Chowanda *J Big Data* (2021) 8:132
https://doi.org/10.1186/s40537-021-00522-x

Journal of Big Data

RESEARCH Open Access

Separable convolutional neural networks for facial expressions recognition

Andry Chowanda*

*Correspondence:
achowanda@binus.edu
Computer Science Department, School of Computer Science, Bina Nusantara University, 11480 Jakarta, Indonesia

Abstract

Social interactions are important for us, humans, as social creatures. Emotions play an important part in social interactions. They usually express meanings along with the spoken utterances to the interlocutors. Automatic facial expressions recognition is one technique to automatically capture, recognise, and understand emotions from the interlocutor. Many techniques proposed to increase the accuracy of emotions recognition from facial cues. Architecture such as convolutional neural networks demonstrates promising results for emotions recognition. However, most of the current models of convolutional neural networks require an enormous computational power to train and process emotional recognition. This research aims to build compact networks with depthwise separable layers while also maintaining performance. Three datasets and three other similar architectures were used to be compared with the proposed architecture. The results show that the proposed architecture performed the best among the other architectures. It achieved up to 13% better accuracy and 6–71% smaller and more compact than the other architectures. The best testing accuracy achieved by the architecture was 99.4%.

Keywords: Facial expression recognition, Convolutional neural networks, Depthwise separable layers, Emotions recognition

Introduction

Emotions play a paramount role in social conversation. They convey essential non-verbal meanings along with the other social signals expressed during social interactions. Emotions can be interpreted from several channels, such as facial expressions, body gestures, and speech patterns. When conveying emotions, our facial expressions form a specific pattern depending on the emotions that we feel. In the 70s, Paul Ekman started to work on a coding system called Facial Action Coding System (FACS), describing the facial muscle actions [1]. The system enables the researcher to understand and recognise the emotions displayed through our facial expressions. When we express our emotions, some specific pattern is displayed in our face depending on the emotions. For example, when we express happiness, our lip corners are pulled back (FACS code AU12), our checks are raised (FACS code AU6), and there are some wrinkles detected around our eyes (FACS code AU1 & FACS code AU2). Those patterns can be automatically detected, learned through machine learning techniques and mapped to the appropriate emotions.

The works in automatic facial expressions recognition have been an attractive topic past these decades. Facial Expressions Recognition is paramount to build an affective system. The system can be implemented in several application such as, but not limited to: medical area (e.g., depression analysis [2], nervous system disorder [3]), entertainment area (e.g., games [4, 5]), virtual humans/agents or conversational agents [4, 6, 7] and many more. Several research efforts have focused on building an automatic facial expressions recognition system, and there remain some challenges yet to be solved. Most of the problems are shared with general problems in the computer vision fields, and they are poses, illumination, partial occlusion and variations [8].

The rise of deep learning has tremendously advanced the accuracy of facial expressions recognition tasks. Recently, various Convolutional Neural Networks (CNN) models have been implemented to solve the problems in recognising emotions from facial expressions. Generally, CNN architecture consists of convolutional, activation, and pooling layers. The convolutional layers perform the inner product of the linear filter and the inputs. The non-linear activation layers filter the important information from the inner product results in the convolutional layers process. Pooling layers are generally providing dimensional reduction. The results are generally called feature maps. A number of architectures have been proposed to solve several problems in the recognition tasks. Most of the CNN architectures perform feature maps construction by performing linear convolution processes followed by non-linear activation functions and reducing the feature maps dimension with pooling functions. Research has shown that achieving a good level of abstraction of the learning model generally requires non-linear functions of the input images [9, 10].

Moreover, generally training and classification process with CNN requires an immense computation power due to its convolution process. Therefore, it is not practical to be implemented into devices that have small or limited computational power. Moreover, a complex system such as virtual humans [4, 7] also require an architecture with an effective process. The facial expressions recognition process is a part of the virtual humans' system [4], which also requires an effective process for training and classification the emotions from facial cues. Hence, inspired by the research done by [9, 10], and [11], this research aims to propose an effective architecture of CNN by creating a separate process to deal with the depth and spatial features by also maintaining the performances level (e.g., accuracy).

To evaluate the proposed architecture, we compare it with a similar network without separable modules. In addition, we also explore the combination of the architectures with global average pooling versus the dense and flatten operation at the end of the network before the classification layers. The results have shown that the proposed architecture performs the best in both training times and accuracy. The proposed architecture has up to four times fewer trainable parameters than the other architectures, resulting in 13–46% faster training times. It also performed up to 13% better than the other architectures. The best accuracy achieved was 99.4% in the CK+ dataset. The rest of the paper is organised as follows: A recent work in CNN architecture and facial expressions recognition is described in the next section. The Proposed Architectures section illustrates the details of the models proposed in this research. Meanwhile, the details of the experiments are thoroughly explained in the Experimental Settings section. The results

of experiments are discussed in the Results and Discussions section. Finally, the Conclusion and Future Work section provides the takeaway messages from this research and the future research directions.

Recent work

Emotions recognition

Emotions convey meaning in social interactions. They generally express significant context along with spoken utterances to the interlocutors. Hence, capturing, recognising, and understanding the emotions conveyed by the interlocutors during conversation automatically are paramount to develop a system that can perceive a more holistic conversation. Affective Computing and Social Signal Processing are the areas that discuss emotions and social interaction between two agents (humans or machines). Affective Computing is a study of system development that can capture, process, recognise, interpret, and synthesise human's affects [12]. Social Signal Processing is a study in the computing domain that aims to model, analyse, and synthesise social signals between agents' interactions [13]. One of the specific studies in both domains dealing with emotions is automatic emotions recognition from humans using sensors. The idea is to have a system capable of perceiving emotions from humans and reacting based on the perceived emotions (e.g., virtual humans [4]). By understanding the humans' emotions from the social conversation, the system provides more colourful interaction to humans [4, 7, 13, 14]. Recognising emotions can be done with several features, such as: brainwave [15], heartbeat [16], voice prosody (e.g., the stress, rhythm and intonation of speech [17], facial expressions [18, 19], and body gestures [20]. The most natural features to be captured during the social conversation are voice prosody, facial expressions, and body gestures. The features can be captured by using microphones and cameras.

Datasets

Dataset is one of the important aspects of the deep learning training process. Dataset acts as the fuel to the deep learning architecture. The quality and the quantity of the data in the dataset can significantly influence the model's results trained by deep learning architecture and algorithms. Several datasets can be used to train emotions recognition from facial cues (e.g., facial expressions). The Cohn-Kanade Dataset (CK) [21], The Facial Expressions Recognition 2013 (FER2013) [22], The Maja Pantic, Michel Valstar and Ioannis Patras (MMI) [23] are the most influential dataset in the early work of facial expression recognition. CK dataset (later extended into The Extended Cohn-Kanade Dataset (CK+)) [21] has eight emotions (six basic emotions, contempt, and neutral) in 593 images from 123 subjects. FER2013 [22] and MMI [23] dataset provide seven emotions classification (six basic emotions and neutral). The FER2013 provides more than 30,000 images, and MMI provides 2900 videos collected from 25 participants. The researcher in the area of Social Signal Processing and Affective Computing recently built a multimodal database in conversation to be implemented in several areas, including facial expressions recognition. The Sustained Emotionally coloured Machinehuman Interaction using Nonverbal Expression (SEMAINE) Dataset [24] is one of the multimodal conversation database collected from the human and agents (i.e., virtual humans)

interactions. The SEMAINE Dataset has 24 interaction sessions with a total of 95 character interactions and 190 video clips [24].

Some datasets in the facial expressions recognition area also were collected with Asian respondents, for example, The Japanese Female Facial Expression (JAFFE) [25], Multimodal Asian Conversation Dataset [26], and the Indonesian Mixed emotion Dataset (IMED) [27]. The Japanese Female Facial Expression (JAFFE) [25] provides 7 classification of emotions (six basic emotions and neutral) from 213 images of 10 subjects. The Multimodal Asian Conversation Dataset [26] provides seven classifications of emotions (six basic emotions and neutral) from more than 100 minutes of videos of 5 subjects. Finally, the Indonesian Mixed emotion Dataset (IMED) [27] consists of 570 videos and 66,819 Images categorised into seven single emotions (Anger, Disgust, Fear, Happy, Sadness, Surprise and Neutral) and twelve mixed emotions [27].

Facial expression recognition with deep learning

The work in emotions recognition from facial expressions has been popular since decades ago, as the facial expression is one of the most natural and universal cues to be recognised [13, 28]. The general pipeline of the facial expressions recognition process is generally pre-processing, training, and evaluation. In the pre-processing phase, generally, face alignment, data augmentation, and face normalisation are performed [19, 28] before inputting all the images to the deep learning architecture for the training process. Several deep learning architectures can be used to train the recognition model. Convolutional Neural Network (CNN) architecture is the most popular architecture used to train the model. It provides simple and straightforward training implementations. CNN architecture also provides a relatively high accuracy score for the model. Zhu et al. [29] proposed a hybrid attention cascade network for facial expression recognition with the highest accuracy of 98.46% in the CK+ dataset. Liu et al. [30] implements CNN for facial expressions recognition with the highest accuracy of 93.70% in the CK+ dataset. There are several CNN implementation to build emotions recognition model from facial cues, there are: [31] and [19].

Some architectures provides temporal aspects (e.g., sequences of images or videos) to the model trained, for example Recurrent Neural Network (RNN) [32, 33] and Temporal CNN [34, 35]. The architectures provide more superior results dealing with temporal information (for example, the onset, apex, and offset of the facial actions units or emotions activation). Finally, some researchers also explored generative models for facial expressions recognition. Kim et al. [36] proposed deep generative-contrastive networks for facial expression recognition with the highest accuracy of 97.93% in the CK+ dataset. Cai et al. [37] also implements discriminative features for facial expression recognition with the highest accuracy of 94.39% in the CK+ dataset.

Proposed architectures

In this research, we propose a model with a depthwise separable convolutional neural network (see Figs. 1, 2). To evaluate the effectiveness of the proposed model, a similar network without separable modules was used to compare their performances (see Fig. 3). In addition, we also aimed to compare the implementation of global average pooling to the fully connected layers at the end of the network. In the global average

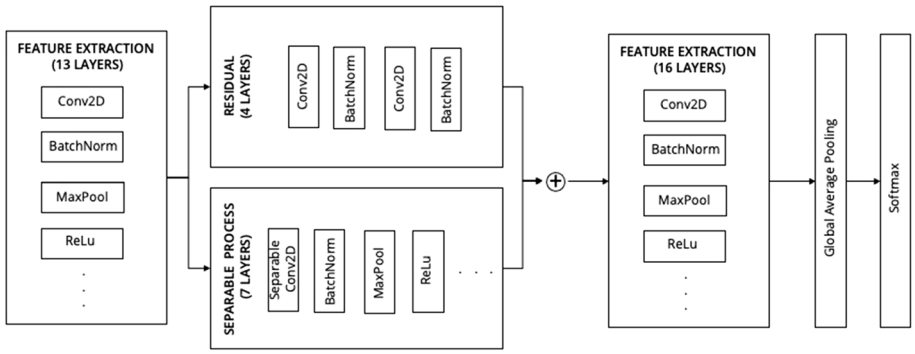

Fig. 1 Architecture with separable convolution and AVG pooling (ARCH-1)

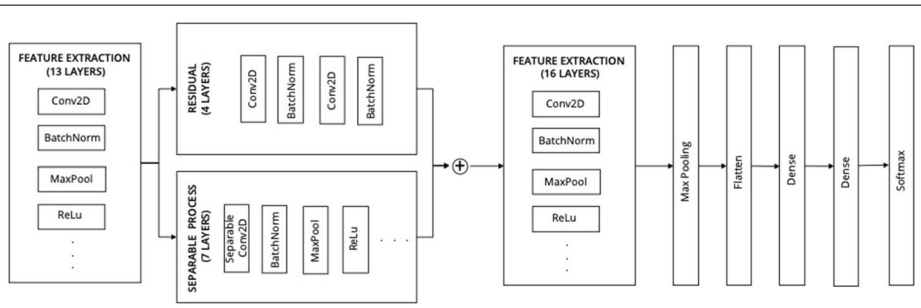

Fig. 2 Architecture with separable convolution and dense layers (ARCH-2)

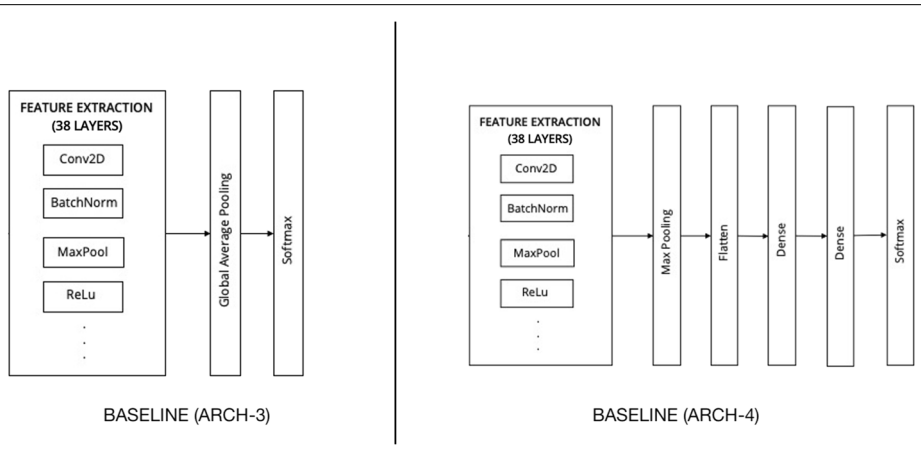

Fig. 3 Baseline architecture ARCH-3 (left) with global AVG and ARCH-4 (right) with flatten

pooling layers, the spatial average of the features maps from the previous layers are fed into the classification layer (e.g., softmax) [9]. Research has shown that global average pooling has some advantages compared to the dense and flatten operation in the classical fully connected layer in the CNN architecture. Global average pooling is robust to spatial translations of the images and reduces overfitting problems [9]. Hence, there were four models evaluated in this research. Figures 1, 2 illustrate the visualisation of the architectures proposed and evaluated in this research.

Moreover, Tables 1, 2, 3, 4 demonstrate the details of every layer of the architectures proposed and evaluated in this research. The goal of this research is to propose a lightweight architecture in CNN without sacrificing performance. The idea is to

Table 1 Architecture with separable convolution and AVG pooling (ARCH-1)

Layer (type)	Output shape	Param #	Connected to
input_1 (InputLayer)	[(None, 48, 48, 1)]	0	None
conv2d (Conv2D)		32	conv2d[0][0]
batch_normalization (BN)	(None, 48, 48, 8)	32	conv2d[0][0]
activation (Activation)	(None, 48, 48, 8)	0	batch_normalization[0][0]
conv2d_1 (Conv2D)	(None, 48, 48, 8)	576	activation[0][0]
batch_normalization_1 (BN)	(None, 48, 48, 8)	32	conv2d_1[0][0]
activation_1 (Activation)	(None, 48, 48, 8)	0	batch_normalization_1[0][0]
conv2d_2 (Conv2D)	(None, 48, 48, 16)	1152	activation_1[0][0]
batch_normalization_2 (BN)	(None, 48, 48, 16)	64	conv2d_2[0][0]
activation_2 (Activation)	(None, 48, 48, 16)	0	batch_normalization_2[0][0]
conv2d_3 (Conv2D)	(None, 48, 48, 16)	2304	activation_2[0][0]
batch_normalization_3 (BN)	(None, 48, 48, 16)	64	conv2d_3[0][0]
max_pooling2d (MaxPooling2D)	(None, 24, 24, 16)	0	batch_normalization_3[0][0]
activation_3 (Activation)	(None, 24, 24, 16)	0	max_pooling2d[0][0]
separable_conv2d (SeparableConv)	(None, 24, 24, 32)	656	activation_3[0][0]
batch_normalization_6 (BN)	None, 24, 24, 32)	128	separable_conv2d[0][0]
activation_4 (Activation)	(None, 24, 24, 32)	0	batch_normalization_6[0][0]
separable_conv2d_1 (SeparableConv)	(None, 24, 24, 32)	1312	activation_4[0][0]
conv2d_4 (Conv2D)	(None, 24, 24, 16)	256	activation_3[0][0]
batch_normalization_7 (BN)	(None, 24, 24, 32)	128	separable_conv2d_1[0][0]
batch_normalization_4 (BN)	(None, 24, 24, 16)	64	conv2d_4[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 32)	0	batch_normalization_7[0][0]
conv2d_5 (Conv2D)	(None, 12, 12, 32)	512	batch_normalization_4[0][0]
activation_5 (Activation)	(None, 12, 12, 32)	0	max_pooling2d_1[0][0]
batch_normalization_5 (BN)	(None, 12, 12, 32)	128	conv2d_5[0][0]
add (Add)	(None, 12, 12, 32)	0	activation_5[0][0] batch_normalization_5[0][0]
activation_6 (Activation)	(None, 12, 12, 32)	0	add[0][0]
conv2d_6 (Conv2D)	(None, 12, 12, 64)	18496	activation_6[0][0]
batch_normalization_8 (BN)	(None, 12, 12, 64)	256	conv2d_6[0][0]
activation_7 (Activation)	(None, 12, 12, 64)	0	batch_normalization_8[0][0]
conv2d_7 (Conv2D)	(None, 12, 12, 64)	36928	activation_7[0][0]
batch_normalization_9 (BN)	(None, 12, 12, 64)	256	conv2d_7[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 64)	0	batch_normalization_9[0][0]
activation_8 (Activation)	(None, 6, 6, 64)	0	max_pooling2d_2[0][0]
conv2d_8 (Conv2D)	(None, 6, 6, 128)	73856	activation_8[0][0]
batch_normalization_10 (BN)	(None, 6, 6, 128)	512	conv2d_8[0][0]
activation_9 (Activation)	(None, 6, 6, 128)	0	batch_normalization_10[0][0]
conv2d_9 (Conv2D)	(None, 6, 6, 128)	147584	activation_9[0][0]
batch_normalization_11(BN)	(None, 6, 6, 128)	512	conv2d_9[0][0]
activation_10 (Activation)	(None, 6, 6, 128)	0	batch_normalization_11[0][0]
conv2d_10 (Conv2D)	(None, 6, 6, 7)	8071	activation_10[0][0]
global_average_pooling2d (GlobalAvg)	(None, 7)	0	conv2d_10[0][0]
predictions (Activation)	(None, 7)	0	global_average_pooling2d[0][0]

Table 2 Architecture with separable convolution and dense layers (ARCH-2)

Layer (type)	Output shape	Param #	Connected to
input_1 (InputLayer)	[(None, 48, 48, 1)]	0	None
conv2d (Conv2D)	(None, 48, 48, 8)	72	conv2d[0][0]
batch_normalization (BN)	(None, 48, 48, 8)	32	conv2d[0][0]
activation (Activation)	(None, 48, 48, 8)	0	batch_normalization[0][0]
conv2d_1 (Conv2D)	(None, 48, 48, 8)	576	activation[0][0]
batch_normalization_1 (BN)	(None, 48, 48, 8)	32	conv2d_1[0][0]
activation_1 (Activation)	(None, 48, 48, 8)	0	batch_normalization_1[0][0]
conv2d_2 (Conv2D)	(None, 48, 48, 16)	1152	activation_1[0][0]
batch_normalization_2 (BN)	(None, 48, 48, 16)	64	conv2d_2[0][0]
activation_2 (Activation)	(None, 48, 48, 16)	0	batch_normalization_2[0][0]
conv2d_3 (Conv2D)	(None, 48, 48, 16)	2304	activation_2[0][0]
batch_normalization_3 (BN)	(None, 48, 48, 16)	64	conv2d_3[0][0]
max_pooling2d (MaxPooling2D)	(None, 24, 24, 16)	0	batch_normalization_3[0][0]
activation_3 (Activation)	(None, 24, 24, 16)	0	max_pooling2d[0][0]
separable_conv2d (SeparableConv)	(None, 24, 24, 32)	656	activation_3[0][0]
batch_normalization_6 (BN)	None, 24, 24, 32)	128	separable_conv2d[0][0]
activation_4 (Activation)	(None, 24, 24, 32)	0	batch_normalization_6[0][0]
separable_conv2d_1 (SeparableConv)	(None, 24, 24, 32)	1312	activation_4[0][0]
conv2d_4 (Conv2D)	(None, 24, 24, 16)	256	activation_3[0][0]
batch_normalization_7 (BN)	(None, 24, 24, 32)	128	separable_conv2d_1[0][0]
batch_normalization_4 (BN)	(None, 24, 24, 16)	64	conv2d_4[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 32)	0	batch_normalization_7[0][0]
conv2d_5 (Conv2D)	(None, 12, 12, 32)	512	batch_normalization_4[0][0]
activation_5 (Activation)	(None, 12, 12, 32)	0	max_pooling2d_1[0][0]
batch_normalization_5 (BN)	(None, 12, 12, 32)	128	conv2d_5[0][0]
add (Add)	(None, 12, 12, 32)	0	activation_5[0][0] batch_normalization_5[0][0]
activation_6 (Activation)	(None, 12, 12, 32)	0	add[0][0]
conv2d_6 (Conv2D)	(None, 12, 12, 64)	18496	activation_6[0][0]
batch_normalization_8 (BN)	(None, 12, 12, 64)	256	conv2d_6[0][0]
activation_7 (Activation)	(None, 12, 12, 64)	0	batch_normalization_8[0][0]
conv2d_7 (Conv2D)	(None, 12, 12, 64)	36928	activation_7[0][0]
batch_normalization_9 (BN)	(None, 12, 12, 64)	256	conv2d_7[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 64)	0	batch_normalization_9[0][0]
activation_8 (Activation)	(None, 6, 6, 64)	0	max_pooling2d_2[0][0]
conv2d_8 (Conv2D)	(None, 6, 6, 128)	73856	activation_8[0][0]
batch_normalization_10 (BN)	(None, 6, 6, 128)	512	conv2d_8[0][0]
activation_9 (Activation)	(None, 6, 6, 128)	0	batch_normalization_10[0][0]
conv2d_9 (Conv2D)	(None, 6, 6, 128)	147584	activation_9[0][0]
batch_normalization_11(BN)	(None, 6, 6, 128)	512	conv2d_9[0][0]
activation_10 (Activation)	(None, 6, 6, 128)	0	batch_normalization_11[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 3, 3, 128)	0	activation_10[0][0]
flatten (Flatten)	(None, 1152)	0	max_pooling2d_3[0][0]
dense (Dense)	(None, 512)	590336	flatten[0][0]
dense_1 (Dense)	(None, 512)	262656	dense[0][0]
prediction (Dense)	(None, 7)	3591	dense_1[0][0]

Table 3 Architecture with no-separable convolution, with AVG pooling (ARCH-3)

Layer (type)	Output shape	Param #
input_1 (InputLayer)	[(None, 48, 48, 1)]	0
conv2d (Conv2D)	(None, 48, 48, 8)	72
batch_normalization (BN)	(None, 48, 48, 8)	32
activation (Activation)	(None, 48, 48, 8)	0
conv2d_1 (Conv2D)	(None, 48, 48, 8)	576
batch_normalization_1 (BN)	(None, 48, 48, 8)	32
activation_1 (Activation)	(None, 48, 48, 8)	0
conv2d_2 (Conv2D)	(None, 48, 48, 16)	1152
batch_normalization_2 (BN)	(None, 48, 48, 8)	32
activation_2 (Activation)	(None, 48, 48, 16)	0
conv2d_3 (Conv2D)	(None, 48, 48, 16)	2304
batch_normalization_3 (BN)	(None, 48, 48, 8)	32
activation_3 (Activation)	(None, 48, 48, 16)	0
conv2d_4 (Conv2D)	(None, 48, 48, 16)	2304
batch_normalization_4 (BN)	(None, 48, 48, 8)	32
max_pooling2d (MaxPooling2D)	(None, 24, 24, 16)	0
activation_4 (Activation)	(None, 24, 24, 16)	0
conv2d_5 (Conv2D)	(None, 24, 24, 32)	4608
batch_normalization_5 (BN)	(None, 48, 48, 8)	32
activation_5 (Activation)	(None, 24, 24, 32)	0
conv2d_6 (Conv2D)	(None, 24, 24, 32)	9216
batch_normalization_6 (BN)	(None, 48, 48, 8)	32
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 32)	0
activation_6 (Activation)	(None, 12, 12, 32)	0
conv2d_7 (Conv2D)	(None, 12, 12, 64)	18432
batch_normalization_7 (BN)	(None, 48, 48, 8)	32
activation_7 (Activation)	(None, 12, 12, 64)	0
conv2d_8 (Conv2D)	(None, 12, 12, 64)	36864
batch_normalization_8 (BN)	(None, 48, 48, 8)	32
max_pooling2d_2 (MaxPooling2D)	(None, 12, 12, 32)	0
activation_8 (Activation)	(None, 6, 6, 64)	0
conv2d_9 (Conv2D)	(None, 6, 6, 128)	73728
batch_normalization_9 (BN)	(None, 48, 48, 8)	32
activation_9 (Activation)	(None, 6, 6, 128)	0
conv2d_10 (Conv2D)	(None, 6, 6, 128)	147456
batch_normalization_10 (BN)	(None, 48, 48, 8)	32
activation_10 (Activation)	(None, 6, 6, 128)	0
conv2d_11 (Conv2D)	(None, 6, 6, 7)	8071
global_average_pooling2d (Global AVG)	(None, 7)	0
predictions (Activation)	(None, 7)	0

propose an architecture with a separable convolution process. The architecture will separate the spatial cross-correlations from the channel cross-correlations to learn richer and smaller features [10]. The architecture processes the depth (i.e., channel) and spatial (i.e., width and height) features of the input (i.e., images) separately. The depthwise separable convolution process has two processes [10]. The first process is called depthwise convolution, where the spatial features are extracted and handled

Table 4 Architecture with-out separable convolution, with dense layers (ARCH-4)

Layer (type)	Output shape	Param #
input_1 (InputLayer)	[(None, 48, 48, 1)]	0
conv2d (Conv2D)	(None, 48, 48, 8)	72
batch_normalization (BN)	(None, 48, 48, 8)	32
activation (Activation)	(None, 48, 48, 8)	0
conv2d_1 (Conv2D)	(None, 48, 48, 8)	576
batch_normalization_1 (BN)	(None, 48, 48, 8)	32
activation_1 (Activation)	(None, 48, 48, 8)	0
conv2d_2 (Conv2D)	(None, 48, 48, 16)	1152
batch_normalization_2 (BN)	(None, 48, 48, 8)	32
activation_2 (Activation)	(None, 48, 48, 16)	0
conv2d_3 (Conv2D)	(None, 48, 48, 16)	2304
batch_normalization_3 (BN)	(None, 48, 48, 8)	32
activation_3 (Activation)	(None, 48, 48, 16)	0
conv2d_4 (Conv2D)	(None, 48, 48, 16)	2304
batch_normalization_4 (BN)	(None, 48, 48, 8)	32
max_pooling2d (MaxPooling2D)	(None, 24, 24, 16)	0
activation_4 (Activation)	(None, 24, 24, 16)	0
conv2d_5 (Conv2D)	(None, 24, 24, 32)	4608
batch_normalization_5 (BN)	(None, 48, 48, 8)	32
activation_5 (Activation)	(None, 24, 24, 32)	0
conv2d_6 (Conv2D)	(None, 24, 24, 32)	9216
batch_normalization_6 (BN)	(None, 48, 48, 8)	32
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 32)	0
activation_6 (Activation)	(None, 12, 12, 32)	0
conv2d_7 (Conv2D)	(None, 12, 12, 64)	18432
batch_normalization_7 (BN)	(None, 48, 48, 8)	32
activation_7 (Activation)	(None, 12, 12, 64)	0
conv2d_8 (Conv2D)	(None, 12, 12, 64)	36864
batch_normalization_8 (BN)	(None, 48, 48, 8)	32
max_pooling2d_2 (MaxPooling2D)	(None, 12, 12, 32)	0
activation_8 (Activation)	(None, 6, 6, 64)	0
conv2d_9 (Conv2D)	(None, 6, 6, 128)	73728
batch_normalization_9 (BN)	(None, 48, 48, 8)	32
activation_9 (Activation)	(None, 6, 6, 128)	0
conv2d_10 (Conv2D)	(None, 6, 6, 128)	147456
batch_normalization_10 (BN)	(None, 48, 48, 8)	32
activation_10 (Activation)	(None, 6, 6, 128)	0
max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 32)	0
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 512)	590336
dense_1 (Dense)	(None, 512)	262656
dense_1 (Dense)	(None, 512)	262656

within this process. The second process is called the pointwise convolution process, where the depth features (e.g., RGB channels) are extracted and handled within this process. In the depthwise convolution process, D number of $X \times X \times D$ kernels were applied in the convolutional process towards $M \times N$ inputs with D dimensions (e.g.,

RGB channels). The output will be $Y \times Y \times D$ features maps. While in the pointwise convolution process, P number of $1 \times 1 \times D$ kernels were applied to the convolutional process towards the inputs (i.e., $Y \times Y \times D$ feature maps). The output will result in $Y \times Y \times P$ feature maps. For examples see Fig. 1 and Table 1. The kernel used in this research was $X = 3 \times X = 3 \times D = 32$ towards $M = 24 \times N = 24$ inputs with $D = 1$ (black and white) or $D = 3$ (RGB) and the output will be $Y = 24 \times Y = 24 \times D = 32$ feature maps (with same padding setting), in the first separable process (Table 1 see separable_conv2d and Fig. 1 see lower block of separable process). Moreover, in the another separable process (the residual, Fig. 1 see upper block of residual), the kernel used was $1 \times 1 \times D = 16$ towards the inputs of $Y = 24 \times Y = 24$ inputs with $D = 1$ (black and white) or $D = 3$ (RGB) and the output will be $Y = 24 \times Y = 24 \times D = 16$ feature maps (with same padding setting, Table 1 see conv2d_4).

In addition, the global average pooling process also enormously reduce the number of parameters while maintaining the spatial translations in the images. The proposed architecture (ARCH-1) has 292,862 trainable parameters (a total of 293,951 parameters), four times smaller than the other similar architectures. Figure 1 illustrates the proposed architecture with separable convolution layers and a global average pooling process. The first feature extraction process has 13 layers of alternating convolutional, batch normalisation, max pooling, and activation (i.e., ReLu) layers. The ReLu function is described as $Relu(x) = max(0, x)$. The next process is divided into two separable processes. The first separable process has seven layers of alternating separable convolution, batch normalisation, max pooling, and activation (i.e., ReLu) layers. The second separable process is four residual layers with alternating convolutional and batch normalisation layers. The first separable process has 3×3 kernel, while the second process (i.e., the residual layers) has 1×1 kernel. The next layers consist of 16 layers of alternating convolutional, batch normalisation, max pooling, and activation (i.e., ReLu) layers. Finally, the classification layers have a global average pooling and an activation layer (i.e., Softmax) . In the global average pooling, an input with $M \times N \times D$ tensor is reduced to a $1 \times 1 \times D$ tensor by taking the average of all $M \times N$ values [9, 38]. The dimension of the activation layer depends on the k number of classes (i.e., $1 \times k$, with $k \in \{0, \ldots, 6\}$ in this case). Table 1 shows the details of each layer in the architectures with the input of $48 \times 48 \times 1$ tensors. To compare the performances, the proposed model was compared with a similar architecture from ARCH-1 architecture. The architecture (ARCH-2) has flatten and dense layers in the classification layers instead of the global average pooling layer. Figure 2 demonstrates the architecture of ARCH-2. The networks are similar with ARCH-1, with Max Pooling, flatten, two dense, and activation (i.e., Softmax) layers as the classification layers. The architecture has 1,142,463 parameters with 1,141,375 trainable parameters, almost four times larger than the proposed architecture (ARCH-1). Table 2 shows the details of each layers in the architectures with the input of $48 \times 48 \times 1$ tensors. Moreover, two more architectures, with no separable convolution layers implemented, were also explored in this paper as a comparison. One architecture (ARCH-3) using a global average pooling and activation (i.e., Softmax) layers as the classification layers, while the other (ARCH-4) use max pooling, flatten, two dense, and activation (i.e., Softmax see Eq. 1) layers as the classification layers. Table 3 illustrates the details of the ARCH-3 layers with 305,807 trainable parameters from a total of 306,831 parameters. Table 4 illustrates the details of

the ARCH-4 layers with 1,154,319 trainable parameters from a total of 1,155,343 parameters with the input of $48 \times 48 \times 1$ tensors.

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{1}$$

Experimental settings

Datasets

Three facial expressions datasets were used to evaluate the proposed architectures, they are: The Extended Cohn-Kanade Dataset (CK+) [21], The Facial Expressions Recognition 2013 (FER-2013) Dataset [22], and Indonesian Mixed emotion Dataset (IMED) [27]. The CK+ dataset consists of almost 600 FACS-coded sequences with seven classifications of emotions: Angry, Disgust, Fear, Happy, Sadness, Surprise, and Contempt [21]. In this research, the proposed architectures were only used to classify the seven emotions, and the AU coding and classification was not used. The second dataset used in this research was the FER-2013, which consists of 35,685 facial expressions images [22]. The dataset is categorised into seven emotions: Happiness, Neutral, Sadness, Anger, Surprise, Disgust, Fear. The third dataset used in this research was the Indonesian Mixed emotion Dataset (IMED), where it consists of 570 videos and 66,819 Images categorised into seven single emotions (Anger, Disgust, Fear, Happy, Sadness, Surprise and Neutral) and twelve mixed emotions [27]. In this research, the proposed architectures were only used to classify the seven single emotions ($k \in \{0, \ldots, 6\}$). The datasets were augmented to enrich the data for training and validation sets. The augmented process is thoroughly explained in the next sub-section.

Pre-processing and initial hyper-parameters settings

Several pre-processing processes were applied to the dataset before being trained with the proposed architectures. First, face detection and localisation were applied to find and crop the face from the images. In the next step, face alignment was applied to the cropped images, and finally, a normalisation was also applied to all the images. To enrich the data, a data augmentation technique was also implemented for all the datasets. The images were rotated with 20 rotation ranges, shifted, zoomed, and flipped horizontally. The datasets then were split into 86-88% for training and 12–14% of validation (test) sets. A total of 81,954 augmented images were used as training (72,520) and validation (9434) sets. Specifically, the augmented FER2013 dataset has 57,418 training images and 7178 validation images. The augmented CK+ dataset has 1308 training images and 186 validation images. Finally, the augmented Indonesian Mixed emotion Dataset (IMED) has 13,794 training images and 2070 validation images. As the initial settings, the learning rate was set to 0.01 and reduced by a factor of 10 every time the model loss encountered a plateau during the learning process. The datasets were trained in a maximum of 200 epochs and 256 of batch size (10 for CK+ dataset) with an early stopping method if there was no significant improvement in the loss of the model. To avoid overfitting, an L2 regularisation of 0.2 and dropout of 0.5 were applied to the models. All the proposed architectures implement Adam as the training optimiser (see Eq. 2). The (θ_{t+1} is the update of the weights at time $t + 1$. The weights were optimised from the previous

weights (θ_t), learning rate α, the Exponential Moving Average (EMA) of the gradient $\nabla f(x_t)$ ($\hat{m}_t$), Exponential Moving Average (EMA) of the gradient $\nabla f(x_t)$ ($\hat{v}_t$). Finally, to prevents the weights are being divided by zero, a regularisation (ϵ) is used.

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2}$$

To create uniform inputs of the images between those three datasets, we resize all the images to $48 \times 48 \times 1$ dimension. Four architectures were explored in this research, resulting in 12 sets of results (three datasets for each architecture). Table 5 illustrates the settings differences between architectures. ARCH-1 is the proposed architecture, and the others were used as the comparison for the performance evaluation. Both ARCH-1 and ARCH-2 have separable layers, where ARCH-1 implements global average pooling in the classification layers and ARCH-2 implements flatten layers. Both ARCH-3 and ARCH-4 have no separable layers, where ARCH-3 implements global average pooling in the classification layers and ARCH-4 implements flatten layers. The initial inputs for all architectures are N number of images with $48 \times 48 \times 1$ tensors, and the outputs are N number of images with 1×7 tensors.

Results and discussions

The proposed architecture (ARCH-1) was evaluated with three datasets (see "Experimental settings"). In addition, the proposed architecture was also compared with three other architectures with the same datasets. Table 5 demonstrates the overview of the settings of the architectures that were explored in this research. Tables 1, 2, 3, 4 illustrate the details of the networks' layers. The model training process took more than 20 h of total training time with Titan V GPU for all combination architectures and datasets, with an average training time of 1.72 h. The longest training time was 3.91 h, with ARCH-4 as the architecture and IMED as the dataset. The fastest training time was 0.217 h (13.02 min), with ARCH-1 as the architecture and CK+ as the dataset. Overall, ARCH-1 provides the fastest training time compared to the other architectures, with the average training time for all datasets was 0.841 h (50.44 minutes). Meanwhile, the longest training time was achieved by ARCH-4 with an average training time of 2.633 h. The results demonstrate that overall the ARCH-1 provides 70% faster training time compared to the other architectures. Another alternative in fast training time is ARCH-3, with an average of 1.013 h training times for all datasets. ARCH-1 has 292,862 trainable parameters, while ARCH-3 has 305,807 trainable parameters. Although the training parameters are extremely compressed, the proposed architecture is capable of learning the important features from the data resulting in the best accuracy compared to the other architectures. Figure 4 illustrates the visualisation of the feature maps from the convolutional layer(conv2d_4) of ARCH-1 architecture.

Table 5 Architectures setting

	Global Avg	**Flatten**
Separable	ARCH-1	ARCH-2
No separable	ARCH-3	ARCH-4

Fig. 4 Conv2D activation filters visualisation of ARCH-1

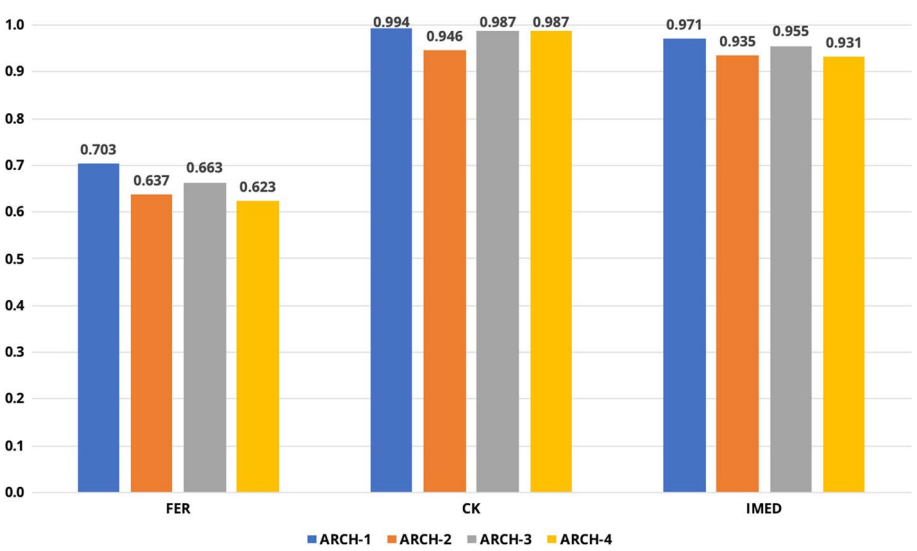

Fig. 5 Overall results

Figure 5 demonstrate the overview results from all combinations of the architectures and the datasets. The results demonstrate ARCH-1 architecture excel in all datasets compared to the other architectures. From the datasets perspective, CK+ provides the best results across all the architectures. Overall, ARCH-1 architecture provides the best training accuracy scores across all the datasets with an average training accuracy score of 88.9%, followed by ARCH-3 architecture that gives an average training accuracy score of 86.8%. ARCH-4 architecture takes third place with the average training accuracy score of 84.7%, while ARCH-2 architecture shows the lowest training accuracy scores across all the datasets with an average training accuracy score of 83.9%. The best testing accuracy was achieved by ARCH-1 trained with CK+ dataset with the training accuracy score of 99.4%. The lowest score achieved by ARCH-4 architecture was trained with the FER2013 dataset with the training accuracy score of 62.3%. Architectures trained

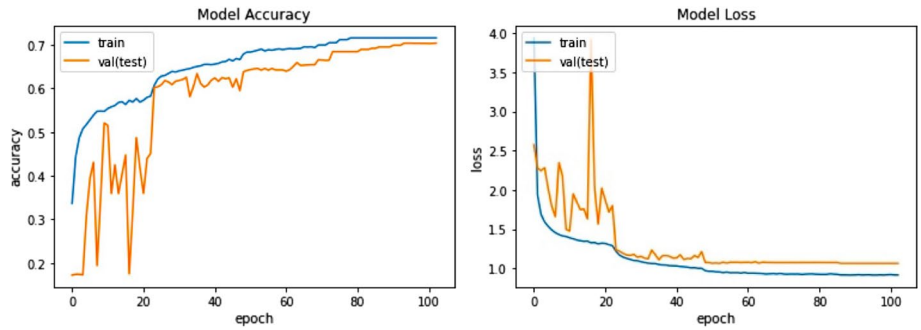

Fig. 6 Training and testing accuracy and loss of ARCH-1 with FER2013 dataset

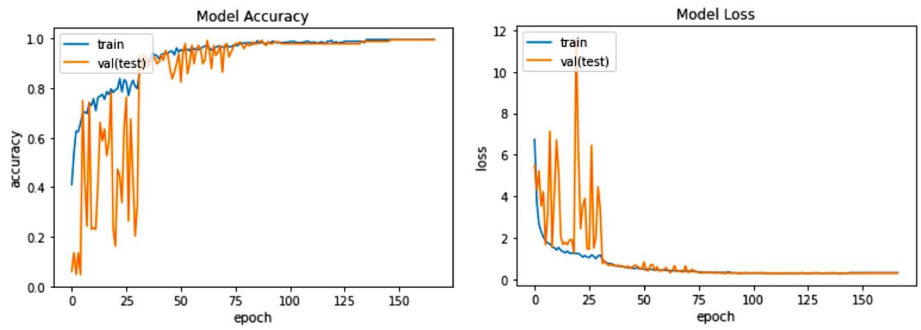

Fig. 7 Training and testing accuracy and loss of ARCH-1 with CK dataset

with CK+ and IMED datasets provide excellent results (> 90%), while the architectures trained with the FER2013 dataset provide the lowest score among the other datasets (61.3 − −70.3). The proposed architecture, ARCH-1, also achieved higher results compared to the existing architecture proposed in the literature. ARCH-1 also achieved a higher training accuracy score compared to one of the best results with CNN architecture from the literature, where Ding et al. [39] achieved 98.6% of training accuracy score and Zhang et al. [40] achieved 98.9% of training accuracy score. Moreover, the proposed architecture also achieved a higher training accuracy score compare with the literature that existed, where Liliana et al. [27] achieved 84.52% of training accuracy score with Support Vector Machine (SVM) as the classifier algorithm.

Figures 6, 7, 8 demonstrate the best model in each dataset trained in the best architecture (ARCH-1). Figure 6 illustrates the model accuracy and loss (from Adam's optimiser) in the FER dataset training and testing phase using ARCH-1 (the best architecture). The training stopped at 102 epochs when the model could not learn more information from the dataset. The model achieved the best training accuracy of 70.5%, testing accuracy of 70.3%, training loss (loss during training phase—towards training set) of 0.91%, and testing loss (loss during validation/testing phase—towards validation/test set) of 1.13%. Figure 7 shows the model accuracy and loss accuracy and loss in CK+ dataset training and testing phase using the best architecture (ARCH-1). The model achieved the best training accuracy of 99.4%, testing accuracy of 99.4%, training loss of 0.13%, and testing loss

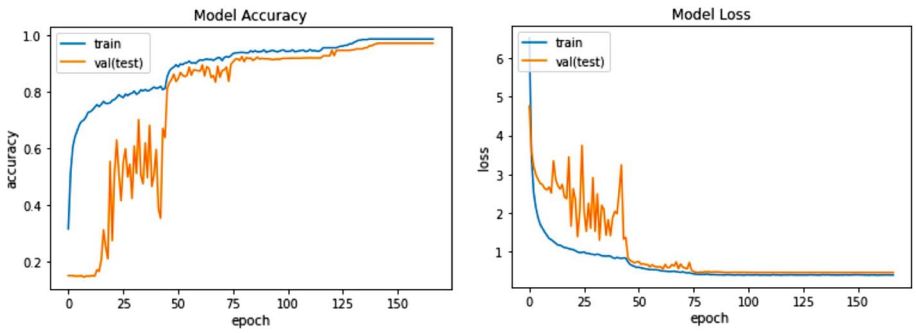

Fig. 8 Training and testing accuracy and loss of ARCH-1 with IMED dataset

of 0.13%. The training stopped at 162 epochs, where the training and testing accuracy and loss started to converge at 93 epochs. Finally, Figure 8 demonstrates the model accuracy and loss accuracy and loss in the IMED dataset training and testing phase using the best architecture (ARCH-1). The model achieved the best training accuracy of 97.1%, testing accuracy of 96.4%, training loss of 0.35%, and testing loss of 0.39%. The training stopped at 164 epochs when the model could not learn significant information from the dataset.

Conclusion and future work

This research proposed a separable convolutional neural network with global average pooling to enhance real-time emotion classification experiences while also improving performance. The proposed methods were evaluated with three datasets and compared with three other architectures. The proposed architecture (ARCH-1) is empirically performing better in terms of training speed and accuracy. The results have shown that the number of trainable parameters was tremendously reduced compare to the other similar architectures. The training times were also reduced to 6–71% compared to similar architectures. In addition, the proposed architecture achieved up to 13% better accuracy compared to the other architectures, with the same dataset and settings. In general, ARCH-2 performed the worst results compared to all architectures explored in this research. The best accuracy achieved belonged to ARCH-1 with CK+ dataset (99.4%), while ARCH-4 achieved the worst accuracy with the FER2013 dataset (62.3%).

Moreover, the architecture with dense and flatten layers resulted in lower accuracy and a slower training process. The limitation of this research is that the architectures were only implemented to train a specific task (i.e., facial expressions) and were only used to train black and white (1-D Channel). The proposed architecture can be used in the other classification tasks with a similar performance level if using the same settings described in this paper. In the future, the proposed architecture will also be trained and evaluated with a more significant number of images and the larger number of image dimensions. The combination of the datasets seems interesting to be explored in the future. The datasets have similar emotions labels and might provide more variation to the models as IMED consists of Asian (i.e., Indonesian) faces, while FER and CK+ consist of the majority of Caucasian faces. The models created in this research will also be implemented to a device with limited computing power (e.g., raspberry, robot) and to

a complex system (e.g., virtual humans [4, 7]). Finally, the temporal aspect or features can also be considered as the future direction for improving the proposed architecture. Different onset, peak, and offset activation times of expressions resulted in a different semantics meaning of emotions (e.g., acted and spontaneous expressions).

Abbreviations
FACS: Facial action coding system; AU: Action unit; CNN: Convolutional neural networks; FER: Facial expression recognition; SVM: Support vector machine.

Acknowledgements
We would like to extend our gratitude to NVIDIA Corporation with the donation of the Titan V Pascal GPU used for this research.

Authors' contributions
AC is the only author in this paper. The author read and approved the final manuscript.

Authors' information
Andry is a Computer Science lecturer in Bina Nusantara University Indonesia. Andry received his Ph.D. degree in Computer Science from The University of Nottingham UK, a master degree in Business Management from BINUS Businesss School ID, and a bachelor degree in Computer Science from BINUS University ID. His research is in agent architecture and Machine (and Deep) Learning. His work mainly on how to model an agent that has capability to sense and perceive the environment and react based on the perceived data in addition to the ability of building a social relationship with the user overtime. In addition, Andry is also interested in serious game and gamification design.

Funding
There is no funding source.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 8 June 2021 Accepted: 2 October 2021
Published online: 16 October 2021

References

1. Ekman R. What the face reveals: basic and applied studies of spontaneous expression using the facial action coding system (FACS). USA: Oxford University Press; 1997.
2. Song S, Jaiswal S, Shen L, Valstar M. Spectral representation of behaviour primitives for depression analysis. IEEE Transactions on Affective Computing. 2020.
3. Ricciardi L, Visco-Comandini F, Erro R, Morgante F, Bologna M, Fasano A, Ricciardi D, Edwards MJ, Kilner J. Facial emotion recognition and expression in Parkinsons disease: an emotional mirror mechanism? PLoS ONE. 2017;12(1):0169110.
4. Chowanda A, Blanchfield P, Flintham M, Valstar M. Erisa: Building emotionally realistic social game-agents companions. In: International Conference on Intelligent Virtual Agents, pp. 134–143 (2014). Springer.
5. Akbar MT, Ilmi MN, Rumayar IV, Moniaga J, Chen T-K, Chowanda A. Enhancing game experience with facial expression recognition as dynamic balancing. Proc Comput Sci. 2019;157:388–95.
6. Mascarenhas S, Guimarães M, Santos PA, Dias J, Prada R, Paiva A. Fatima toolkit–toward an effective and accessible tool for the development of intelligent virtual agents and social robots. arXiv preprint arXiv:2103.03020 (2021).
7. Sutoyo R, Chowanda A, Kurniati A, Wongso R. Designing an emotionally realistic chatbot framework to enhance its believability with aiml and information states. Proc Comput Sci. 2019;157:621–8.
8. Szeliski R. Computer vision: algorithms and applications. USA: Springer; 2010.
9. Lin M, Chen Q, Yan S. Network in network. arXiv preprint arXiv:1312.4400 (2013).
10. Chollet F. Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1251–1258 (2017).
11. Szegedy C, Ioffe S, Vanhoucke V, Alemi A. Inception-v4, inception-resnet and the impact of residual connections on learning. arXiv preprint arXiv:1602.07261 (2016).
12. Picard RW. Affective computing. USA: MIT press; 2000.

13. Vinciarelli A, Pantic M, Bourlard H. Social signal processing: survey of an emerging domain. Image Vis Comput. 2009;27(12):1743–59.
14. Zhu W, Chowanda A, Valstar M. Topic switch models for dialogue management in virtual humans. In: International Conference on Intelligent Virtual Agents, pp. 407–411 (2016). Springer.
15. Alarcao SM, Fonseca MJ. Emotions recognition using EEG signals: a survey. IEEE Trans Affect Comput. 2017;10(3):374–93.
16. Valenza G, Citi L, Lanatá A, Scilingo EP, Barbieri R. Revealing real-time emotional responses: a personalized assessment based on heartbeat dynamics. Sci Rep. 2014;4(1):1–13.
17. Zhao Z, Li Q, Zhang Z, Cummins N, Wang H, Tao J, Schuller BW. Combining a parallel 2d cnn with a self-attention dilated residual network for ctc-based discrete speech emotion recognition. Neural Netw. 2021;141:52–60.
18. Valstar M, Zafeiriou S, Pantic M. 11 facial actions as social signals. Social signal processing. 2017;123.
19. Chowanda A, Sutoyo R. Convolutional neural network for face recognition in mobile phones. ICIC Express Lett. 2019;13(7):569–74.
20. Piana S, Stagliano A, Odone F, Verri A, Camurri A. Real-time automatic emotion recognition from body gestures. arXiv preprint arXiv:1402.5047 (2014).
21. Lucey P, Cohn JF, Kanade T, Saragih J, Ambadar Z, Matthews I. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-workshops, pp. 94–101 (2010). IEEE.
22. Goodfellow IJ, Erhan D, Carrier PL, Courville A, Mirza M, Hamner B, Cukierski W, Tang Y, Thaler D, Lee D-H. Challenges in representation learning: a report on three machine learning contests. In: International Conference on Neural Information Processing, pp. 117–124 (2013). Springer
23. Pantic M, Valstar M, Rademaker R, Maat L. Web-based database for facial expression analysis. In: 2005 IEEE International Conference on Multimedia and Expo, p. 5 (2005). IEEE.
24. McKeown G, Valstar M, Cowie R, Pantic M, Schroder M. The semaine database: annotated multimodal records of emotionally colored conversations between a person and a limited agent. IEEE Trans Affect Comput. 2012;3(1):5–17. https://doi.org/10.1109/T-AFFC.2011.20.
25. Lyons M, Akamatsu S, Kamachi M, Gyoba J. Coding facial expressions with gabor wavelets. In: Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition, pp. 200–205 (1998). IEEE.
26. Suryani D, Ekaputra V, Chowanda A. Multi-modal Asian conversation mobile video dataset for recognition task. Int J Electr Comput Eng (IJECE). 2018;8(5):4042–6.
27. Liliana DY, Basaruddin T, Oriza IID. The indonesian mixed emotion dataset (imed) a facial expression dataset for mixed emotion recognition. In: Proceedings of the 2018 International Conference on Artificial Intelligence and Virtual Reality, pp. 56–60 (2018).
28. Li S, Deng W. Deep facial expression recognition: a survey. IEEE Transactions on Affective Computing. 2020.
29. Zhu X, Ye S, Zhao L, Dai Z. Hybrid attention cascade network for facial expression recognition. Sensors. 2021;21(6):2003.
30. Liu M, Li S, Shan S, Chen X. Au-inspired deep networks for facial expression feature learning. Neurocomputing. 2015;159:126–36.
31. Pham L, Vu TH, Tran TA. Facial expression recognition using residual masking network. In: 2020 25th International Conference on Pattern Recognition (ICPR), pp. 4513–4519 (2021). IEEE.
32. Daihong J, Lei D, Jin P, et al. Facial expression recognition based on attention mechanism. Scientific Programming. 2021;2021.
33. Liang X, Xu L, Liu J, Liu Z, Cheng G, Xu J, Liu L. Patch attention layer of embedding handcrafted features in cnn for facial expression recognition. Sensors. 2021;21(3):833.
34. Reddy SPT, Karri ST, Dubey SR, Mukherjee S. Spontaneous facial micro-expression recognition using 3d spatiotemporal convolutional neural networks. In: 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2019). IEEE.
35. Pan X. Fusing hog and convolutional neural network spatial-temporal features for video-based facial expression recognition. IET Image Proc. 2020;14(1):176–82.
36. Kim Y, Yoo B, Kwak Y, Choi C, Kim J. Deep generative-contrastive networks for facial expression recognition. 2019; 1703:07140.
37. Cai J, Meng Z, Khan AS, Li Z, O'Reilly J, Tong Y. Island loss for learning discriminative features in facial expression recognition. In: 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), pp. 302–309 (2018). IEEE.
38. Zhou B, Khosla A, Lapedriza A, Oliva A, Torralba A. Learning deep features for discriminative localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2921–2929 (2016).
39. Ding H, Zhou SK, Chellappa R. Facenet2expnet: Regularizing a deep face recognition net for expression recognition. In: 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), pp. 118–126 (2017). IEEE.
40. Zhang Z, Luo P, Loy CC, Tang X. From facial expression recognition to interpersonal relation prediction. Int J Comput Vis. 2018;126(5):550–69.

Publisher's Note