Journal of Informatics and Web Engineering

Vol. 2 No. 2 (September 2023) eISSN: 2821-370X

Face and Facial Expressions Recognition System for Blind People using ResNet50 Architecture and CNN

Jia-Rou Lee[1], Kok-Why Ng[2*], Yih-Jian Yoong[3]

[1,2,3]Faculty of Computing and Informatics, Multimedia University, Persiaran Multimedia, 63100 Cyberjaya, Selangor, Malaysia.

**corresponding author: (kwng@mmu.edu.my, ORCiD: 0000-0003-4516-4634)*

Abstract - Many blind individuals have difficulties in recognizing people's facial expression which may impact their social interaction. With the recognition, the blind individuals can accurately interpret and respond to the emotions. There is a lack in the existing application with the combination of face and facial expressions recognition. The blind individuals have to rely on multiple applications to accomplish the same task, making it difficult and time-consuming for them to use. The paper aims to recognize faces and facial expressions for blind individuals and provides feedback in real-time. Three face detection algorithms of Haar Cascade Classifier, Dlib, and RetinaFace are compared. Dlib is chosen to process with Histogram of Oriented Gradients (HOG) and Support Vector Machine (SVM). It loads the pre-trained model, computes the HOG features, slide the window scanning at different scales, classify the windows using the SVM classifier, generate bounding boxes, and applying non-maximum suppression. ResNet50 architecture is employed to recognize face and Convolutional Neural Networks (CNN) is applied to recognize facial expression. The training accuracy is 70% and validation accuracy is 60%.

Keywords— Face Recognition, Facial Expressions Recognition, ResNet50, Dlib, Convolutional Neural Networks, Model Training

Received: 13 June 2023; Accepted: 20 August 2023; Published: 16 September 2023

I. INTRODUCTION

According to current estimates, there are 285 million people globally who are visually impaired, with 39 million being blind and 246 million having low vision. Especially blind people, they live a life in darkness and cannot experience life like a normal person. Blindness greatly hinders a broad range of daily activities, including recognising faces and facial expressions. When blind people do not have the ability to recognise known human individuals, this will make it more difficult for them to understand social nuances and minimise their opportunities to acquire appropriate social skills.

Journal of Informatics and Web Engineering
https://doi.org/10.33093/jiwe.2023.2.2.20
© Universiti Telekom Sdn Bhd. This work is licensed under the Creative Commons BY-NC-ND 4.0 International License.
Published by MMU Press. URL: https://journals.mmupress.com/jiwe

Effective communication is crucial for maintaining relationships. When two sighted people interact, a significant amount of information is conveyed through nonverbal cues such as gestures and facial expressions. Facial expression is widely linked to emotion and gives insight into the message being conveyed. Without sight, blind people do not understand people's emotion during conversation with them. When there is any silence during the conversation, blind people will tend to feel uncomfortable because they cannot see facial expression.

This paper applies ResNet50 Architecture to perform face recognition. ResNet50 can learn intricate features from large datasets and generalize well to unseen data. Its depth enables it to capture complex patterns, and its pre-trained weights from image classification tasks aid transfer learning. As compared to existing approaches such as FaceNet and VGGFace, Besides, Convolutional Neural Networks (CNN) to perform facial expression recognition. The networks use multiple layers of convolutions to extract and learn relevant patterns and features from images. The learned features are then to be employed for classification tasks for facial expression recognition. CNNs excel in capturing complex and abstract features, making them highly effective in tasks that require high-level feature representations like facial expression recognition. As compared to existing approaches such as traditional methods like Histogram of Oriented Gradients (HOG) and Local Binary Patterns (LBP) rely on handcrafted features, which might not capture the richness of facial expression patterns as effectively as CNN. In terms of generalization, CNN has demonstrated better generalization capabilities, enabling them to adapt to various expression variations and unseen data. In contrast, HOG and LBP may not generalize well to unseen expressions or diverse facial appearances.

Next section will look into the literature review of the existing work. Section III is the proposed methodology. Section IV is the result generated by this paper. Section V will discuss the result before the conclusion on Section VI.

II. LITERATURE REVIEW

A. Facial Expressions Recognition Techniques

In the works done by Islam et al. [1], the authors presented a novel approach for emotion recognition from facial expressions. Their method involved a combination of Histogram of Oriented Gradients (HOG) and Local Binary Patterns (LBP) to extract features from preprocessed input images. To effectively analyze the facial region, the authors proposed a segmentation method that divided it into four expression regions. Feature extraction was performed on these segmented regions using a fusion of HOG and LBP techniques. To reduce the feature vector's dimensionality, the authors employed Principal Component Analysis (PCA). For classification, a multiclass Support Vector Machine (SVM) was utilized. The performance evaluation of their method was conducted on three well-known datasets: JAFFE, CK+, and RaFD.

Joseph et al. [2] developed a Facial Expression Recognition model using the sequential model in Keras. The model comprised 10 convolutional layers, each paired with a corresponding max pooling layer. The number of kernels in each pair of convolutional layers increased incrementally, with 16 in the first pair, 32 in the second, 64 in the third, 128 in the fourth, and finally 256 in the last pair. Training the model on the FER-2013 dataset resulted in an accuracy of 67.18% after 150 epochs. The model successfully classified input images into seven categories, which included happy, sad, angry, surprise, disgust, fear, and neutral expressions.

Lin et al [3] proposed a continuous facial expression recognition model that focused on identifying emotion patterns over time. They combined Convolutional Neural Network (CNN) for image recognition with an enhanced version of Recurrent Neural Networks (RNN) known as Long Short-Term Memory (LSTM) to capture temporal properties in the data. The training data, sourced from the FER2013 dataset, underwent preprocessing and was used to train a basic facial expression classification model based on the CNN AlexNet architecture. To build the continuous facial expression classification model, the author utilized another dataset called RAVDESS from Zenodo, which was also preprocessed and fed into the previously trained facial expression classification model. The softmax layer in this model generated vectors for all the facial expression image frames. These vectors were then used as input data for the LSTM model, which aimed to identify patterns of emotional changes over time, enabling continuous facial expression recognition. The results showed that the normal facial expression classification model achieved an accuracy of 43.1%, while the their proposed continuous facial expression classification model achieves a significantly improved accuracy of 74.07%.

Liu et al. [4] introduced a novel approach for Facial Expression Recognition (FER) in videos by incorporating a Graph Convolutional Network (GCN) layer into a CNN-RNN based model. This method utilized the GCN layer to extract

facial expression features that are more relevant and focus on specific regions. The sharing of information between the CNN features of different nodes allowed for improved feature extraction. The learned features from the GCN layer were then passed through a LSTM layer to capture long-term dependencies and model variations in the video sequences. To enhance the classification performance, a weight assignment mechanism was introduced, which assigned weights to the output of different nodes based on the intensity of expressions present in each frame. The proposed model was evaluated on three datasets: CK+ dataset, Oulu-CASIA dataset, and MMI dataset.

Supta et al. [5] proposed an automated Facial Expression Recognition (FER) system that relied on Histogram of Oriented Gradients (HOG) and Support Vector Machine (SVM). This system was designed to recognize facial expressions in both static images and real-time scenarios. The process involved several steps: firstly, the system detected the face and its individual facial parts in the input image. Subsequently, preprocessing techniques such as histogram equalization and image sharpening were applied to mitigate illumination effects. Then, HOG was utilized to extract distinctive features from these facial regions, which were later combined into a single feature vector. Finally, the system employed SVM with a polynomial kernel function for accurate expression classification. The proposed FER system was evaluated on well-known databases, namely JAFFE and Cohn-Kanade, which contained seven basic facial expressions.

Xu et al. [6] applied Graph Convolutional Neural Network (GCNN) for feature extraction and classification. GCNN classifier was implemented using PyTorch deep learning framework. Statistical information of local neighbourhood was calculated as a single vertex to reduce the complexity of the model in the output of pool layer. As a result, some of the output failed due to similar motion of mouth or facial expressions.

Ullah et al. [7] conducted a comparative study between two facial landmark detection and feature extraction methods. The first approach employed traditional image processing techniques, such as histogram equalization, thresholding, color conversion, and morphological operations. The second method utilized the Dlib library for facial landmark detection. Both methods were evaluated using two classification algorithms: Support Vector Machine (SVM) and Multi-layer Perceptron (MLP). The evaluation was carried out on three facial expression databases: 10k US Adult Faces Database, the MUG Facial Expression Database, and a personal database. The main objective was to classify three different facial expressions: happiness, surprise, and neutrality. The pros and cons of the existing facial expressions recognition methods are summarized in Table 1.

B. Face Recognition Techniques

Convolutional Neural Network (CNN) was employed in facial recognition, utilizing the VGG16 architecture to create an unique set of weights for each face that becomes its representation for recognition purposes. The CNN was trained to differentiate between the relevant class for recognition and irrelevant class for artifact modeling in video streams. This led to an average accuracy rate of 90%. The training process for the CNN was crucial for the success of network training and the final recognition rate [9].

Local Binary Patterns Histogram (LBPH) algorithm [10] was further discussed and explored. The Local Binary Pattern Histogram (LBPH) algorithm had the capability to identify and classify images of both front and side faces within a given dataset. When it came to extracting facial features, the Local Binary Pattern (LBP) operation was employed to generate an image that effectively emphasized the distinctive qualities of the original image.

The EigenFace, Fisher algorithm, and LBP algorithms were employed for face recognition [11]. In OpenCV, there were adjustable parameters such as the recognition threshold and feature points that could be set to improve the accuracy of face recognition. The EigenFace algorithm demonstrated a higher rate of recognition for standard images. When the training data was moderate, both the EigenFace and LBP algorithms exhibited relatively high recognition rates. However, if the training data was limited, the accuracy of the EigenFace algorithm might experience a slight decrease.

Table 1. Pros And Cons For The Existing Facial Expressions Recognition Techniques

Existing Methods	***Pros***	***Cons***
LBP	• Low calculation cost • Resistance to fluctuations in image gray scale values	• Limited discriminative power • Sensitivity to occlusion • Lack of interpretability
HOG	• Low false positive rate • Relatively fast • Low computational complexity	• Limited to grayscale images • Sensitivity to orientation • Limited to facial recognition
GCNN	• Able to learn complex relationships between facial features and expressions • Able to handle large dan high-dimensional data • Can be trained on a large dataset and generalize well on unseen data	• Require significant computational resources and a substantial volume of training data • Prone to being affected by changes in lighting, pose, and other related factors • Can be affected by overfitting
CNN	• High accuracy in terms of recognition • Have the ability to autonomously learn and extract features from images. • Can be trained on large datasets	• High computational cost • Slow to train for complex tasks
SVM	• Performs well in high-dimensional feature spaces, making it suitable for complex datasets with many features • Performs well even with limited training data.	• Can be computationally expensive, especially when dealing with large datasets or a high number of features • May struggle with imbalanced datasets

Sarwar et al [12] studied on the use of facial recognition technology to assist visually impaired individuals. The authors proposed a face recognition system that used the Local Binary Patterns Histograms (LBPH) algorithm. The authors of the paper claimed that their proposed system improved the accuracy of face recognition for visually impaired individuals compared to traditional methods, and that it could also be used to help individuals with other disabilities such as Down Syndrome, facial paralysis, and aging. The proposed system in the study used the LBPH algorithm for face recognition and was tested on two datasets. The first dataset contained images of visually impaired individuals and the system achieved an accuracy of 96.15%. The second dataset contained images of people with Down Syndrome, and the system achieved an accuracy of 95.5%. These results were promising and indicated that the LBPH algorithm might be an effective method for facial recognition for people with disabilities. The authors suggested that this technology could be used to improve accessibility and independence for visually impaired individuals in various settings, such as in homes, workplaces, and public spaces.

Naik et al. [13] investigated the application of the LBPH algorithm for facial recognition. The authors proposed an implementation of the LBPH algorithm that could be run on a Graphics Processing Unit (GPU) in order to improve the speed and efficiency of the face recognition process. The authors of the paper claimed that their proposed implementation of the LBPH algorithm on a GPU was able to improve the speed of the face recognition process by up to 40 times compared to traditional methods that used a Central Processing Unit (CPU). Additionally, the authors claimed that the proposed system was able to achieve high accuracy rates for both frontal and side profile face recognition. The study included a dataset that included images of faces captured from both frontal and side profiles,

and the results showed that the proposed system had an accuracy of 98.5% for frontal faces and 96.5% for side profiles. The study's results were promising and suggested that the LBPH algorithm could be an effective method for facial recognition, and that implementing it on a GPU could greatly improve the speed and efficiency of the process. The authors suggested that this technology could be used in various applications such as security systems, human-computer interaction and biometric identification systems.

Flores et al. [14] discussed the application of Local Binary Patterns Histograms (LBPH) for real-time face recognition. The authors presented a system combining LBPH for feature extraction and k-Nearest Neighbors (k-NN) classification to recognize faces in real-time with high accuracy. It handled variations in lighting, pose, and facial expressions and performed well with small datasets, making it suitable for practical use. Results from experiments using a 40-person dataset showed an accuracy of 98%. The system also achieved high accuracy on ORL and Yale face databases with 96% and 93% respectively. The study's results were promising and suggested that the LBPH algorithm combined with k-NN classifier could be an effective method for real-time facial recognition.

Wang et al. [15] presented a combination of Fisherface and machine learning techniques for improved facial recognition accuracy. The authors proposed using Fisherface with machine learning methods such as SVM and k-NN to enhance recognition performance. This combination reduced face image dimensionality while maintaining important facial features. The study results showed that the proposed algorithm had an accuracy of 98.26%, compared to 96.3% for the traditional Fisherface method. The proposed algorithm also performed well on standard datasets, ORL and Yale face, outperforming other state-of-the-art methods. The results suggested that the Fisherface-machine learning combination could be effective for facial recognition and had potential applications in security, biometric identification, and human-computer interaction.

Rosnelly et al. [16] presented a study on the use of the Eigenface algorithm for facial recognition using a laptop camera. The authors of the paper claimed that their proposed system improved the accuracy of face recognition compared to traditional methods when using a laptop camera, which typically had lower resolution and worse lighting conditions than other types of cameras. The study focused on using the Eigenface algorithm for facial recognition through a laptop camera. The results showed an accuracy of 96.5% using a dataset of face images captured by the laptop camera. The Eigenface algorithm, being one of the first and most straightforward methods, represented an image by calculating the eigenvectors of the covariance matrix of a set of images. The results were encouraging and suggested that the Eigenface algorithm could be an efficient solution for facial recognition using a laptop camera. The authors suggested that this technology could be used in various settings, such as in homes, workplaces, and public spaces where the use of a laptop camera was common.

Table 2. Pros And Cons For The Existing Face Recognition Techniques

Existing Methods	**Pros**	**Cons**
CNN	• High accuracy • Robust to variations • Good at handling large and complex datasets	• High computational cost • Requires large amount of data
LBPH	• Good texture descriptor performance • Image can be analyzed independently • Resistant to single-directional changes in the grayscale of an image • Has a low computational complexity • Able to recognise both side and front faces	• Generates lengthy histograms which results in a slower recognition speed. • Not able to recognize faces that are rotated in the image.
EigenFace	• better performance involving small databases or training sets	• Prone to variations in lighting

	• Raw intensity data are used directly for learning and recognition	• Vulnerable to inaccuracies in pixel alignment • Susceptible to changes in pose and facial expressions
Fisher algorithm	• Able to project high-dimensional data onto a lower-dimensional space • Simple to implement and computationally efficient. • Do not require any assumption about the underlying data distribution	• Sensitive to outliers and noise in the data • Sensitive to the scaling of the data
ResNet50	• High recognition accuracy • Can differentiate between individuals with high precision • Suitable for real-time or near real-time face recognition applications	• Requires significant computational resources for training and inference • Can be sensitive to the quality of the training data

Mohite et al. [17] examined the use of thermal and visual facial recognition technology to enhance the accuracy of face recognition in various lighting and environmental conditions. The authors proposed a system that employed Eigenfaces and Transfer learning algorithms for face recognition. The authors used a pre-trained model trained on a large dataset of facial images, fine-tuning it with thermal and visual facial images to enhance its performance. The study claimed that their system led to improved face recognition accuracy compared to traditional methods, especially under varying lighting and environmental conditions. The system was tested using a dataset of thermal and visual face images, with results showing an accuracy of 99.2%, which was considered high. The study's results were promising and suggested that the combination of Eigenfaces and Transfer learning algorithm could be an effective method for facial recognition in different lighting and environmental conditions. The authors suggested that this technology could be used in various settings such as security, surveillance, and access control.

Winarno et al. [18] presented a study on the use of facial recognition technology to create an attendance system. The authors proposed utilizing a combination of Convolutional Neural Networks (CNN) and Principal Component Analysis (PCA) to enhance the accuracy of face recognition. They claimed that by incorporating CNN-PCA, the recognition performance could be optimized in real-time camera scenarios where lighting conditions might not be optimal. The experiment involved a dataset of student images and yielded an accuracy of 98.87%. The system was also tested on a dataset that included images of students in varied lighting situations and achieved an accuracy of 96.5%. The pros and cons of the existing face recognition methods are summarized in Table 2.

III. RESEARCH METHODOLOGY

A. Convolutional Neural Network (CNN)

This paper applies CNN to extract relevant features from facial images. It composes of multiple layers of interconnected neurons, which processes the input data and learns to extract features at each layer [8][19]. The feature extraction layer is applied to identify specific patterns in the input data. The filters are implemented by performing a convolution operation on the input, which slides the filter over the input data and calculates the dot-product between the filter and the input data at each position. This process extracts unique features from the image and produces a feature map. The convolution operation is defined in Equation (1).

$$X_j^l = f\left(\sum_{i\in M_j} X_i^{l-1} * K_{ij}^l + b_j^l\right) \quad (1)$$

where X_j^l represents the j characteristic pattern of the l layer. K_{ij}^l is convolution kernel function, $f(\cdot)$ is activation function, b_j^l is offset parameters.

The C1 convolutional layer applies a 5*5 convolution filter to the input image with a size of 96*96 pixels. This results in a feature map size of (96-5+1)*(96-5+1) = 92*92, with each neuron covering a 5*5-pixel local sensing area. By using 32 different convolution kernels, 32 different local expression features are extracted, resulting in 32 feature maps. Each neuron in the same feature map shares the same weight, but is fed with input from different local receptive fields. The C2 layer performs convolution on the feature maps produced by the C1 layer, using 5*5 convolution kernels, producing 64 feature maps with a size of (92-5+1)*(92-5+1) = 88*88. The C3 convolutional layer then takes the feature maps produced by the pooling layer S1 and performs convolution with 128 5*5 convolution filters, resulting in 128 feature maps with a size of (44-5+1)*(44-5+1) = 40*40.

As the number of convolutional layers increases, so does the dimensionality of the learned features. This can cause a problem when using all of the features to train the Softmax classifier. To address this issue, a pooling layer is added after the convolutional layer to reduce the input data dimensionality. This is achieved through max pooling. The pooling layer reduces the complexity of the model, prevent overfitting by reducing the feature dimension, and make the model more robust to small changes in the input data by filtering out noise and irrelevant information. The pooling process is depicted in Equation (2).

$$X_j^l = f(\beta_j^l down\left(X_j^{l-1}\right) + b_j^l) \quad (2)$$

In the pooling layer S1, the feature maps output from convolution layer C2 are reduced in size by using a 2*2 window, resulting in feature maps of size 44*44 and a total of 64 feature maps remaining after downsampling. Similarly, in the pooling layer S2, the feature maps output from convolution layer C3 undergo downsampling, resulting in 128 feature maps with a size of 20*20.

Next, a dense layer takes the output from the pooling layer and combines the input features using a linear combination of learned weights and biases to perform classification. The output from the pooling layer is transformed into a one-dimensional array, which is then multiplied by a weight matrix to produce a new set of features. A bias term is added to the new features and an activation function is applied, resulting in the final output of the fully connected layer. This output is then passed onto the softmax layer classify the input data.

Lastly, the Softmax Classifier evaluates the likelihood of an input belonging to each class. It has neurons that emit a score between 0 and 1, which represents the possibility of the input belonging to that specific class. The class with the highest probability score is chosen as the final prediction.

The slight changes in the output of the previous layer caused by the activation function can have a compounding impact on the output of subsequent layers during training, leading to a shift in the distribution of both the training and test sets. To address this issue, this paper uses batch normalization to normalize the inputs, bringing their distributions closer to a standard normal distribution with a mean of 0 and a variance of 1. This effectively combats gradient disappearance, improving both the training process and the accuracy of the classifier.

An Linear Correction Unit (LCU) takes the output of a layer and performs a linear correction on it by adding a bias term and scaling the result by a learnable factor. The correction helps adjust the output distribution so that it is better suited for the activation function used in the next layer. The bias and scale parameters are learned through backpropagation during the training process, allowing the network to adapt to the input data.

To mitigate overfitting and improve generalization in Convolutional Neural Networks (CNNs), Dropout is employed to randomly dropping out a specified percentage of neurons in each layer during each forward pass. The dropout probability, which ranges from 0.5 to 0.8, is a tunable hyperparameter that can be adjusted for optimal performance.

B. Pre-trained Model using Dlib

During the initial stages of application development, the Haar Cascade Classifier was utilized for face detection. This classifier is well-known and has been used for a long time. However, several issues arose when attempting to import cropped videos and apply face detection. The Haar Cascade Classifier tended to generate numerous false positive detections, incorrectly identifying non-human faces as human faces. Additionally, it was limited to detecting only frontal faces, making it incapable of detecting faces at odd angles. Consequently, it was challenging to identify videos where the classifier could accurately detect faces in each extracted frame. To address this problem, alternative face detection algorithms were thoroughly studied and examined from various perspectives. Three face detection algorithms were compared: Haar Cascade Classifier, Dlib, and RetinaFace. The comparison is shown in Table 3.

Table 3. Comparison on The Three Face Detection Algorithms

Aspects	Haar Cascade Classifier	Dlib (HOG + SVM)	RetinaFace
Speed (Per Image)	0.19s	0.28s	6.18s
Work Under Occlusion	No	Yes (Slightly)	Yes
Accuracy	8/10	9/10	9/10
Sensitive To Lighting	Yes	Yes (Slightly)	No
False Positives	Yes	No	No

Overall, the Dlib face detection outperforms the remaining algorithms. RetinaFace was the first to be excluded from this project due to its long processing time. Its speed of approximately 6 seconds to process a single image is too slow and reduce the effectiveness of the application. Next, the Haar Cascade Classifier was also removed from the choices due to its high false positives rate. Also, it is sensitive to lighting and it cannot work well under occlusion such as hand covering. This increases the chance of detecting false positives in the real-time videos. Dlib is the best out of the three face detection algorithms. Thus, Dlib was chosen as the face detection method in this project.

C. Face Detection process of Dlib Library

The detection process using the Dlib library with Histogram of Oriented Gradients (HOG) and Support Vector Machine (SVM) is a systematic and efficient way of detecting faces in an image. Firstly, a trained HOG and SVM face detection model is loaded from the dlib library. This model has been previously trained on a large dataset of positive and negative examples to learn the distinguishing characteristics of faces.

To identify faces in an input image, the process involves several steps. First, the input image is converted to grayscale to facilitate efficient processing. Next, the HOG (Histogram of Oriented Gradients) feature descriptor is computed for the image. The HOG features capture information about the local gradient orientations, offering insights into the image's object shapes and textures. Computation of the HOG features involves dividing the image into small cells, computing the gradient orientations within each cell, and constructing histograms based on these orientations.

Next, a sliding window approach is used to scan the image at different scales and positions. At each window location, the HOG features are extracted. These features are then fed into the trained SVM classifier, which determines the likelihood of the window containing a face. The SVM classifier has learned to separate face and non-face regions based on the HOG features.

To efficiently process the image, the detection is performed in a pyramid fashion, starting from a smaller scale and gradually increasing it. This approach allows for detecting faces at different sizes in the image. The scaling factor and window stride can be adjusted to control the size of the sliding window and the step size during the scan, respectively. During the sliding window scan, if the SVM classifier predicts a positive detection, indicating the presence of a face, a bounding box is generated around that region. Multiple detections may be generated for a single face due to

overlapping windows and different scales. To eliminate duplicate detections and refine the final bounding boxes, a non-maximum suppression algorithm is applied. This algorithm removes redundant and overlapping bounding boxes by keeping only the most confident detection in each region.

In summary, the face detection process using dlib's HOG and SVM approach involves loading the pre-trained model, computing the HOG features, sliding window scanning at different scales, classifying the windows using the SVM classifier, generating bounding boxes, and applying non-maximum suppression. This process allows for efficient and accurate detection of faces in images using the combined power of HOG features and SVM classification provided by the dlib library.

D. Pre-Trained Face Recognition Model Using ResNet50 Architecture

The face recognition model is being pre-trained by using ResNet50 architecture and Dlib. Firstly, a dataset of labeled face images is collected, ensuring diversity in individuals and capturing variations in pose, lighting, and expression. These images are then preprocessed to standardize sizes, align faces to a canonical pose, and normalize pixel values. Preprocessing helps reduce variations and enhances the model's ability to generalize to unseen data.

Next, Dlib's face detection and alignment capabilities are utilized to detect and align the faces in the dataset. This step ensures that all faces are properly aligned and centered, which is crucial for accurate feature extraction. Afterwards, the aligned face images are processed using a pre-trained ResNet50 model, which has been trained on extensive image datasets. This model utilizes deep convolutional neural networks to extract comprehensive representations of facial features. By leveraging the ResNet50 model, the system can learn and capture intricate characteristics of the face images. Each face image is passed through the model, resulting in a high-dimensional feature representation, commonly a 128-dimensional embedding.

The labels associated with each face image are encoded using suitable encoding techniques, converting categorical labels into numerical representations. These encoded labels are then used, along with the extracted face features, to train a classifier or neural network. The classifier learns to map the high-dimensional face embeddings to their respective labels, enabling the model to recognize and differentiate between different individuals.

When using a distance threshold of 0.6, the ResNet50 with dlib model obtains an accuracy of 99.38% on the standard LFW face recognition benchmark.

E. Classification Model Training

CNN model is designed and implemented, which involves selecting the appropriate layers, filters, and other hyperparameters. In terms of batch size which is an important hyperparameter that determines the number of samples used in each iteration to update the model’s parameters, a smaller batch size will result in faster convergence and in this case, batch size will be 64. Training and validation sets are created using the ImageDataGenerator from Keras library, which provides real-time data augmentation to increase the size and diversity of the data in order to avoid overfitting and ensure the variation in training data [20]. To construct the model, several 2D CNN Layers which are BatchNormalization, Activation(ReLU), MaxPooling and Dropout Layer are added to the model. After implementation, the model is compiled using Adam optimizer and trained with the training set by inputting the training set into the model and adjusting its weights and biases to lower the error between predicted and actual expressions.

During the training process, the model's performance will be monitored using various metrics, such as accuracy and loss. This can be done by evaluating the model on the test set and comparing the predicted expressions to the true expressions. If the model is not performing well, adjustments can be made to the architecture or hyperparameters to improve its performance.

Once the model is trained, it can be tested on the validation set to evaluate its performance on unseen data. If the model performs well on the validation set, it can be considered ready for deployment. If not, further training or adjustments may be necessary before deploying the model.

As compared to the existing CNN models in recognizing the facial expressions, the proposed approach utilizes and adjusted the 2D CNN layers to be added to the model to obtain higher recognition accuracy. In terms of recognition

accuracy, the inclusion of BatchNormalization, ReLU activation and dropout layers further enhances the model's ability to capture complex patterns and representations, leading to improved recognition accuracy. The dropout layer acts as a regularization technique, preventing overfitting during training.It randomly drops out some neurons' activations, reducing interdependence among neurons and making the model more robust and less likely to memorize noise in the training data. BatchNormalization stabilizes the training process by normalizing the activations, allowing for higher learning rates and faster convergence. This can speed up the training process and make the model more efficient. In terms of speed and real-time performance, the combination of ReLU activations and max pooling layers helps to create a more computationally efficient model. It reduces the spatial dimensions of feature maps and captures essential information, making the model suitable for real-time applications. The proposed model's architecture, which includes activation functions like ReLU and max pooling layers, allows for effective feature learning. It can capture both local and global features in facial images, providing a holistic representation of expressions and enhancing recognition performance.

F. Text to Speech Conversion Using Google Text to Speech API

In the context of assisting blind to recognize the facial expressions of people, an audio message is generated as the output. In this case, once the program captures people's facial expression and the output is being generated from the program, it will convert the output from text to speech. At the same time, the blind people will notice the audio and recognize that particular people's facial expression. The output will be generated based on a certain buffer time which is around 20 seconds. In terms of text-to-speech conversion, Google Text to Speech API will be used because it offers a wide range of high-quality and natural-sounding voices in different languages and accents. It is also cloud-based which means that the text-to-speech processing is handled on Google's servers. This reduces the computational burden and ensures that the TTS service can scale efficiently based on demand.

IV. RESULTS AND DISCUSSIONS

Figure 1 shows 9 images in grid using the Matplotlib library and the images are 48x48 pixels and are loaded from a directory with the expression "disgust". The images are displayed in 3x3 grid using a loop and the subplot function from Matplotlib.

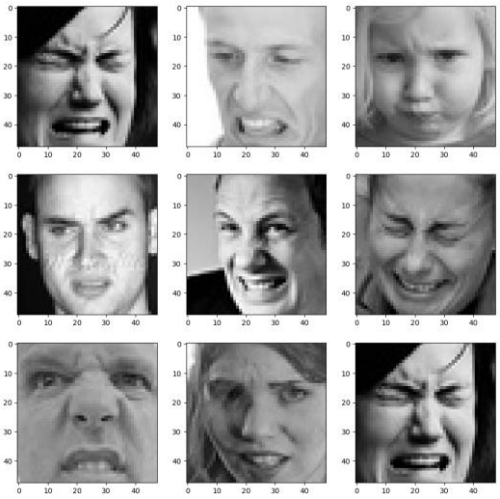

Figure 1. Displaying 9 Images From "Disgust" Directory

Figure 2 shows the final progress of the model training. It iterates 19 times of the training the model on the entire training dataset and shows the completion of processing all 1369 batches in the current epoch. The "loss" and "accuracy" are the performance metrics of the model on the training data. The lower the loss and higher the accuracy, the better the model is performing. In this case, the loss is 0.6625 and the accuracy is 0.7537, which means the model is doing a relatively good job in predicting the labels for the training data.

The val_loss and val_accuracy are the performance metrics of the model on the validation data. The model uses these metrics to avoid overfitting, which is when the model performs well on the training data but poorly on new unseen data. In this case, the validation loss is 1.1403 and the validation accuracy is 0.5986.

The lr represents the learning rate, which is a hyperparameter that determines how much the model updates its parameters in each iteration. A higher learning rate means the model updates its parameters more aggressively, and a lower learning rate means the model updates its parameters more slowly. In this case, the learning rate is 1.0000e-04, which is a very low value, indicating that the model is making small updates to its parameters.

```
Epoch 19: ReduceLROnPlateau reducing learning rate to 1.9999999494757503e-05.
1369/1369 [==============================] - 142s 104ms/step - loss: 0.7901 - accuracy: 0.7028 - val_loss: 1.1403 - val_accur
acy: 0.5986 - lr: 1.0000e-04
```

Figure 2. 19th Iteration Of Model Training (Final Iteration)

Figure 3, Figure 4, Figure 5 and Figure 6 display the sample output of the face recognition result and predicted facial expressions recognition from the program, including emotions of happy, sad and angry. The face recognition result and predicted emotions are demonstrated to be accurate in their representation.

Figure 3. Sample Output Of One Happy Man called Mark

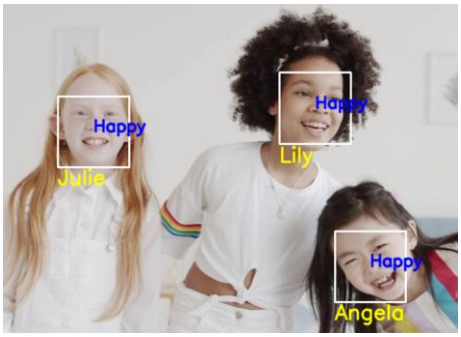

Figure 4. Sample Output Of Three Happy Kids Called Julie, Lily And Angela

Figure 5. Sample Output Of A Angry Woman Called Susan

Figure 6. Sample Output Of A Sad Woman Called Sally

Figure 7 shows the loss graph and accuracy graph of CNN model by using Adam Optimizer. From the loss graph, it plots the values of the loss function for each training epoch. The loss function measures how well the model is doing at predicting the true labels for the training data. The objective of training is to minimize the loss, so a decrease in the loss over time indicates that the model is learning and improving its predictions. As shown in the loss graph, training loss decreases gradually from 1.65 to 0.79, indicating the model is improving its ability to predict the correct outputs based on the inputs over time. This decrease in training loss indicates that the model is learning the correct relationships between inputs and outputs, and is reducing the amount of error it makes in its predictions. The gradual decrease in loss is a positive sign that the model is being trained effectively, and that the optimization algorithm is effectively updating the model parameters to minimize the loss. As for validation loss, it decreases from 1.65 to 1.07 and starts to increase to 1.14. The decrease in validation loss from 1.65 to around 1.07 means that the model is becoming more and more accurate in its predictions on the validation data. This is a positive sign as it suggests that the model is generalizing well to new, unseen data.

From the accuracy graph, the training accuracy is increasing gradually from 0.34 to 0.70. The increase in training accuracy from 0.34 to 0.70 indicates that the model has improved its performance on the training data over the course of the training process. This improvement in accuracy is likely the result of the optimization algorithm, such as Adam, adjusting the model's weights and biases to minimize the training loss. The fact that the accuracy has increased to around 0.70 means that the model is now able to correctly classify or predict around 70% of the training data. As for validation accuracy, it increased gradually from 0.33 to 0.6 and start to decrease to 0.59. This indicates that the model improved its accuracy in predicting the validation set, reaching a plateau after a certain point. Further analysis and adjustments to the model may be necessary to improve its performance.

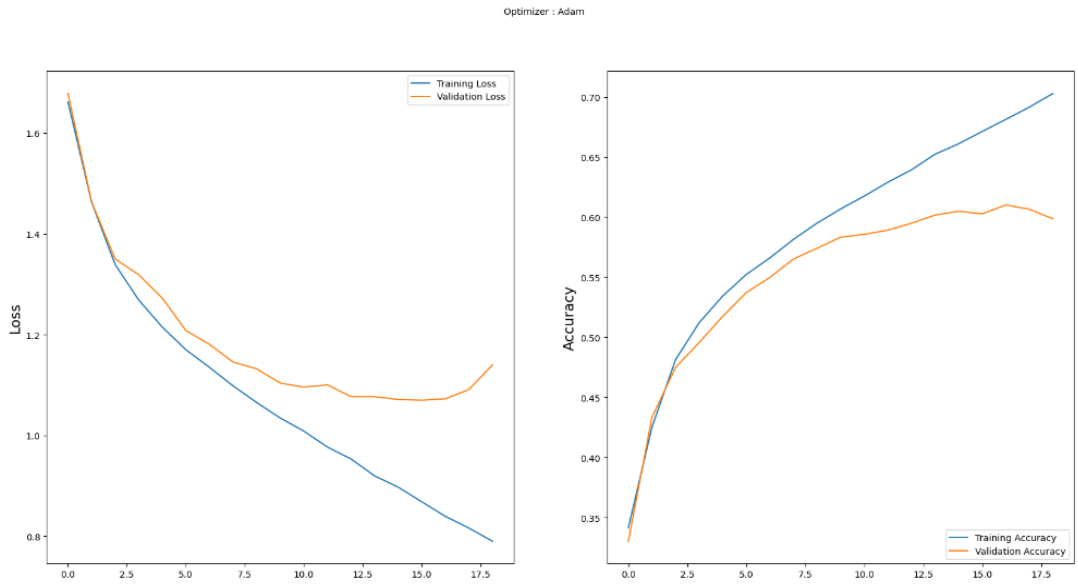

Figure 7. Loss And Accuracy Graph

In facial expression recognition using a 2D convolutional neural network, various pre-trained models developed by others can be utilized for a wide range of image-related tasks. To determine the best-performing model for emotion recognition based on facial expressions, a comparison was conducted between a self-built model, VGG16, and ResNet50. The implementation process remains similar to the self-built model, except that VGG16 and ResNet50 are imported as the initial layer from a library instead of manually defining the layers. The classification layers, also known as the FC layers, from the self-built model are employed in all cases. The architecture of two pretrained models are shown in Figure 8 and Figure 9.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 512)               14713536
_________________________________________________________________
dense_3 (Dense)              (None, 1000)              513000
_________________________________________________________________
dropout_2 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_4 (Dense)              (None, 100)               100100
_________________________________________________________________
dropout_3 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_5 (Dense)              (None, 4)                 404
=================================================================
Total params: 15,327,040
Trainable params: 15,327,040
Non-trainable params: 0
```

Figure 8. Model Architecture Of VGG16

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 2048)              23581440
_________________________________________________________________
dense_18 (Dense)             (None, 1000)              2049000
_________________________________________________________________
dropout_12 (Dropout)         (None, 1000)              0
_________________________________________________________________
dense_19 (Dense)             (None, 100)               100100
_________________________________________________________________
dropout_13 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_20 (Dense)             (None, 4)                 404
=================================================================
Total params: 25,730,944
Trainable params: 25,677,824
Non-trainable params: 53,120
```

Figure 9. Model Architecture Of ResNet50

The comparison table is shown in Table 4. The low accuracy of 34% achieved by the two pretrained models is attributed to the weight setting being set to none. This setting prevents the models from effectively learning the features present in the grayscale images of the FER-2013 dataset. While this weight setting is necessary due to VGG16 and ResNet50 only accepting color images as input, it ultimately impacts the performance of these models. Consequently, the self-built CNN model outperforms the other two models in terms of accuracy.

Table 4. Comparison Of Self-build Model And Pre-trained Models

Aspects	Self-build CNN	VGG16	ResNet50
Training Set Size	1369	4096	2048
Stopped Epoch	19	6	36
Training Loss (of final epoch)	0.7901	1.3644	1.3597
Validation Loss (of final epoch)	1.1403	1.3611	1.3619
Training Accuracy (of final epoch)	0. 7028	0.3429	0.3437
Validation Accuracy (of final epoch)	0.5986	0.3447	0.343

V. CONCLUSION

In conclusion, facial expression recognition has made significant progress in recent years, including the advancements in computer vision and machine learning techniques. However, it is important to acknowledge certain limitations that still exist in this field. One limitation is the dependence of facial expression recognition on well-lit and clear images. Poor lighting conditions, occlusions, or low-resolution images can affect the accuracy and reliability of the recognition algorithms. Future research should focus on developing robust techniques that can handle challenging imaging conditions and improve the performance of facial expression recognition in real-world scenarios. Furthermore, existing datasets for facial expression recognition often suffer from biases and limitations in terms of sample size, diversity, and representation. Future research should address these issues by collecting larger and more diverse datasets, including individuals from various age groups, ethnicities, and backgrounds, to ensure a more comprehensive understanding of facial expressions and improve the fairness and inclusivity of recognition systems.

ACKNOWLEDGEMENT

A version of this paper was presented at the third International Conference on Computer, Information Technology and Intelligent Computing, CITIC 2023, held in Malaysia on 26th-28th July 2023.

REFERENCES

[1] B. Islam, F. Mahmud, A. Hossain, “High performance facial expression recognition system using facial region segmentation, fusion of HOG and LBP features and multiclass SVM”, International Conference on Electrical and Computer Engineering (ICECE), 2018, pp. 42-45. doi: 10.1109/ICECE.2018.8636780.

[2] J. L. Joseph and S. P. Mathew, “Facial expression recognition for the blind using deep learning”, International conference on computing, power and communication technologies (GUCON), 2021, pp. 1-5. doi: 10.1109/GUCON50781.2021.9574035.

[3] S. Y. Lin, Y. W. Tseng, C. R. Wu, Y. C. Kung, Y. Z. Chen, C. M. Wu, “A continuous facial expression recognition model based on deep learning method”, International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), pp. 1-2, 2019. doi: 10.1109/ISPACS48206.2019.8986360.

[4] D. Liu, H. Zhang, P. Zhou, “Video-based facial expression recognition using graph convolutional networks”, International Conference on Pattern Recognition (ICPR), 2021, pp. 607-614. doi: 10.1109/ICPR48806.2021.9413094.

[5] S. R. Supta, M. R. Sahriar, M. G. Rashed, D. Das, R. Yasmin, “An Effective Facial Expression Recognition System”, International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE), 2020, pp. 66-69. doi: 10.1109/WIECON-ECE52138.2020.9397965.

[6] X. Xu, Z. Ruan, L. Yang, “Facial expression recognition based on graph neural network”, International Conference on Image, Vision and Computing (ICIVC), 2020, pp. 211-214. doi: 10.1109/ICIVC50857.2020.9177430.

[7] S. Ullah, A. Jan, G. M. Khan, "Facial expression recognition using machine learning techniques", International Conference on Engineering and Emerging Technologies (ICEET), 2021, pp. 1-6. doi: 10.1109/ICEET53442.2021.9659631.

[8] J. Zou, X. Cao, S. Zhang, B. Ge, "A facial expression recognition based on improved convolutional neural network", International Conference of Intelligent Applied Systems on Engineering (ICIASE), 2019, pp. 301-304. doi: 10.1109/ICIASE45644.2019.9074074.

[9] D. Wang, H. Yu, D. Wang, G. Li, "Face recognition system based on CNN", International Conference on Computer Information and Big Data Applications (CIBDA), 2020, pp. 470-473. doi: 10.1109/CIBDA50819.2020.00111.

[10] B. T. Chinimilli, T. Anjali, A. Kotturi, V. R. Kaipu, J. V. Mandapati, "Face recognition based attendance system using Haar cascade and local binary pattern histogram algorithm", International conference on trends in electronics and informatics (ICOEI)(48184), 2020, pp. 701-704. doi: 10.1109/ICOEI48184.2020.9143046.

[11] L. Fu and X. Shao, "Research and implementation of face detection, tracking and recognition based on video", International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), 2020, pp. 914-917. doi: 10.1109/ICITBS49701.2020.00202.

[12] M. G. Sarwar, A. Dey, A. Das, "Developing a LBPH-based face recognition system for visually impaired people", International Conference on Artificial Intelligence and Data Analytics (CAIDA), 2021, pp. 286-289. doi: 10.1109/CAIDA51941.2021.9425275.

[13] A. U. Naik and N. Guinde, "LBPH algorithm for frontal and side profile face recognition on GPU", International Conference on Smart Systems and Inventive Technology (ICSSIT), 2020, pp. 776-779. doi: 10.1109/ICSSIT48917.2020.9214228.

[14] R. R. Flores and M. J. Domínguez, "Real time automatic face recognition system using LBPH technique", CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), 2021, pp. 1-6. doi: 10.1109/CHILECON54041.2021.9702960.

[15] X. Wang, L. Liu, S. Yang, "An Improved Face Recognition Algorithm based on Fisherface and Machine Learning", International Conference on Machine Learning and Computer Application (ICMLCA), 2021, pp. 1-4.

[16] R. Rosnelly, M. S. Simanjuntak, A. C. Sitepu, M. Azhari, S. Kosasi, "Face recognition using eigenface algorithm on laptop camera", International Conference on Cyber and IT Service Management (CITSM), 2020, pp. 1-4. doi: 10.1109/CITSM50537.2020.9268907.

[17] P. Mohite, K. Vaibhav, P. K. Annapurani, "Thermal and Visual Face Recognition using Eigenfaces and Transfer Learning", International Conference on Applied Artificial Intelligence and Computing ICAAIC, 2022, pp. 656-662. doi: 10.1109/ICAAIC53929.2022.9792914.

[18] E. Winarno, I. H. Al Amin, H. Februariyanti, P. W. Adi, W. Hadikurniawati, M. T. Anwar, "Attendance system based on face recognition system using cnn-pca method and real-time camera", International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), pp. 301-304, 2019. doi: 10.1109/ISRITI48646.2019.9034596.

[19] Y. Lim, K. W. Ng, P. Naveen, S. C. Haw, "Emotion Recognition by Facial Expression and Voice: Review and Analysis", Journal of Informatics and Web Engineering, vol. 1, no. 2, pp. 45-54, 2022. https://doi.org/10.33093/jiwe.2022.1.2.4.

[20] M. Xin, L. W. Ang, S. Palaniappan, "A Data Augmented Method for Plant Disease Leaf Image Recognition based on Enhanced GAN Model Network", Journal of Informatics and Web Engineering, vol. 2, no. 1, pp. 1-12, 2023. https://doi.org/10.33093/jiwe.2023.2.1.1.