in classifying data across various skin disease classes, although there is room for improvement, particularly in the more challenging classes.

By using the best hyperparameter combination, the model's performance was subsequently improved, which is further discussed in the following section. The results indicate that proper hyperparameter tuning, particularly through Random Search, plays a critical role in optimizing the performance of deep learning models like DenseNet-201 when applied to complex image classification tasks.

The main difference between the two DenseNet-201 architectures lies in the additional layers in the modified version. The default DenseNet-201 architecture consists of the base model followed by global average pooling and a softmax output layer for classification. In contrast, the modified version introduces extra layers: a convolutional layer with 3x3 filters, a max-pooling layer, and a dropout layer. These additions aim to enhance feature extraction, reduce overfitting, and refine the model's performance before the final output layer. Both versions ultimately use a softmax activation for multi-class classification but differ in their layer configurations and complexity.

In the modified model (Figure 8), a convolutional layer with 3x3 filters and 48 filters is added after the DenseNet-201 base model. This layer helps in refining the feature extraction process by applying additional convolution operations. Following this, a MaxPooling2D layer is used, which reduces the dimensions of the feature map by selecting the maximum value from each 2x2 patch. This operation helps retain the most important features and accelerates the training process. Additionally, a fully connected layer with 256 units and a dropout layer (with a rate of 0.4) are introduced to prevent overfitting and improve generalization. Finally, the model concludes with a softmax output layer for classification into 6 classes. Additionally, GlobalAveragePooling2D is employed, which calculates the average of all values in the feature map. This method produces one value per channel and serves to drastically reduce dimensionality, thereby enhancing the model's robustness to spatial variations in the images [15].

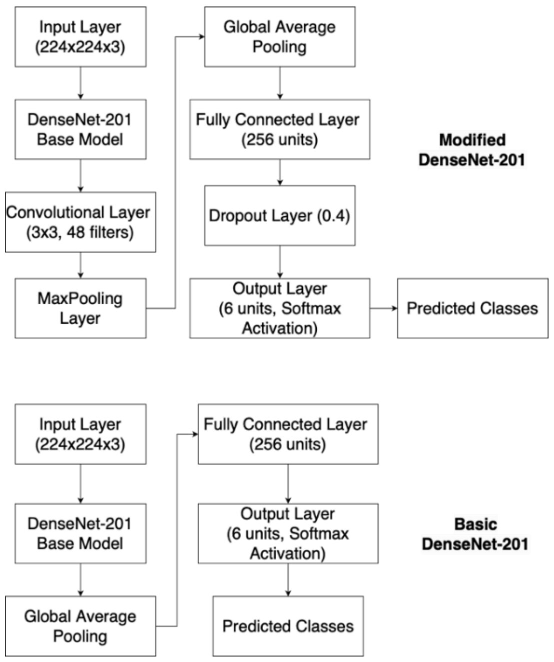
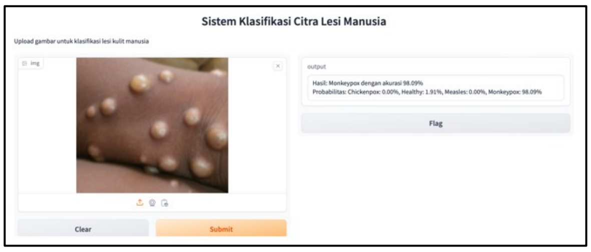


Fig 8. Architecture of the modified model

Compared to the basic DenseNet-201 architecture, these added layers contribute to a more detailed and robust feature extraction process, improving classification accuracy. Following the testing of the model, a simple system was developed for classifying types of human skin rashes. The system interface is shown in Figure 9, which includes a field for inputting images and displays the classification results along with the percentage likelihood of other classes.

Table 7 summarizes the comparison of the proposed model in this study with the modified DenseNet-201 architecture and the research conducted by Bala et al., who utilized the modified DenseNet-201 architecture.


Sistem Klasifikasi Citra Lesi Manusia

Upload gambar untuk klasifikasi lesi kulit manusia

Hasil: Monkeypox dengan akurasi 98.09%
Probabilitas: Chickenpox: 0.00%, Healthy: 1.91%, Measles: 0.00%, Monkeypox: 98.09%

Fig 9. Skin rash classification system

Table 7. Comparision With Previous Research

Research	**Method**	**Accuracy**
[7]	Modified DenseNet-201	93,91% (Original Image) and 98,91% (Augmented Image)
Basic Model	DenseNet-201	Overall accuracy of 63%.
Proposed Model	Modified DenseNet-201	Overall accuracy of 80%.

CONCLUSION

This study successfully demonstrated the application of hyperparameter optimization using Random Search on the DenseNet-201 architecture for classifying skin diseases, specifically monkeypox, chickenpox, measles, HFMD, and normal skin. Based on the testing results, the base DenseNet-201 model achieved an accuracy of 63%, with the best performance observed in the Healthy and HFMD classes. However, the performance in the Chickenpox and Measles classes still needs improvement, particularly in recall and F1-score.

On the other hand, the modified model with hyperparameter optimization through Random Search showed a significant improvement in performance, achieving an overall accuracy of 80%, with better precision, recall, and F1-score across all classes. The best performance was noted in the HFMD and normal skin classes, although performance in the Cowpox and Measles classes still requires further enhancement.

A comparison with previous research shows that the modified model in this study outperforms the basic DenseNet-201 model, but it is still below the performance of the DenseNet-201 model modified in Bala et al. (2023). Nevertheless, this study demonstrates that hyperparameter optimization using Random Search can improve the efficiency and performance of the model, making it an effective approach for enhancing the accuracy and precision of skin disease detection using Deep CNN.

REFERENCES

[1] R. A. Farahat *et al.*, "Human monkeypox disease (MPX)," *Infez Med*, vol. 30, no. 3, pp. 372–391, 2022, doi: 10.53854/LIIM-3003-6.

[2] G. Mande *et al.*, "Enhanced surveillance of monkeypox in Bas-Uélé, Democratic Republic of Congo: the limitations of symptom-based case definitions," *Int J Infect Dis*, vol. 122, pp. 647–655, Sep. 2022, doi: 10.1016/J.IJID.2022.06.060.

[3] T. Nayak *et al.*, "Deep learning based detection of monkeypox virus using skin rash images," *Med Nov Technol Devices*, vol. 18, p. 100243, Jun. 2023, doi: 10.1016/J.MEDNTD.2023.100243.

[4] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," *Comput Electron Agric*, vol. 161, pp. 272–279, Jun. 2019, doi: 10.1016/J.COMPAG.2018.03.032.

[5] A. S. Sabyasachi, B. M. Sahoo, and A. Ranganath, "Deep CNN and LSTM Approaches for Efficient Workload Prediction in Cloud Environment," *Procedia Comput Sci*, vol. 235, pp. 2651–2661, Jan. 2024, doi: 10.1016/J.PROCS.2024.04.250.

[6] T. C. Pham, C. M. Luong, M. Visani, and V. D. Hoang, "Deep CNN and Data Augmentation for Skin Rash Classification," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018. doi: 10.1007/978-3-319-75420-8_54.

[7] D. Bala *et al.*, "MonkeyNet: A robust deep convolutional neural network for monkeypox disease detection and classification," *Neural Networks*, vol. 161, pp. 757–775, Apr. 2023, doi: 10.1016/J.NEUNET.2023.02.022.

[8] L. Villalobos-Arias, C. Quesada-López, J. Guevara-Coto, A. Martínez, and M. Jenkins, "Evaluating hyper-parameter tuning using random search in support vector machines for software effort estimation," *PROMISE 2020 - Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering, Co-located with ESEC/FSE 2020*, pp. 31–40, Nov. 2020, doi: 10.1145/3416508.3417201.

[9] N. R. Huber *et al.*, "Random Search as a Neural Network Optimization Strategy for Convolutional-Neural-Network (CNN)-based Noise Reduction in CT," *Proc SPIE Int Soc Opt Eng*, vol. 11596, p. 62, Feb. 2021, doi: 10.1117/12.2582143.

[10] S. N. Ali *et al.*, "A Web-based Mpox Skin Rash Detection System Using State-of-the-art Deep Learning Models Considering Racial Diversity," *IEEE J Biomed Health Inform*, vol. XX, p. 1, Jun. 2023, Accessed: Oct. 06, 2024. [Online]. Available: https://arxiv.org/abs/2306.14169v1

[11] M. Elgendi *et al.*, "The Effectiveness of Image Augmentation in Deep Learning Networks for Detecting COVID-19: A Geometric Transformation Perspective," *Front Med (Lausanne)*, vol. 8, 2021, doi: 10.3389/fmed.2021.629134.

[12] A. Gabriel Sooai, S. Daeng Bakka Mau, D. Joseph Manehat, Y. Carmeneja Hoar Siki, S. Crossifixio Sianturi, and A. Herlin Mondolang a-d, "Optimizing Lantana Classification: High-Accuracy Model Utilizing Feature Extraction," *Jurnal Ilmiah Kursor*, vol. 12, no. 2, pp. 49–58, Dec. 2023, doi: 10.21107/KURSOR.V12I2.347.

[13] M. Hasnain, M. F. Pasha, I. Ghani, M. Imran, M. Y. Alzahrani, and R. Budiarto, "Evaluating Trust Prediction and Confusion Matrix Measures for Web Services Ranking," *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.2994222.

[14] R. Widodo, T. Badriyah, I. Syarif, W. Sandhika, P. Elektronika, and N. Surabaya, "Segmentation Of Lung Cancer Image Based On Cytologic Examination Using Thresholding Method," *Jurnal Ilmiah Kursor*, vol. 12, no. 1, pp. 41–48, Jul. 2023, doi: 10.21107/KURSOR.V12I01.277.

[15] S. S. Sabila and H. P. A. Tjahyaningtyas, "The Multiple Brain Tumor with Modified DenseNet201 Architecture Using Brain MRI Images: Classification Multiclass Brain Tumor Using Brain MRI Images with Modified DenseNet201," *Jurnal Ilmiah Kursor*, vol. 12, no. 3, pp. 147–158, Jul. 2024, doi: 10.21107/KURSOR.V12I3.379.