Terbit *online* pada laman : http://teknosi.fti.unand.ac.id/

Jurnal Nasional Teknologi dan Sistem Informasi

| ISSN (Print) 2460-3465 | ISSN (Online) 2476-8812 |

Case Study

A Web App for Acne Severity Identification with RGB Image Color Scheme

Fajril Akbar [a,*], *Erick Octa Wardana* [a]

[a] *Departemen Sistem Informasi Fakultas Teknologi Informasi, Universitas Andalas, Kampus Limau Manih, Padang 25163, Indonesia*

INFORMASI ARTIKEL

Sejarah Artikel:
Diterima Redaksi: 21 Juni 2025
Revisi Akhir: 04 Mei 2026
Diterbitkan *Online*: 05 Mei 2026

KATA KUNCI

Severity of acne,
Web apps,
Deep learning,
RGB image

KORESPONDENSI

E-mail: ijab@it.unand.ac.id*

A B S T R A C T

This study addresses the need to detect the severity of acne in skin care and beauty, particularly in managing the diagnosis of acne severity more efficiently and objectively. Acne, a common skin problem, can significantly affect an individual's quality of life and self-confidence. Referring to the prevalence of acne, this study leverages advancements in Deep Learning and Convolutional Neural Networks (CNN) to design a classification model capable of identifying the severity of acne in facial images using the RGB color scheme. CNN is an artificial neural network architecture that can process image data efficiently and accurately. CNN can extract essential features from acne images, such as color, texture, and shape, and classify the severity of acne into four categories: level 0, level 1, level 2, and level 3. The simple application of the model not only provides an efficient solution for acne diagnosis but also has the potential to improve consistency and objectivity in healthcare services. By incorporating transfer learning and color schemes (RGB), the testing results show that the model successfully classifies the severity of acne with an accuracy of 86.89%. Thus, this research contributes to technical and technological advancements and has the potential to positively impact the overall quality of facial skin care services, marking a significant first step in improving facial skin care services.

1. INTRODUCTION

Acne vulgaris, commonly known as acne, is one of the most common skin conditions worldwide and can affect individuals of all ages. In addition to its physiological effects, acne can negatively impact a person's quality of life and self-confidence. Therefore, acne management, diagnosis of acne severity, and selection of appropriate treatment are crucial in dermatology [1]. Acne is one of the most common skin conditions, affecting approximately 9% of the global population, primarily among younger age groups from adolescence to early adulthood [1]. In determining the severity of acne, doctors need to diagnose manually by assessing the type of lesions (abnormalities or abnormalities in any part or tissue of the body, but more commonly occurring on the skin), quantity, and density of involvement. This manual method can be time-consuming [2]. In some situations, it may be difficult to objectively detect acne from color images to evaluate acne lesions accurately. Accurate assessment of acne severity is crucial in acne treatment. Therefore, a technique is needed to easily detect acne on the face using facial acne images and classify it accurately based on acne characteristics while minimizing potential errors [3].

In the context of data training, machine learning (ML) and deep learning (DL) use data modelling to address various diagnostic tasks and problems, such as medical diagnosis, speech recognition, computer vision, fraud detection, and behaviour recognition. One application of deep learning technology is acne detection using computer vision, which can determine the severity of a person's acne simply by inputting a facial image as a reference[4]. Convolutional neural networks have several applications in object detection, localization, video, and text processing. Applications that utilize these principles operate based on the fundamental use of convolutional neural networks to provide engineered features[5]

Ziying Vanessa Lim and her colleagues conducted research that addressed the issue of how to objectively and consistently assess and classify the severity of acne using facial images. Visual assessment and classification of acne by doctors can be subjective and vary between observers and within the same observer. The method used in this study was deep learning based on convolutional neural networks, utilizing facial photos with acne for training and testing. The study found that the acne detection model could generate automatic IGA scores aligned with doctors' manual IGA scores with 67% accuracy. The Pearson correlation between automatic and manual IGA scores for each model and image size was 0.77. The highest correlation was obtained using Inception v4 as the leading network on the largest image size, 1200x1600. Two sets of manual IGA scores showed a high correlation of 0.77, verifying the reliability of the ground truth labels. This study demonstrates that machine learning and deep learning can provide innovative solutions to address these challenges, particularly in acne detection [5].

https://doi.org/10.25077/TEKNOSI.v12i1.2026.079-086

Deep learning is required to classify acne levels, with performance capabilities across all application domains. Optimized features are automatically learned from the processed acne photo data, achieving resilience to common input changes. Different data types or applications can utilize deep learning with the same techniques, an approach often referred to as transfer learning[6]. Deep learning is highly scalable, with architectures such as ResNet [7].

One classification model that can be used in deep learning is the convolutional neural network (CNN), which has a deep learning architecture that works similarly to human vision. Convolutional mathematical operations extract feature meanings such as edges, textures, and colors in images. Clinical photos are the target of analysis by the CNN algorithm [2]. Applying this method can significantly assist doctors and beauty clinics in determining an individual's acne severity level with higher accuracy at an early stage before further treatment by specialists in the field, which can serve as the initial step in classifying acne severity levels [8].

The deep learning approach using the CNN method is promising [6], [9]. Various studies have been conducted using deep learning with image recognition to identify and classify acne [10], [11], [12], [13]. In addition, the development of mobile applications has also been researched to assess the severity of acne on the face [14]. By leveraging this technology, this study aims to develop a classification model capable of identifying the severity of acne on the face, enabling more consistent and accurate initial diagnoses. This model also has the potential to address the limited number of available dermatologists to serve patients.

Therefore, the problem formulation that this study attempts to solve is how to accurately identify the severity of acne on the face using an acne level classification model with a web-based system and a convolutional neural network. In this study, a deep learning model capable of classifying the severity of acne using a convolutional neural network was developed, and the performance of the deep learning model in categorizing the severity of acne was evaluated.

2. METHOD

This research consists of what can be seen in Figure 1. At the data collection stage, the dataset needed to support the research is taken from the GitHub user dataset. The dataset is in the form of an object image, namely, a piece of a person's face with acne. In the dataset, data is already divided into specific classes stored in a folder, which is then processed with the best method determined in the previous stage.

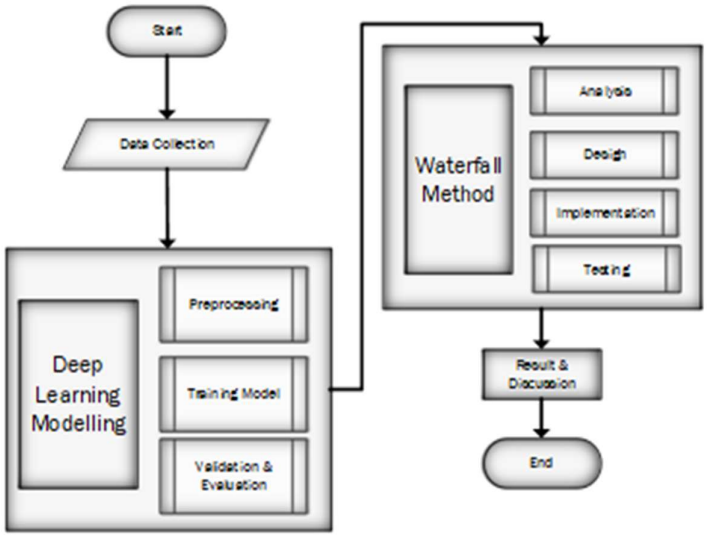


Figure 1. Research flowchart

At the deep learning modelling stage, analysis is conducted to find the best architecture to build an appropriate model. The results of the analysis are obtained from literature studies of previous similar studies. At this stage, testing is carried out on the data received from the Kaggle site. The data has been divided into test and training data, so data testing and training can be executed to obtain an algorithm with the best model accuracy. The data training and testing process is done by optimizing each neural network layer to minimize the difference between the labels given to the trained dataset and the output prediction.

3. RESULT AND DISCUSSION

3.1. Data Collection

The data used in this final project research is the Acne Dataset taken from the Github user "Team Anti Acne" with URL https://github.com/yinchuangsum/acne_demo/tree/master/data. In making this model, acne image data is needed that has been annotated by separating the dataset into various classes, namely:

- Level 0: Normal skin without acne.

- Level 1: The Percentage difference in skin color with normal skin (level 0) is 30%.
- Level 2: Percentage difference in skin color with normal skin (level 0) above 30%, below 60%.
- Level 3: Having acne with a very contrasting color difference between acne-prone skin and non-acne skin, and having a percentage difference in skin color with normal skin (level 0) above 60%.

There are a total of 610 acne image items divided into four classes, which will later be divided into training, testing, and validation data.

3.2. Deep Learning Modeling

3.2.1. Preprocessing Stage

Based on literature studies and related research that have been done previously, the architecture used for the image data recognition process in the deep learning model is the Convolutional Neural Network (CNN). At this stage, it is explained how the CNN architecture is used to build a deep learning model.

Classification using the CNN method is based on the RGB color scheme. Before being implemented, preprocessing is done by separating the data into four folders and labelling them. After that, the image classification process is carried out by grouping training, testing, and validation data, until the percentage can be determined according to the existing categories.

In the dataset used, the dataset was annotated, noise was removed, and the cheeks where acne is most often found were extracted. Therefore, the model will not learn the eyes, nose, and so on found in the dataset. This will improve the quality of the dataset. Data processing used in processing different data uses seven types of data augmentation methods:

- Rotation: 40 degrees
- Width Shift: 20% of the image width
- Height Shift: 20% of the image height
- Shear: Stretch the image by 20% of the original image
- Crop: 20% minimum zoom, 20% maximum zoom
- Flip: Horizontal, Vertical
- Fill Mode: Nearest

Data preparation begins by separating all data into four folders and giving each folder a name according to its level.

3.2.2. CNN Model training stage

The CNN architecture used in training this model is the Xception architecture, as seen in Figure 2. This architecture comprises 36 convolution layers and 35 filters and uses average pooling.

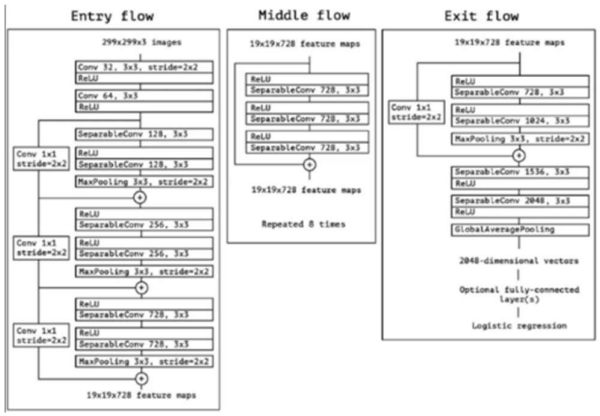


Figure 2. Arsitektur Xception [15]

Training data consists of four levels, namely level 0, level 1, level 2, and level 3. The data is collected and trained using Google Colab with the Convolutional Neural Network method, so it will later be used as a model to be a reference for testing data. Then upload the dataset to Google Drive and place it in one file directory.

After setting the drive directory used, several operations were made to load image data and create a DataFrame using the Pandas library in the Python environment, as seen in Figure 3.

```
filepaths=[]
labels=[]
classlist=os.listdir(data_dir)
for klass in classlist:
    classpath=os.path.join(data_dir,klass)
    If os.path.isdir(classpath):
        flist=os.listdir(classpath)
        for f in flist:
            fpath=os.path.join(classpath,f)
            filepaths.append(fpath)
            labels.append(klass)
Fseries= pd.Series(filepaths, name='filepaths')
Lseries=pd.Series(labels, name='labels')
df=pd.concat([Fseries, Lseries], axis=1)
print (df.head())
print (df['labels'].value_counts())
```

Figure 3. Creating DataFrame using Panda

After the class labels are entered into the DataFrame, the next step is to divide the dataset into train, test, and validation subsets, as seen in Figure 4.

```
train_split=.8
test_split=.1
dummy_split=test_split/(1-train_split)
train_df, dummy_df=train_test_split(df, train_size=train_split,
shuffle=True, random_state=123)
test_df, valid_df=train_test_split(dummy_df, train_size=dummy_split,
shuffle=True, random_state=123)
print ('train df length: ', len(train df), '  test df length: ',
len(test_df), '  valid_df length: ', len(valid_df))
```

Figure 4. Dataset distribution

Using the 'train_test_split' function to split the dataset into three subsets: train, test, and validation, with a proportion of 80% for train data, 10% for test data, and 10% for validation data.

```
train_df length:  488   test_df length:  61   valid_df length:  61
```

Figure 5. Splitting *Dataset*

Determine the height 'height=224', width 'width=224', number of color channels 'channel=3', and batch size 'batch_size=64' to be used to process the image. Create the types 'img_shape' and 'img_size' to represent the shape and size of the image. Calculate

https://doi.org/10.25077/TEKNOSI.v12i1.2026.079-086

the total amount of data in the test dataset 'test_df'. Calculate the batch size to be used on the test dataset by finding the divisor factors of the total data that meet several criteria, with a batch size that should not be more than 80, and the results are sorted in descending order, and the most significant value is taken.

Next, create an 'ImageDataGenerator' module from Keras to create an image data generator for training, testing, and validation. Create an 'ImageDataGenerator' object for data augmentation on the image. 'rescale=1.255' is used to change the pixel intensity scale to the range [0,1]. Creating a data generator for train, test, and validation with several parameters set involves using a dataframe with filepaths and labels columns, and various data augmentation settings such as rotation, shift, shear, zoom, and horizontal flip, with an output.

The next step is to create a deep learning model using the Xception architecture. Create a base of the model using the Xception architecture from keras with the configuration: 'include_top=False', 'weights= (imagenet)', 'input_tensor = Input(shape=224, 224,4)'. Use the 'get_layer' method to get a reference to the layer named 'block13_sepconv2' owned by the Xception model. Next, a convolutional neural network model will be created, as seen in Figure 6 and 7.

```
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras import Model
from tensorflow.keras import layers
x = layers.Flatten()(last_output)
x = layers.Dense(1024, activation='relu')(x)
x = layers.Dense(1, activation='softmax')(x)
model = Model( base_model.input, x)
model.compile(optimizer = RMSprop(learning_rate=0.0001),
              loss = 'categorical_crossentropy',
              metrics = ['acc'])
model.trainable = False
```

Figure 6. Xception CNN model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten,
Dropout
model_name='AcneLevel'
print("Building model with", base_model)
model = tf.keras.Sequential([
            # Note the input shape is the desired size of the image 128x128
with 3 bytes color
            # This is the first convolution
            base_model,
            tf.keras.layers.Conv2D(filters=32, padding='same',
kernel_size=3, activation='relu', strides=1),
            tf.keras.layers.MaxPool2D(pool_size=2, strides=2),
            tf.keras.layers.Dropout(rate=0.5),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(4, activation='softmax') ])
```

Figure 7. Model CNN

Create an Xception model using a Dense layer with 1024 units and a ReLU activation function. Re-add a Dense layer with 1 unit and a softmax activation function, and compile the model with the RMSprop optimizer, the categorical_crossentropy loss function, and the accuracy metric. Set 'model.trainable = False' to lock the weights of the Xception model, so that only the added layers can be trained. After creating the Xception model, create a second CNN model using the sequential API and add some layers. Set 'model_name' as 'AcneLevel', add a Conv2D layer with 32 filters, the same padding, a 3x3 kernel size, a ReLU activation, and a stride of 1. Add a MaxPooling2D layer with a pool size of 2x2 and a stride of 2. Add a Dropout layer with a dropout rate of 0.5, a Flatten layer, and a Dense layer with four units and a softmax activation function. The learning parameters initialized in the code above are the learning rate parameters with the Adam optimizer. Adam is an optimization algorithm that can substitute for the classical stochastic gradient descent procedure to update weights based on training data iteratively.

After the model is compiled, it is ready to be trained. When this fitting model is run, the CNN architecture that has been formed will work to execute the data that has been prepared previously. The training process involves 10 epochs and a batch size of 64. After this training process, the accuracy and loss values will be obtained; the epoch results can be seen in Figure 8.

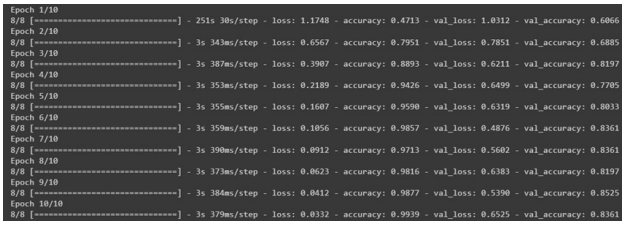


Figure 8. Epoch of model.fit

The result of this training is in the form of a model formed from the training process that has been carried out. After getting the model, the next step is to display the visualization graph as seen in Figure 9.

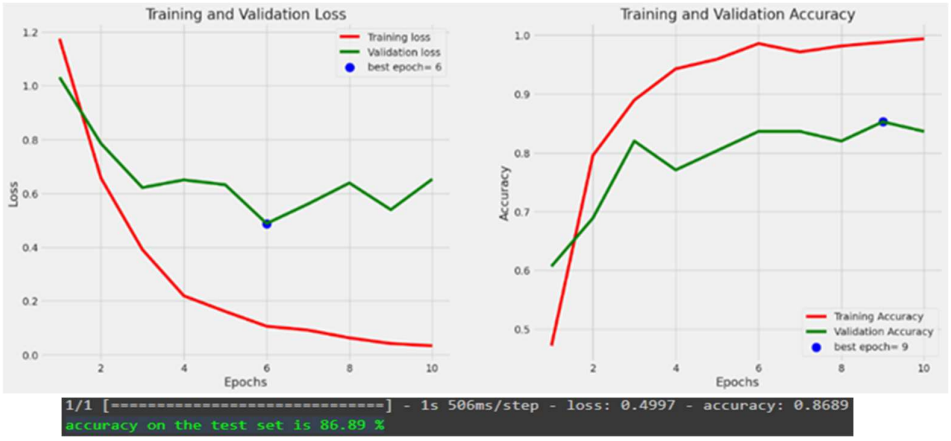


Figure 9. Visualization Graph

In the visualization graph results, the training and validation loss graph shows the best epoch at 6, with a validation loss of 0.4876. In the training and validation accuracy graph, the best epoch is the 9th, with a validation accuracy of 0.8525 and a test set accuracy of 86.89%.

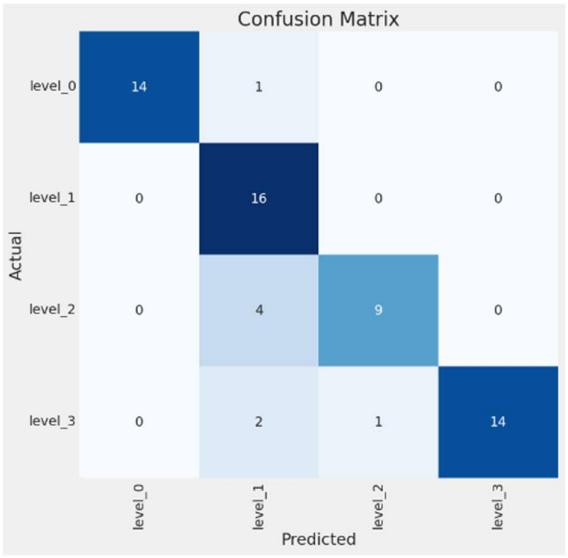


Figure 10. *Confusion Matrix* of CNN model

3.2.3. Validation and Evaluation

At this stage, testing is done after the model is implemented on the built web-based system. The manual calculation process uses a confusion matrix, a tool that visualizes the classification algorithm's performance. In the testing dataset, visualization is done with a confusion matrix as in Figure 10 and Table 2.

Table 2: *Confusion Matrix* pada *testing dataset*

Confusion Matrix					
		Predicted			
		level_0	level_1	level_2	level_3
Actual	level_0	14	1	0	0
	level_1	0	16	0	0
	level_2	0	4	9	0
	level_3	0	2	1	14

From the training results using CNN and Transfer Learning, an accuracy of 86.89% was obtained with fine-tuning of 10 epochs. From 61 test datasets, a Classification Report can be generated with four metrics for comparison: precision, recall, f1-score, and support. The results of the Classification Report are shown in Table 3, followed by an explanation afterwards.

Table 3: *Classification Report*

Classification report				
	precision	recall	f1-score	support
level_0	1.00	0.93	0.97	15
level_1	0.70	1.00	0.82.	16
level_2	0.90	0.69	0.78	13
level_3	1.00	0.82	0.90	17
accuracy			0.87	61
macro avg	0.90	0.86	0.87	61
weighted avg	0.90	0.86	0.87	61

https://doi.org/10.25077/TEKNOSI.v12i1.2026.079-086

Table 4. Testing Result of the Dataset level 0 classification

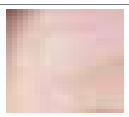
No.	Input	Output	Classification Result
3		Level 0	Correct
9		Level 1	Wrong
10		Level 1	Wrong
11		Level 0	Correct
13		Level 1	Wrong
17		Level 0	Correct
18		Level 1	Wrong
20		Level 0	Correct

Table 4 and 5 shows that, among the 20 tested Acne images, the overall classification results match the trained dataset labels. Based on the classification results in Table 4, the classification is incorrect in the 9th, 10th, 13th, and 18th tests.

Table 5. Testing Result of the Dataset level 3 classification

No.	Input	Output	Classification Result
5		Level 3	Correct
6		Level 3	Correct
7		Level 3	Correct
10		Level 3	Correct
11		Level 3	Correct
12		Level 3	Correct
13		Level 1	Wrong
14		Level 1	Wrong
15		Level 3	Correct
16		Level 3	Correct

In Table 4, it is shown that from 20 tested Acne images, the overall classification results are based on the trained dataset label. Based on the classification results in Table 4, in the 9th, 10th, 13th, and 18th tests, the classification results are wrong. So the amount of accuracy obtained from the following calculation is:

$$Accuracy = \frac{All\ True\ Positive}{Total\ Number\ Testing\ Entries} \times 100\%$$

$$Accuracy = \frac{16}{20} \times 100\%$$

So, the accuracy obtained from testing the Level 0 dataset is 80%, based on the testing data sample.

3.3. Design and Implementation of Web Apps

The implementation stage is the development stage of the application design. The program code is created at this stage to build the acne level classification application. The program uses

Python and HTML, and produces a web application using Flask as the framework for the user interface. This acne level classification application is web-based and uses deep learning convolutional neural networks implemented in Python with the Flask framework.

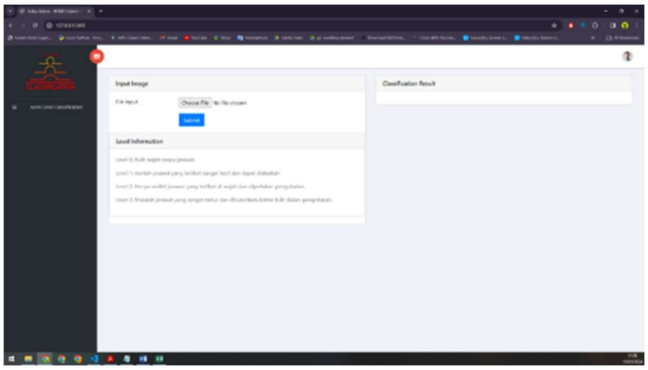


Figure 11. Front page of web apps

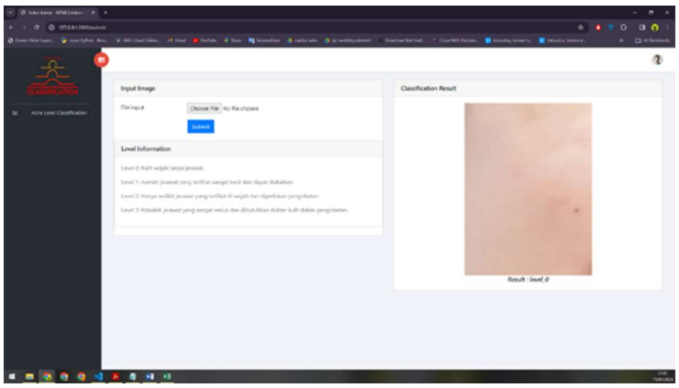


Figure 12. Upload image and Classification Result

Figure 11 shows the implementation of the application's main page, which directly displays the file selection for uploading images. Figure 12 shows the classification implementation and the classification results on the right side.

4. CONCLUSION

The implementation of deep learning to determine the level of acne, which is done using the Convolutional Neural Network method, has results that can be concluded that the Convolutional Neural Network method has succeeded in classifying acne based on its level: Level 0, Level 1, Level 2, Level 3 with a color scheme (RGB) and transfer learning. In addition, training the Deep Learning model using the Convolutional Neural Network method yields an accuracy of 86.89% in recognizing acne images and their levels, as calculated from the confusion matrix. The Deep Learning model has been successfully applied to web-based applications using the Flask framework.

REFERENCES

[1] J. K. Tan and K. Bhate, "A global perspective on the epidemiology of acne," *British Journal of Dermatology*, vol. 172, no. S1, pp. 3–12, 2015, doi: 10.1111/bjd.13462.

[2] A. Alzahrani, I. Petri, Y. Rezgui, and A. Ghoroghi, "Decarbonisation of seaports: A review and directions for future research," *Energy Strategy Reviews*, vol. 38, p. 100727, Nov. 2021, doi: 10.1016/j.esr.2021.100727.

[3] N. Yadav, S. M. Alfayeed, A. Khamparia, B. Pandey, D. N. Thanh, and S. Pande, "HSV model-based segmentation driven facial acne detection using deep learning," *Expert Systems*, vol. 39, no. 3, p. e12760, 2022, doi: 10.1111/exsy.12760.

[4] C. Aggarwal, "Neural Networks and Deep Learning. Cham: Springer International Publishing, 2018".

[5] Z. V. Lim *et al.*, "Automated grading of acne vulgaris by deep learning with convolutional neural networks," *Skin Research and Technology*, vol. 26, no. 2, pp. 187–192, 2020, doi: 10.1111/srt.12794.

[6] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021, doi: 10.1186/s40537-021-00444-8.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," presented at the Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[8] H. Zhang and T. Ma, "Acne detection by ensemble neural networks," *Sensors*, vol. 22, no. 18, p. 6828, 2022, doi: 10.3390/s22186828.

[9] Q. T. Huynh *et al.*, "Automatic acne object detection and acne severity grading using smartphone images and artificial intelligence," *Diagnostics*, vol. 12, no. 8, p. 1879, 2022, doi: 10.3390/diagnostics12081879.

[10] S. Liu *et al.*, "AcneGrader: An ensemble pruning of the deep learning base models to grade acne," *Skin Research and Technology*, vol. 28, no. 5, pp. 677–688, 2022, doi: 10.1111/srt.13166.

[11] A. Quattrini, C. Boër, T. Leidi, and R. Paydar, "A deep learning-based facial acne classification system," *Clinical, Cosmetic and investigational dermatology*, pp. 851–857, 2022, doi: 10.2147/ccid.s360450.

[12] S. Liu *et al.*, "AcneTyper: an automatic diagnosis method of dermoscopic acne image via self-ensemble and stacking," *Technology and Health Care*, vol. 31, no. 4, pp. 1171–1187, 2023, doi: 10.3233/thc-220295.

[13] S. K. Roy, G. Krishna, S. R. Dubey, and B. B. Chaudhuri, "HybridSN: Exploring 3-D–2-D CNN feature hierarchy for hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 2, pp. 277–281, 2019, doi: 10.1109/LGRS.2019.2918719.

[14] J. Wang *et al.*, "A cell phone app for facial acne severity assessment," *Applied Intelligence*, vol. 53, no. 7, pp. 7614–7633, 2023, doi: 10.1007/s10489-022-03774-z.

[15] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," presented at the Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1251–1258. doi: 10.1109/CVPR.2017.195.