Scientific Journal of Informatics
Vol. 11, No. 2, May 2024
p-ISSN 2407-7658 https://journal.unnes.ac.id/journals/sji/index e-ISSN 2460-0040

An Exploration of TensorFlow-Enabled Convolutional Neural Network Model Development for Facial Recognition: Advancements in Student Attendance System

Anie Rose Irawati[1], Didik Kurniawan[2*], Yohana Tri Utami[3], Rahman Taufik[4]
[1,2,3,4]Department of Computer Science, Faculty of Mathematics and Natural Sciences, Universitas Lampung, Indonesia

Abstract.
Purpose: Face recognition has become an increasingly intriguing field in artificial intelligence research. In this study, This study aims to explore the application of CNNs, implemented through TensorFlow, to develop a robust model for enhancing facial recognition accuracy in student attendance systems. The focus of this research is the development of a model capable of recognizing student faces under various lighting conditions and poses in an academic environment, using a multi-class dataset of student images collected from internship attendance records at the Computer Science Department.
Methods: The dataset, comprising facial images from 19 students, served as the foundation for training and validating the CNN model. The dataset, sourced from the computer science department's internship attendance records, included a total of 231 images for training and 59 images for validation. The preprocessing phase included facial area detection and categorization, resulting in a well-organized dataset for training and validation. The CNN architecture, consisting of seven layers, was meticulously designed to achieve optimal performance.
Result: The model exhibited exceptional accuracy, reaching 93% on the validation dataset after 300 training epochs. Precision, recall, and F1-score metrics were employed for a detailed evaluation across diverse classes, highlighting the model's proficiency in accurately categorizing facial images. Comparative analyses with a VGG-16-based model showcased the superiority of the proposed CNN architecture. Moreover, the implementation of a web service demonstrated the practical applicability of the model, providing accurate predictions with a remarkable response time of less than 0.3 seconds.
Novelty: This comprehensive study not only advances face recognition technology but also presents a model applicable to real-world scenarios, particularly in student attendance systems.

Keywords: Face recognition, Machine learning, Deep learning, CNN
Received May 2024 / **Revised** May 2024 / **Accepted** May 2024

INTRODUCTION

In recent years, there have been remarkable advancements in facial recognition technology, particularly with the utilization of Convolutional Neural Networks (CNN) and platforms like TensorFlow. CNN, a subset of deep learning, has seen widespread adoption in applications such as access control, payment systems, and security. However, despite these technological strides, the integration of facial recognition, often coupled with TensorFlow, into attendance systems within educational institutions remains limited.

Recent research highlights the critical importance of face recognition technology across various applications, particularly in attendance systems. Tarmissi et al. [1] and Annubaha et al. [2] demonstrate the effectiveness of this technology in enhancing the accuracy and efficiency of attendance systems, reducing the need for human intervention and improving data security. Additionally, face recognition techniques are explored in broader contexts by Mughaid et al. [3] Sethi and Jaiswal [4], and Maddu and Murugappan [5] for applications ranging from social network analysis to assessing student comprehension and detecting learner engagement. This technology not only facilitates the automation of attendance management but also opens new avenues for understanding and interpreting human interactions, crucial for a more dynamic and interactive attendance approach.

*Corresponding author.
Email addresses: anie.roseirawati@fmipa.unila.ac.id (Irawati), *didik.kurniawan@fmipa.unila.ac.id (Kurniawan), yohana.utami@fmipa.unila.ac.id (Utami), rahman.taufik@fmipa.unila.ac.id (Taufik)
DOI: 10.15294/sji.v11i2.3585

The synergy of facial recognition, driven by CNN and implemented through TensorFlow, presents a promising avenue. CNN excels in recognizing faces under diverse conditions, offering heightened accuracy compared to alternative methods. TensorFlow, as a powerful open-source machine learning library, complements CNN in the development and deployment of robust facial recognition models.

This research involves developing a CNN model using TensorFlow to recognize student faces from images taken during internship attendance. The process includes collecting a multi-class dataset of student images, training the CNN model, and evaluating its performance against existing attendance methods. Key performance metrics such as accuracy, efficiency, and scalability will be analyzed.

Face recognition is a rapidly evolving research field with potential applications in various domains such as security, surveillance, biometrics, and entertainment. There are two main categories in the literature review of face recognition methods: traditional methods and deep learning-based methods. Traditional face recognition methods rely on features and often make use of techniques like PCA, LBP, and SVM [6], [7], [8], while deep learning-based methods utilize deep learning techniques such as CNN and DBN to automatically learn facial feature representations [9], [10]. Deep learning-based methods exhibit superior performance in face recognition tasks, especially in challenging environments, but they require a substantial amount of training data.

Deep learning is a branch of machine learning that has garnered significant attention in recent years due to its ability to automatically learn from large volumes of data. The core idea behind deep learning is to use artificial neural networks to learn hierarchical representations of data that capture underlying patterns and relationships. This enables deep learning models to perform complex tasks with remarkable accuracy, such as image recognition, speech recognition, and natural language processing.

Deep learning-based methods, particularly CNNs, automatically learn hierarchical feature representations from raw image data [9], leading to significantly improved performance. CNNs, composed of convolutional layers, pooling layers, and fully connected layers, excel at capturing spatial hierarchies in images CNNs have achieved state-of-the-art results on various benchmark datasets, like ImageNet, and are commonly employed in applications such as object recognition, scene recognition, and face recognition [10].

Research on face recognition has seen significant advancements with the use of convolutional neural networks (CNNs), demonstrating high effectiveness in various applications, including attendance systems. Lou and Shi [11] developed a CNN model optimized for recognizing faces with high accuracy under various lighting conditions and angles, while Li et al. [12] focused on facial expression recognition, which can enhance the accuracy of attendance systems by ensuring the recognition of not only the identity but also the user's expressions. Jin et al. [13] extended the application of face recognition to medical diagnosis using deep transfer learning, showing the flexibility of CNN models. Rusia and Singh [14] reviewed various techniques to address challenges in face recognition such as pose and lighting variations, which are crucial for reliable attendance systems. Additionally, Aglasia et al. developed a criminal face recognition system based on sketches using Haar Wavelet Transform and Local Binary Pattern, which effectively addresses the challenges of recognizing incomplete or distorted facial sketches [15], Wirdiani et al. investigated face identification using the K-Nearest Neighbor (KNN) algorithm, demonstrating high accuracy in face identification by comparing unknown faces with a known dataset [16]. Meanwhile, Boussaad and Boucetta [17] explored the use of deep learning-based descriptors, which can improve the performance of face recognition in attendance systems. These studies highlight the critical role of deep learning techniques in achieving better results in face recognition for attendance systems.

Recurrent Neural Networks (RNNs) represent another category of deep learning models extensively used for sequential data, such as audio signals and natural language text. RNNs can capture long-range dependencies in sequential data and find application in various tasks, including speech recognition, language translation, and sentiment analysis [18], [19].

In addition to CNNs and RNNs, there are other deep learning models developed for various tasks. Generative Adversarial Networks (GANs) are deep learning models that generate new data samples resembling training data [20]. Autoencoders (AEs) are deep learning models capable of learning compact data representations in an unsupervised manner [21]. Deep Belief Networks (DBNs) are deep learning models that learn hierarchical data representations in an unsupervised manner [21].

Convolutional Neural Networks (CNN), a type of machine learning model, have been successfully applied to various computer vision tasks, including image classification, object detection, and segmentation [9], [22], [23]. CNNs consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers, which process and extract hierarchical feature representations from images [21]. Convolutional layers apply filters to local regions of the input image and generate feature maps, which are then down sampled by the pooling layers. Finally, fully connected layers perform classification based on the extracted features [20]. Figure 1 illustrates the process flow of a Convolutional Neural Network.

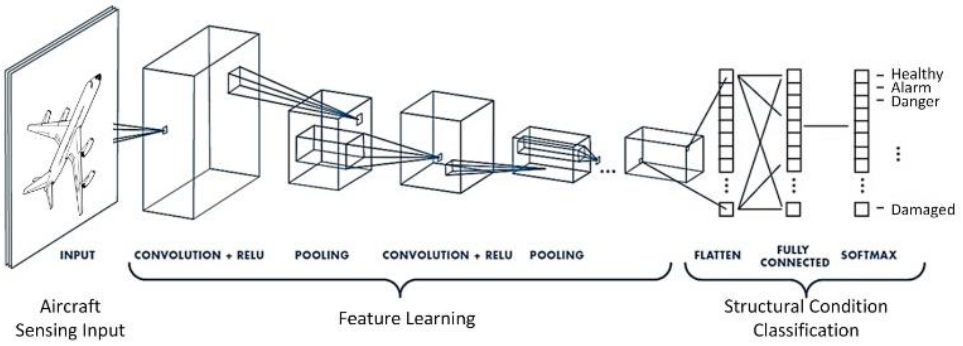

Figure 1. Convolutional neural network process

In mathematics, specifically within the field of functional analysis, convolution is a mathematical operation involving two functions, denoted as f and g, that results in a third function denoted as $(f * g)$.

Continuous Convolution denote as (1) [24]:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau).g(t-\tau)d\tau \quad (1)$$

Here, $(f * g)(t)$ represents the value of the convolution of f and g at time t. The integral involves multiplying the values of f and g at each point (τ) and summing them up over all possible values of τ.

In the discrete case, the convolution is calculated by summing the products of corresponding elements of the two functions f and g at each point n.

Discrete Convolution denote as (2) [25]:

$$(f * g)[n] = \sum_{k=-\infty}^{\infty} f(k).g(n-k) \quad (2)$$

In the case of CNNs, it's a mathematical operation on two matrices: the input matrix (often the image) and the filter (also called a kernel or a convolutional kernel).

Given an input matrix I and a filter matrix K, the convolution operation is denoted as $I * K$, and it's calculated as (3) [26]:

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I_{[i-m,j-n]} \cdot K_{[m,n]} \quad (3)$$

Here, $S(i,j)$ is the value at position (i,j) in the output feature map.

One of the key advantages of CNN is its ability to automatically learn feature representations from large training datasets without the need for manual feature engineering [9]. This results in top performance in various computer vision tasks, particularly image classification [27]. Another advantage of CNN is its capability to generalize effectively to new data and tasks due to its ability to learn hierarchical feature representations of images [23].

Activation functions serve as the non-linearities that introduce expressive power to neural networks, enabling them to capture complex relationships in data. In the context of CNNs, the choice of activation functions has a profound impact on the model's ability to learn and generalize from input data.

Activation functions play a vital role in the training of Convolutional Neural Networks, which can activate the feature of neurons to solve nonlinear problems. A proper activation function has a better ability to map data in dimensions [28], [29]. Activation functions play a crucial role in deciding whether a neuron should be activated. While various types of activation functions exist, ReLU (Rectified Linear Unit function) stands out as the most popular. ReLU is preferred for its faster gradient descent, making it considered superior to Sigmoid and Tanh functions [30].

Rectified Linear Unit (ReLU)

Introduced by Nair and Hinton in 2010, ReLU replaces all negative values in the input with zero (4), introducing non-linearity to the network. ReLU is a classical activation function, which effectiveness has been verified in previous works [30], [31], [32]. The success of ReLU owes to identically propagating all the positive inputs, which alleviates gradient vanishing and allows the supervised training of much deeper neural networks. In addition, ReLU is computational efficient by just outputing zero for negative inputs, and thus widely used in neural networks.

$$ReLU(x) = f(x) = \max(0, x) \tag{4}$$

Leaky ReLU (5), introduces a small slope (α) for negative values to avoid dead neurons.

$$f(x) = \max(\alpha x, x) \tag{5}$$

Parametric ReLU (6) (PReLU), Similar to Leaky ReLU, but the slope (α) is learned during training.

$$f(x) = \max(\alpha x, x) \tag{6}$$

Sigmoid and Hyperbolic Tangent (tanh)

Sigmoid and hyperbolic tangent (tanh) activation functions have been foundational in neural network research. Sigmoid (7), with its range between 0 and 1, finds application in the output layer for binary classification tasks, providing a probability-like output.

Sigmoid Function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{7}$$

Hyperbolic Function

$$f(x) = \tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{8}$$

Hyperbolic function (8) similar to the sigmoid but with an output range from -1 to 1.

Softmax function

SoftMax function Used in the output layer for multi-class classification problems. It converts a vector of real numbers into a probability distribution (9).

$$softmax(x)_i = \frac{e^{xi}}{\sum_j e^{xj}} \tag{9}$$

Pooling layers are used to down sample the spatial dimensions of the feature maps produced by the convolutional layers. Max pooling is a common technique where the maximum value in a local region is selected. The mathematical representation as (10).

$$P(i, j) = \max_{m,n} S\,(2i + m, 2j + n) \tag{10}$$

Here, $P(i, j)$ is the value at position (i, j) in the pooled feature map and S is the output of the convolutional layer (3).

Keras Tensorflow is developed by François Chollet as part of a research project and released as an open-source project in March 2015, Keras stands out as a high-level Deep Learning API. Its appeal lies in its ability to streamline the building, training, evaluation, and execution of various neural networks. To handle the demanding computations required by neural networks, the keras-team relies on a computation backend. The three prominent open-source deep learning libraries used as backends are TensorFlow, Microsoft Cognitive Toolkit (CNTK), and Theano [33].

The research study highlights the superiority of TensorFlow and Microsoft Cognitive Toolkit (CNTK) over Theano [34]. Several considerations favor the use of Tensor-Flow, particularly in comparison to CNTK are:

- TensorFlow's popularity stems from its user-friendly high-level API,
- TensorFlow boasts a larger and more active community. Furthermore, as an open-source framework, it was developed by the Google Brain Team [35],
- TensorFlow's strength lies in its extensive ecosystem, supporting diverse platforms and devices such as CPUs, GPUs, and TPUs. The framework constructs elaborate computation graphs, where each node signifies a mathematical operation, and edges denote communication between nodes. This data flow graph enables explicit communication between sub computations, facilitating parallel execution or the use of multiple devices for partitioned computations[36].
- TensorFlow provides flexibility for deploying models across various platforms, including mobile devices (TensorFlow Lite) and web browsers (TensorFlow.js).
- The versatility of TensorFlow is evident in its widespread use across various industries and research domains, making it a preferred choice for diverse applications.

Based on the background and literature review, this study aims to evaluate the effectiveness of CNNs for student attendance systems by collecting a multi-class dataset of student images captured under varying conditions to simulate real-world scenarios, developing and training the CNN model using TensorFlow's high-level APIs for streamlined model building and training processes, and providing a comprehensive evaluation of the CNN-based facial recognition system, highlighting its advantages over traditional methods, with the goal of demonstrating the model's effectiveness in enhancing accuracy and efficiency, paving the way for more widespread adoption of deep learning technologies in educational institutions, and offering recommendations for future research and development to further improve model performance and explore additional applications of facial recognition technology.

METHODS

This research methodology comprises several key phases. The initial phase involves a comprehensive literature review to acquaint the study with the latest advancements in facial recognition techniques and attendance systems.

The second phase, Image Collection, focuses on gathering student image data, with a subsequent process to detect and store facial regions within the images.

The third phase centers around Developing a Deep Learning Model, which encompasses several steps:

- Data Preprocessing: This step prepares the collected student facial image data for modeling. Tasks such as image resizing, data normalization, and the partitioning of data into training, validation, and testing sets are conducted to ensure data cleanliness and readiness for model input.
- Model Selection: The research selects the appropriate model type based on data and research objectives. Convolutional Neural Networks (CNN) were chosen for this study. Model selection is based on validation results from the previous phase and considers factors like accuracy, speed, and complexity.
- Model Training: The selected model is trained using the prepared training data to enable it to recognize patterns and features in the data, thus achieving accurate facial recognition.
- Model Evaluation: This phase assesses the trained model's performance using separate testing data. Metrics like accuracy, precision, recall, and F1-score are measured to determine how well the model performs facial recognition. The model is advanced to the next phase based on evaluation results meeting established criteria.

The final step is suggested to involve integrating the model into the attendance system. During this phase, a prototype attendance system will be constructed with the aim of implementing the acquired model to detect multiple facial objects simultaneously. The complete step of this research is shown in Figure 2.

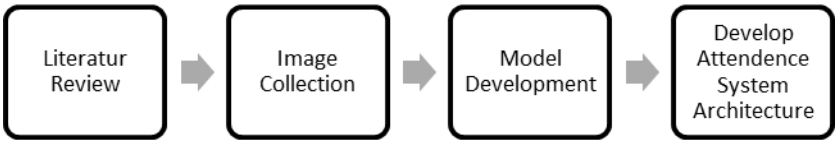

Figure 2. Research methodology

Below is a comprehensive explanation of phases 2 and 3 of the research conducted.

Image collection

Data collection involved gathering facial images from the computer science department's internship attendance records. The researcher successfully acquired 876 facial images from 64 students. These facial images form the dataset for the subsequent phases of the research. Following this, facial area detection was performed on each image, and the images were categorized and stored in folders corresponding to each student. This step is intended to prepare the data for further processing and modeling. The process utilized facial detection algorithms as described in the literature.

Data preprocessing

In this phase, the data undergoes preparation through a series of transformations and processing steps applied to the previously collected dataset. One crucial step involves resizing the images to meet specific requirements. This is vital to ensure that the data to be processed maintains a consistent and uniform resolution. Furthermore, additional operations include normalizing pixel intensities, converting the images to grayscale, and eliminating noise.

All these actions serve the purpose of improving the image data's quality, which, in turn, enhances the performance of the model under development.

The processed image data is confined to the facial area, with unchanged resolution and converted to grayscale, as depicted in Figure 3 below:

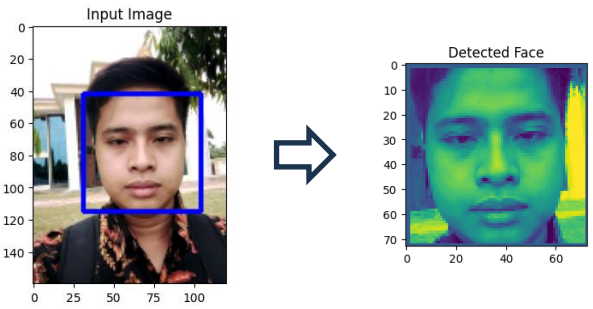

Figure 3. Example of processed image data

Model selection

In this research, the primary model architecture for identifying faces within a multi-class dataset is the Convolutional Neural Network (CNN). This CNN model comprises several key components that are instrumental in achieving a high level of accuracy. A comprehensive explanation of the CNN's design, including the convolutional layers for feature extraction, max-pooling layers for dimension reduction, fully connected (dense) layers for high-level feature abstraction, and output layers for class predictions, is provided. The architecture is expected to deliver accurate face identification across various scenarios and different identity classes within the dataset.

Here is an explanation of the sequence of layers in the CNN architecture used in this study:

1st layer: Convolutional Layers 1

The first convolutional layer employs 32 filters with a kernel size of 3x3. These filters scan the input images to detect specific features. The Rectified Linear Unit (ReLU) activation function is applied. If $X_{i,j,k}$ represents the input images data or input feature map at position (i, j) in the spatial dimensions and at channel k, the output of the first convolutional layer, denoted as $Y^{(1)}$, based on (4), $Y^{(1)}$ computed as (11).

$$Y^{(1)}_{i',j',k'} = \text{ReLU}\left(\sum_{i,j,k} X_{i,j,k} \cdot W^{(1)}_{i',j',k'} + b^{(1)}_{k'}\right) \tag{11}$$

Here, $Y^{(1)}_{i',j',k'}$ represents the activation of the (k')-th filter at position (i', j') in the output feature map. $W^{(1)}_{i',j',k'}$ is the weight associated with the (k')-th filter, and $b^{(1)}_{k'}$ is the bias term.

2nd layer: Max-Pooling Layers 1

Max-pooling layers are added after each convolutional layer to reduce the dimension of the feature maps generated by the convolutional layers. Max-pooling is performed using a 2x2 window, which means that for each 2x2 pixel grid in the feature map, only the maximum pixel value is retained, effectively reducing the feature map's size.

The output of the first max pooling layer, denoted as $Y^{(2)}$, is obtained by applying max pooling (12) with a size of 2×2.

$$Y^{(2)}_{i',j',k'} = \max_{m,n} Y^{(1)}_{2i'+m,2j'+n,k'} \tag{12}$$

This operation downsamples the spatial dimensions of the feature maps obtained from the previous layer.

3rd layer: Convolutional Layers 2

The second convolutional layer, following the first, consists of 64 filters with a 3x3 kernel size. The output of the second convolutional layer, denoted as $Y^{(3)}$, is computed similarly to the first convolutional layer using (13).

$$Y^{(3)}_{i',j',k'} = \text{ReLU}\left(\sum_{i,j,k} Y^{(2)}_{i,j,k} \cdot W^{(2)}_{i',j',k'} + b^{(2)}_{k'}\right) \tag{13}$$

4th layer: Max-Pooling Layers 2

The output of the second max pooling layer, denoted as $Y^{(4)}$, is obtained by applying max pooling (14) with a size of 2×2.

$$Y^{(4)}_{i',j',k'} = \max_{m,n} Y^{(3)}_{2i'+m,2j'+n,k'} \tag{14}$$

5th layer: Flatten Layer

The Flatten layer transforms the 3D matrix $Y^{(4)}$ into a 1D vector $Y^{(5)}$ using (15).

$$Y^{(5)}_i = Y^{(4)}_{i',j',k'} \tag{15}$$

6th layer: Fully Connected (Dense) Layers 1

After that, we add a fully connected (dense) layer with 128 units and ReLU activation function. This layer aims to process previously extracted features and learn more complex relationships.

The output of the first dense layer, denoted as $Y^{(6)}$, is computed as (16).

$$Y^{(6)}_i = \text{ReLU}\left(\sum_j Y^{(5)}_j \cdot W^{(3)}_{i,j} + b^{(3)}_i\right) \tag{16}$$

7th layer: Dense Layer 2 (Output Layer)

The final layer is the output layer, responsible for generating class predictions. Given the multi-class nature of the problem (19 classes), the SoftMax activation function is used in this layer. SoftMax transforms the network's output into a probability distribution over the classes, making it suitable for multi-class

classification. The number of units in the output layer corresponds to the number of classes in the dataset (19 classes), ensuring that the network produces a probability distribution across all possible classes.
The output of the second dense layer (output layer), denoted as $Y^{(7)}$, is computed using the SoftMax activation function (17).

$$Y_i^{(7)} = \frac{e^{Y_i^{(6)}}}{\sum_k e^{Y_k^{(6)}}} \tag{17}$$

Develop attendance system architecture

After successfully building the classification model, the next step is to test the model through the implementation of an application in the form of a web service. This web service is designed to receive inputs in the form of images from users and return the classification predictions generated by the model. The architecture of the system illustrated in figure 4 and figure 5 illustrates the layout of the main components involved in the workflow of the web service.

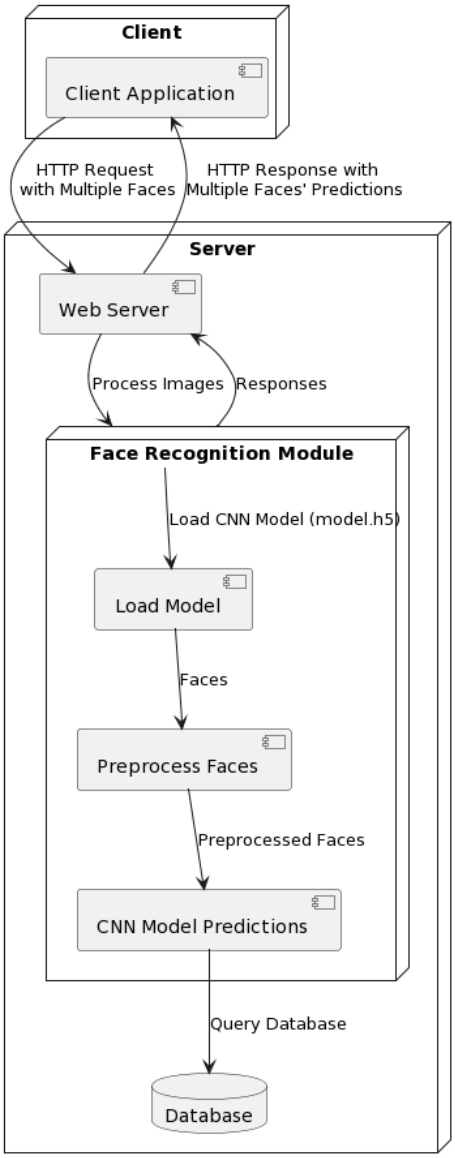

Figure 4. System architecture

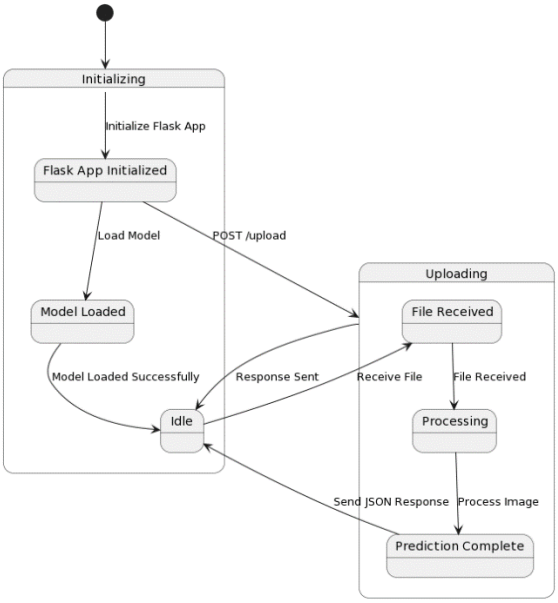

Figure 5. Workflow of the web service

This research roadmap centers on addressing facial recognition challenges utilizing Convolutional Neural Networks (CNN). he primary aim of this research is to develop a CNN model to enhance accuracy and efficiency in facial recognition, with potential application in a student attendance system. The investigation unfolds across three pivotal stages: image collection, the development of a deep learning model, and the evaluation of the suggested model's performance.

RESULTS AND DISCUSSIONS

Based on the architecture defined in the methodology section, the following is the implementation in Python code for defining the model using TensorFlow and Keras. This implementation details each layer discussed in the design of the CNN architecture, from the initial convolutional layer to the output layer that performs classification. Model architecture is defined in figure 6.

```
# Initialize the Sequential model
model = Sequential()

# Add the first convolutional layer with 32 filters of size 3x3
# and ReLU activation function, define the input shape for the image
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(54, 54, 3)))

# Add the first Max Pooling layer with a pool size of 2x2
model.add(MaxPooling2D((2, 2)))

# Add the second convolutional layer with 64 filters of size 3x3
# and ReLU activation function
model.add(Conv2D(64, (3, 3), activation='relu'))

# Add the second Max Pooling layer with a pool size of 2x2
model.add(MaxPooling2D((2, 2)))

# Add the Flattening layer to transform the data into a one-dimensional vector
model.add(Flatten())

# Add the first Dense layer (fully connected layer) with 128 neurons
# and ReLU activation function
model.add(Dense(128, activation='relu'))

# Add the second Dense layer (output layer) with 61 neurons
# and softmax activation function for multi-class classification
model.add(Dense(61, activation='softmax'))

# Display the summary of the constructed model
model.summary()
```

Figure 6. Code for CNN model construction

The code Figure 6 constructs a CNN model including two convolutional and max pooling layers for feature extraction, followed by a flattening layer and two dense layers acting as classifiers. The output layer uses the softmax activation function for multi-class classification, suitable for the 61 desired classes.
The visualization of the model is depicted in Figure 7.

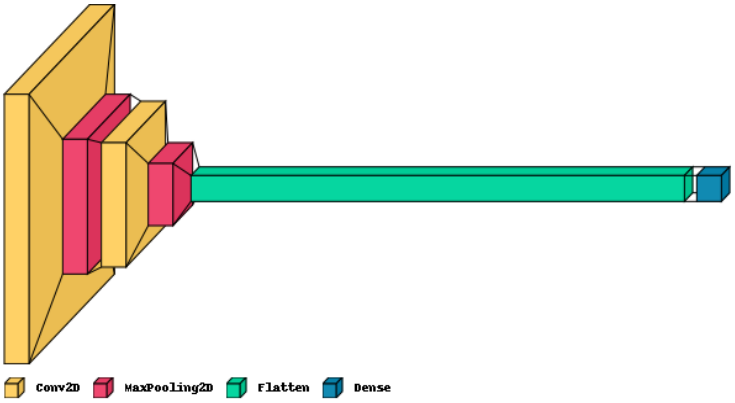

Figure 7. Illustration of the CNN architecture used in the research

Model training

During the model training process, several key strategies are adopted. Firstly, the Categorical Cross entropy loss function, a widely used metric in multi-class classification tasks, is employed. Additionally, the Adam optimizer is selected to efficiently optimize the model's parameters. The Adam optimizer is known for its effectiveness in training complex neural networks.

To mitigate the potential issue of overfitting during training, the research incorporates the Early Stopping technique. This means that training will automatically cease if the accuracy on the validation data doesn't improve for 20 consecutive epochs. This approach prevents the model from excessively adapting the training data and ensures that it can generalize well to previously unseen data.

Moreover, the model is trained for 300 epochs, enable the model to capture complex patterns within the training data. This training strategy aim to equip the model with strong capabilities to identify faces with a high level of accuracy.

Model performance

The outcomes of the experiments demonstrate that the newly created model attains an exceptionally high level of accuracy on the validation dataset, reaching an accuracy rate of 93% after 300 training epochs. This improvement in accuracy emphasizes the model's efficacy in categorizing facial images across diverse classes within this complex dataset. Figure 8 depicts the accuracy progression per epoch, while Figure 9 visualizes the Training and Validation Loss per Epoch.

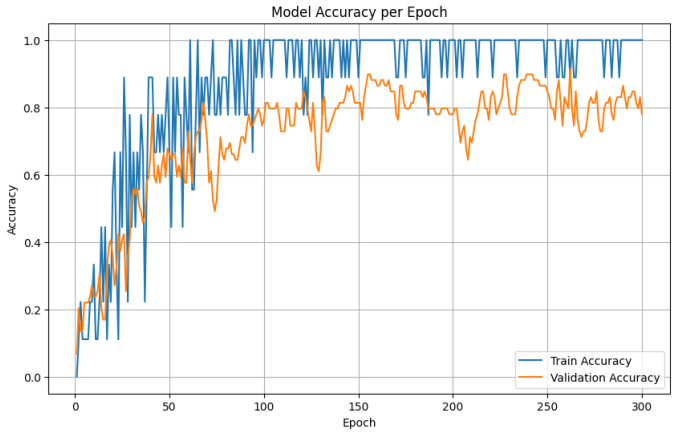

Figure 8. Illustration of model accuration per epoch

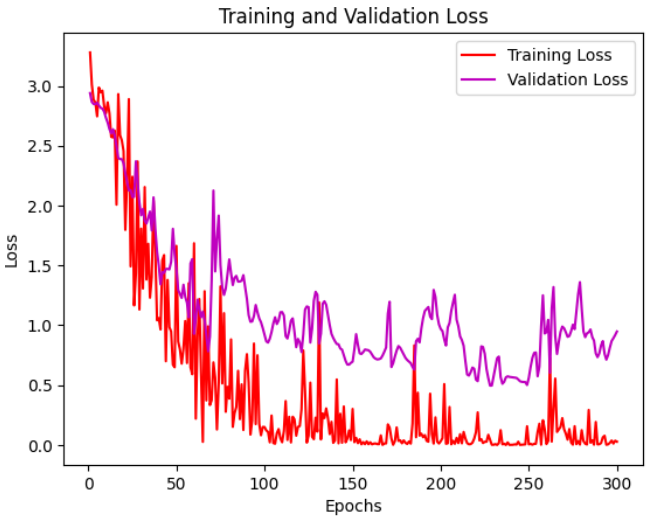

Figure 9. Illustration of training and validation loss per epoch

To determine whether the constructed model is effective, we compared it by building a model using the VGG-16 architecture. Based on the training process with the same dataset, the results obtained are indicated in Table 1.

Table 1 represents a comparison of the model architectures used in the study with VGG-16.

Table 1. Comparison of model architectures

Model	*Accuracy*	*Architecture Layer*	*Development Time (s)*
Used Model	93%	7	37.81
VGG-16	83%	16	347.9

The used model underwent training and testing using an extensive dataset comprising 19 distinct classes. The subsequent sections offer an in-depth examination of crucial metrics, encompassing precision, recall, F1-score, and the confusion matrix, to evaluate the model's performance across diverse classes. Table 2 provides a consolidated overview of the metrics for specific classes.

Tables 2 and 3 present a comparative evaluation of two models based on precision, recall, and F1-score metrics. In Table 2, the "Used Model" showcases consistent and high precision, recall, and F1-score across various classes, with perfect scores for several classes. This leads to elevated micro, macro, and weighted averages, indicating overall effectiveness in classification.

Table 2. Summaries model metrics

Class	*Precision*	*Recall*	*F1-Score*
Class 1	1.00	0.80	0.89
Class 2	1.00	0.67	0.80
Class 3	0.89	1.00	0.94
Class 4	1.00	0.80	0.89
Class 5	1.00	1.00	1.00
Class 6	0.00	0.00	0.00
Class 7	0.00	0.00	0.00
Class 8	1.00	1.00	1.00
Class 9	0.67	1.00	0.80
Class 10	0.67	1.00	0.80
Class 11	1.00	0.75	0.86

Class 12	1.00	1.00	1.00
Class 13	1.00	1.00	1.00
Class 14	1.00	1.00	1.00
Class 15	1.00	1.00	1.00
Class 16	1.00	1.00	1.00
Class 17	1.00	1.00	1.00
Class 18	1.00	1.00	1.00
Class 19	1.00	1.00	1.00
Micro Avg	0.93	0.93	0.93
Macro Avg	0.85	0.84	0.84
Weighted Avg	0.96	0.93	0.94

Table 2 demonstrates the robustness of the suggested model through its precision, recall, and F1-scores for each class, showcasing its capability to accurately classify instances across various categories. Classes 1, 2, 4, 5, 8, 12, 13, 14, 15, 16, 17, 18, and 19 exhibit exceptional performance, indicating high predictive accuracy. In contrast, Classes 3, 9, 10, and 11 show variability in performance, and Classes 6 and 7 present zero values across all metrics, suggesting potential limitations in these specific categories.
The aggregate metrics further underscore the model's overall effectiveness. The high Micro Average performance reflects the model's strong global prediction capability. The Macro Average, treating each class equally, highlights performance variations. The Weighted Average, accounting for class imbalance, demonstrates consistently high performance, though slightly lower than the Micro Average.

We also compared it with a model developed using the VGG-16 architecture, and the results are shown in Table 3 below.

Table 3. Evaluation summaries using VGG-16 architecture

Class	***Precision***	***Recall***	***F1-Score***
Class 1	0.71	1.00	0.83
Class 2	0.67	0.67	0.67
Class 3	1.00	0.88	0.93
Class 4	0.80	0.80	0.80
Class 5	0.00	0.00	0.00
Class 6	0.00	0.00	0.00
Class 7	0.00	0.00	0.00
Class 8	1.00	1.00	1.00
Class 9	0.67	1.00	0.80
Class 10	1.00	1.00	1.00
Class 11	0.67	1.00	0.80
Class 12	0.33	0.50	0.40
Class 13	0.00	0.00	0.00
Class 14	1.00	1.00	1.00
Class 15	0.89	0.80	0.84
Class 16	1.00	0.80	0.89
Class 17	1.00	1.00	1.00
Class 18	0.00	0.00	0.00
Class 19	1.00	1.00	1.00
Micro Avg	0.83	0.83	0.83
Macro Avg	0.62	0.65	0.63
Weighted Avg	0.82	0.83	0.82

Comparatively, Table 3 illustrates a model using the VGG-16 architecture, which shows more varied performance across classes. Some classes have lower precision, recall, and F1-scores, leading to lower micro, macro, and weighted averages compared to those in Table 2. This indicates that the "Used Model" in Table 2 outperforms the VGG-16-based model, emphasizing the impact of model architecture on achieving superior classification results.

After successfully developing the classification model, the next step is to test the model through the implementation of a server application in the form of a web service. This web service is designed to receive inputs in the form of images from users and return the classification predictions generated by the model.

After developing the classification model, it was tested through a server application implemented as a web service. This web service allows users to submit images and receive classification predictions. Figure 10 and 11 represent the input and output of the web service.

Figure 10. Input image

Figure 11. Output: prediction

The test results indicate that the web service can respond well to user requests. The predictions generated by the web service are consistent with the accuracy of the classification model tested earlier. The results depicted several individuals within green bounding boxes, indicating successful detection, with each box labeled with a class such as 2, 3, 4, 11, 13, 15, and 18, showcasing the system's ability to differentiate between facial features. However, the study identified certain limitations, including one face that was undetected possibly due to distance or clarity issues, and a misclassification error where a face that should have been labeled as Class 4 was erroneously classified as Class 3. These issues underscore the challenges in deploying facial recognition technology in varying conditions and highlight the necessity for ongoing model training and refinement to enhance accuracy and reliability. The main advantage of this implementation is the web service's ability to provide predictions with a response time of less than 0.07 second, creating a responsive and efficient user experience. The detailed prediction time of the web service based on input given depicted in Table 4.

Table 4. Prediction time of the web service

Class	Prediction Time (s)
Class 2	0.15305638313293457
Class 3	0.06874251365661621
Class 3	0.06912469863891602
Class 2	0.06974911689758301
Class 15	0.07637834548950195
Class 15	0.06870913505554199
Class 18	0.06613659858703613
Class 3	0.06746387481689453
Class 11	0.06158852577209473
Class 11	0.06870317459106445
Class 11	0.06642532348632812
Class 18	0.0701744556427002
Class 4	0.06722068786621094
Class 13	0.0684659481048584
Average Prediction Time	0.07442419869559

CONCLUSION

In conclusion, this study explored the development of a robust Convolutional Neural Network (CNN) model for facial recognition, leveraging TensorFlow to enhance a student attendance system. The constructed model, referred to as the "Used Model," demonstrated exceptional performance on a diverse dataset with 19 distinct classes, achieving an impressive accuracy of 93% with a simplified architecture of just 7 layers. Comparative analysis revealed that this model outperformed a VGG-16-based model, showcasing superior precision, recall, and F1-scores across most individual classes, thus underscoring its robustness and proficiency in facial image classification.

While the model exhibited notable strengths, challenges in specific classes indicate areas for future refinement. Furthermore, the practical applicability of the model was validated through the implementation of a web service, allowing users to submit images and receive accurate predictions with a response time of less than 0.07 seconds. This deployment highlights the model's real-world efficiency and effectiveness, marking significant advancements in the development of student attendance systems. The comprehensive evaluation and successful deployment of the TensorFlow-enabled model emphasize its robustness and potential for widespread application in similar domains.

ACKNOWLEDGEMENT

We wish to acknowledge the financial support provided by University of Lampung (Unila) under the Ministry of Culture and Research (Kementerian Kebudayaan dan Riset) for their financial support in this research. We would also like to express our appreciation to the research institution of Unila for the essential facilities and resources provided, which played a crucial role in this study. Finally, we acknowledge the

invaluable contributions of all the student participants who greatly aided in data collection and analysis, without whom this study would not have been as comprehensive.

REFERENCES

[1] K. Tarmissi, H. Allaaboun, O. Abouellil, S. Alharbi, and M. Soqati, "Automated Attendance Taking System using Face Recognition," 21st International Learning and Technology Conference: Reality and Science Fiction in Education, L and T 2024, pp. 19–24, 2024, doi: 10.1109/LT60077.2024.10469452.

[2] C. Annubaha, A. P. Widodo, and K. Adi, "Implementation of eigenface method and support vector machine for face recognition absence information system," Indonesian Journal of Electrical Engineering and Computer Science, vol. 26, no. 3, pp. 1624–1633, Jun. 2022, doi: 10.11591/IJEECS.V26.I3.PP1624-1633.

[3] A. Mughaid et al., "A novel machine learning and face recognition technique for fake accounts detection system on cyber social networks," Multimed Tools Appl, vol. 82, no. 17, pp. 26353–26378, Jul. 2023, doi: 10.1007/S11042-023-14347-8/METRICS.

[4] K. Sethi and V. Jaiswal, "PSU-CNN: Prediction of student understanding in the classroom through student facial images using convolutional neural network," Mater Today Proc, vol. 62, pp. 4957–4964, Jan. 2022, doi: 10.1016/J.MATPR.2022.03.691.

[5] R. B. R. Maddu and S. Murugappan, "Online learners' engagement detection via facial emotion recognition in online learning context using hybrid classification model," Soc Netw Anal Min, vol. 14, no. 1, pp. 1–19, Dec. 2024, doi: 10.1007/S13278-023-01181-X/METRICS.

[6] M. A. Turk and A. P. Pentland, "Face Recognition Using Eigenfaces," in 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1991, pp. 586–591.

[7] T. Ojala, M. Pietika, and T. Ma, "Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns," IEEE Trans Pattern Anal Mach Intell, vol. 24, no. 7, pp. 971–987, 2002, doi: 10.1109/tpami.2002.1017623.

[8] B. Schölkopf and A. J. Smola, Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2002.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Commun ACM, vol. 60, no. 6, pp. 84–90, Jun. 2017, doi: 10.1145/3065386.

[10] Y. Sun, X. Wang, and X. Tang, "Deep Learning Face Representation from Predicting 10,000 Classes," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 1891–1898.

[11] G. Lou and H. Shi, "Face image recognition based on convolutional neural network," China Communications, vol. 17, no. 2, pp. 117–124, Feb. 2020, doi: 10.23919/JCC.2020.02.010.

[12] K. Li, Y. Jin, M. W. Akram, R. Han, and J. Chen, "Facial expression recognition with convolutional neural networks via a new face cropping and rotation strategy," Visual Computer, vol. 36, no. 2, pp. 391–404, Feb. 2020, doi: 10.1007/S00371-019-01627-4/METRICS.

[13] B. Jin, L. Cruz, and N. Goncalves, "Deep Facial Diagnosis: Deep Transfer Learning from Face Recognition to Facial Diagnosis," IEEE Access, vol. 8, pp. 123649–123661, 2020, doi: 10.1109/ACCESS.2020.3005687.

[14] M. K. Rusia and D. K. Singh, "A comprehensive survey on techniques to handle face identity threats: challenges and opportunities," Multimedia Tools and Applications 2022 82:2, vol. 82, no. 2, pp. 1669–1748, Jun. 2022, doi: 10.1007/S11042-022-13248-6.

[15] A. Aglasia, S. Y. Irianto, S. Karnila, and D. Yuliawati, "Image Sketch Based Criminal Face Recognition Using Content Based Image Retrieval," Scientific Journal of Informatics, vol. 8, no. 2, pp. 176–182, Nov. 2021, doi: 10.15294/SJI.V8I2.27865.

[16] N. Kadek Ayu Wirdiani, P. Hridayami, N. Putu Ayu Widiari, K. Diva Rismawan, P. Bagus Candradinatha, and I. Putu Deva Jayantha, "Face Identification Based on K-Nearest Neighbor," Scientific Journal of Informatics, vol. 6, no. 2, pp. 150–159, Nov. 2019, doi: 10.15294/SJI.V6I2.19503.

[17] L. Boussaad and A. Boucetta, "Deep-learning based descriptors in application to aging problem in face recognition," Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 6, pp. 2975–2981, Jun. 2022, doi: 10.1016/J.JKSUCI.2020.10.002.

[18] A. Graves, S. Fernández, and J. Schmidhuber, "Multi-dimensional recurrent neural networks," in International conference on artificial neural networks, 2007, pp. 549–558.

[19] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv preprint arXiv:1409.0473, 2014.

[20] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. MIT press, 2016.
[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.
[22] R. Girshick, "Fast r-cnn," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.
[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
[24] S. W. Smith and others, "The scientist and engineer's guide to digital signal processing." California Technical Pub. San Diego, 1997.
[25] S. B. Damelin and W. Miller, The mathematics of signal processing, no. 48. Cambridge University Press, 2012.
[26] F. Tariq, "Breaking Down the Mathematics Behind CNN Models: A Comprehensive Guide," Medium. Accessed: Dec. 18, 2023. [Online]. Available: https://medium.com/@beingfarina/breaking-down-the-mathematics-behind-cnn-models-a-comprehensive-guide-1853aa6b011e
[27] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," Adv Neural Inf Process Syst, vol. 28, 2015.
[28] G. Maguolo, L. Nanni, and S. Ghidoni, "Ensemble of convolutional neural networks trained with different activation functions," Expert Syst Appl, vol. 166, p. 114048, 2021.
[29] A. K. Dubey and V. Jain, "Comparative study of convolution neural network's relu and leaky-relu activation functions," in Applications of Computing, Automation and Wireless Systems in Electrical Engineering: Proceedings of MARC 2018, 2019, pp. 873–880.
[30] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807–814.
[31] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, pp. 315–323.
[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[33] A. Géron, Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. " O'Reilly Media, Inc.," 2022.
[34] A. Shatnawi, G. Al-Bdour, R. Al-Qurran, and M. Al-Ayyoub, "A comparative study of open source deep learning frameworks," in 2018 9th international conference on information and communication systems (icics), 2018, pp. 72–77.
[35] M. Abadi et al., "TensorFlow: a system for Large-Scale machine learning," in 12th USENIX symposium on operating systems design and implementation (OSDI 16), 2016, pp. 265–283.
[36] M. Abadi et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," arXiv preprint arXiv:1603.04467, 2016.