Extract the pain features from facial expressions by utilizing two DCNN model

Elaf N.Saddam [1], Saad Mutashar [1] and Wissam H. Ali[1]

[1]Department of Electrical Engineering, University of Technology-Baghdad, Iraq
eee.19.07@grad.uotechnology.edu.iq
Saad.M.Abbas@uotechnology.edu.iq
30088@uotechnology.edu.iq

***Abstract*—adopting facial expressions in automatic pain recognition is a complicated matter that has intrigued the interest of the scientific community. This paper provides a performance comparison between different Deep Convolution Neural Network (DCNN) architectures with different deepnesses (number of layers and connections): ResNet-50 and DenseNet-121. These models were using as feature extractors by removing the fully connected layer and freezing its convolution layers. Then the features vector is fed to four types of classification algorithms: SVM (Support Vector Machine), KNN (k Nearest Neighbor), LR (Logistic Regression), and building a new DNN (Deep Neural Network) classifier from scratch. The data set is collected from 110 subjects adults and teenagers (71 males and 39 females) based on natural environment and different times. The best classification results is 96.15% by utilizing the DenseNet-121 model in mirage it with different classifiers. The most important things that must be perceived from this result are the depth of the network and the proper choice of transfer learning technique.**

***Index Terms*— Pain recognition, facial expression, deep learning and visual features.**

I. Introduction

Facial expressions provide an indicator and spontaneous responses to experiences of pain. Therefore, the human face is considered as a high content treasure of nonverbal information data related to the health state [1]. Facial Action Coding System (FACS), which divided the face into 44 subset, called Action Units (AUs) elaborate in the construction of different facial expressions; it is the oldest and most widely used tool in facial expression coding and has been used in the majority of previous studies on facial expression.

Recent research has primarily focused on using the models of machine learning [2] or deep learning [3] to recognize pain-related facial expressions. The studies revealed the effectiveness of traditional machine learning and deep learning models for recognition tasks, particularly when trained on large fully labelled datasets and tested under controlled environmental conditions. However, these models' robustness, accuracy, and complexity continue to be an issue in relation to the images of pain themselves, which are generally unbalanced and not available in sufficient quantity to develop and create a robust and authentic method based on deep learning algorithms, which require large datasets.

This paper provides an alternative method for automatic pain recognition based on facial expressions using pre-trained convolution neural network (CNN) architectures, such as ResNet-50 [4] and DenseNet-121 [5]. The selection of these architectures has been inspired by their excellent performance in various recognition problems, as demonstrated by the ImageNet Large Scale Visual Recognition Challenge [6]. These structures have been trained with over a million photos in order to characterize photos into 1000 different labeled classes. These networks' structures are used in the mode of feature extraction, where the convolution layers of the networks exploit them to extract features and then feed these features to classification algorithms, such as SVM (Support Vector Machine) [7], KNN (k Nearest Neighbor) [8], LR (Logistic Regression) [9] and building new Deep Neural Network (DNN) [10] classifiers. The experiments have been done on a collected data set in the natural environment, containing over 6000 images. The experiments revealed exciting results about the structure of CNN's utility for automatic pain recognition from facial expressions.

The essential contributions of this study are:

- Find the best performance between the models of CNNs (ResNet-50 and DenseNet-121) based on their extracted features, which are affected by the model structure.
- Provide experiment results on a collected dataset in the natural environment, containing over 6000 images.
- Evaluate different types of classifiers (SVM, KNN, and LR) with the CNN architecture to find the best grouping (features + classifier).
- Create a DNN classifier from scratch and evaluate it with a natural-condition environment dataset.
- Aims to make all of the materials and code used in this study accessible to the research community (via GitHub) in order to promote the principles of unbiased comparison and reproducible investigation.

The works in the paper ordered as in section 2 presents some of related work regards to pain recognition. Section 3 discusses the various CNN architectures, as well as the methodological

approach. Section 4 describes the experiments that were carried out and the results that were obtained. Section 5 conclusions with some recommendations and possible directions.

II. Related Work

Recently, an automatic pain recognition based on facial expressions has received a lot of attention. The important mission is to determine whether or not there is any pain present a (Binary Classification). In these works generally, a Facial Action Coding System (FACS) is used, which divides the face into 44 subsets called Action Units (AUs) that are used to create various facial expressions. This depending is accomplished by using the FACS to code the strength of specific action units, which is then determined for every single video frame. Several approaches to pain recognition as a binary classification problem have been proposed, with the goal of distinguishing between pain and no pain expressions, here take a look at some of the existed works like what's Lucey et al [11] SVMs were used to address Action Units then pain identification. They followed two ways for pain detection, straight by using features of image or approach of two-step, where Action Units are detecting firstly then by using Logistical Linear Regression (LLR) fused the output for detect the pain. Chen et al [12] presented framework for detect pain in videos. Histogram of Oriented Gradients (HOG) used to identify the facial pain expression as frame level features. Then as classifier, Support Vector Machine (SVM) trained to do this mission. Rodriguez et al [13] exploit VGG-Face, which is pre-trained network for extract features from face. Then Theses features fused to a Long Short-Term Memory (LSTM). Ilyas et al [14] this work exploit the approach of hybrid deep learning, through grouping Convolution Neural Network(CNN) and Recurrent Neural Network (RNN) which allowed the use of spatio-temporal information of the collected data to detect the patient's state from facial expressions. Bargshady et al [15] they have been using two various RNN that had been pre-trained with VGGFace-CNN and afterwards merged together as a pain assessment network. Bargshady et al [16] presented their work by exploit the pre-trained CGG-Face model to extract feature. The method employs a hybrid deep structure that combines two-stream CNNs associated with long short-term memory CNN-BiLSTM. To distinguish pain from the face, a (VGG-Face) which is pre-trained CNN and LSTM algorithm were used. Huang et al [17] proposed to use multidimensional features were extracted by using 3D CNN for spatiotemporal features , 2D CNN for spatial features and 1D CNN for geometric information then all these features united for regression to estimate pain. Pooja et el [18] propose system by using VGG-16 model which is pre-trained on ImageNet dataset. Then by using transfer learning adjust the model for pain recognition task by using softmax layer. Othman et el [19] propose system consist of two branch to classify pain cases, the first one called random forest baseline model, starts with detect face from patient video and then by Facial Activity Descriptor (FAD) extract facial AUs. Then passed the features passed on the random forest classifier and the predicted result is a starting point for the next branch, which consist of two simple convolution neural network. Despite the fact, that many previous studies on pain state have used deep learning depending on transfer learning and adjustments of VGG-Face but the accuracy still not satisfactory. This study, investigated two various CNN architectures to use for pain recognition based on facial expression

III. Proposed Method

Lately, transfer learning has witnessed extended use for feature extraction, particularly in computer vision. It does this by boosting the use of previous knowledge gained from previous tasks. In this study, the investigated models are also extensively reliant on transfer learning [20]. The proposed approach exploits popular CNN architectures such as ResNet-50 and DenseNet-121. These networks are used in feature extraction mode. This means that the features from different convolution layers are extracted and used as inputs to SVM (Support Vector Machine), kNN (k Nearest Neighbor), LR (Logistic Regression), and the construction of a new DNN (Deep Neural Network). Popular CNN architectures such as ResNet-50 and DenseNet-121. These networks are used in feature extraction mode. This means that the features from different convolution layers are extracted and used as inputs to SVM (Support Vector Machine), KNN (K Nearest Neighbor), LR (Logistic Regression), and the construction of a new DNN (Deep Neural Network) classifier. The most discriminating features of the face are automatically extracted from the training data, which has been collected from patients in an unideal environment, conditions for much more reliable results to be close to reality. Better visual interpretation does not always exist in these characteristics, but they symbolize features, which are essential to differentiating pain. The two models were initially trained on the ImageNet dataset, which contained over 1,000 classes. This study elaborates on modified versions of these models for our pain recognition task by freezing the convolutional layers and removing the fully connected layer from each network. The proposed methodology is depicted in Figure 1. Both of the CNN architectures that were used throughout the experiments are briefly described below.

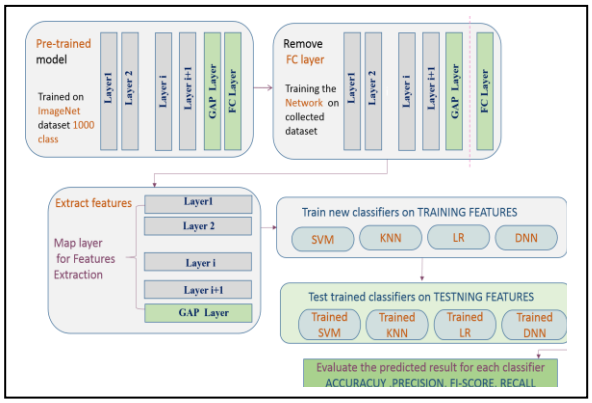

Figure 1: Framework of proposed system.

ResNet-50: Residual Networks is a Microsoft team model consisting of 50 layers. The essential parts are residual blocks and the identity connection, which takes the input directly to the end of each block [4]. In a residual block, the information from

DOI: http://doi.org/10.24086/cocos2022/paper.686

each layer feeds into the next layer, and connections to the layers are about 2–3 hops away. It achieved on ImageNet, directly via the identity database, a top-1 error rate of 20.47% and a top-5 error rate of 5.25%. Each residual block consists of three convolution layers (1x1, 3x3, and 1x1 respectively). Their structure is explained in Table 1.

DenseNet-121: Dense Convolutional Network, which focuses in its design on enabling deep learning networks to go deeper while also making them more efficient to train by leveraging shorter connections between layers [21]. DenseNet design where each layer is connected to all the other deeper layers. This direct access enables it to retrieve information and then overcome the gradient problem. DenseNet consists, in addition to the basic convolutional, pooling, and classification layers, of two important blocks: the Dense Blocks and the Transition Layers. In the structure of the DenseNet-121 model, there are four dense blocks and three transition layers. Its structure is explained in Table2.

Table 1: Structure of ResNet-50

Name of layer	Output size	Resnet-50
Convolution	112 × 112	7 × 7, 64, stride2
Conve 2	56 × 56	3 × 3 max pool, stride 2 [1x1,64 . 3x3,64 . 1x1,256]x3
Conve 3	28 × 28	[1x1,128 . 3x3,128 . 1x1,512]x4
Conve 4	14 × 14	[1x1,256 . 3x3,256 . 1x1,1024]x6
Conve 5	7 × 7	[1x1,512 . 3x3,512 . 1x1,2048]x3
Average pool	1 × 1	global average pooling
Classification (Fully connected & softmax)	1000fc	2048 x1000 fc

Table 2: Structure of DenseNet-121

Name of layer	Output size	Densenet-121
Conve 1	112 × 112	7 × 7, 64, stride2
Poo ling	56 × 56	3 × 3 max pool, stride 2
Dense block1	56 × 56	[1x1,conv . 3x3,conv]x6
Transition layer	56×56 28x28	1x1,conv 2x2 average pooling stride 2
Dense block 2	28x28	[1x1,conv . 3x3,conv]x12
Transition layer	28×28 14x14	1x1,conv 2x2 avarege pooling stride 2
Dense block 3	14x14	[1x1,conv . 3x3,conv]x24
Transition layer	14×14 7x7	1x1,conv 2x2 avarege pooling stride 2
Dense block 4	7 × 7	[1x1,conv . 3x3,conv]x16
Average pool	1 × 1	global average pooling
Classification (Fully connected & softmax)	1000fc	102400 fc

IV. Experimental Analysis

This section aims to explain the experimental steps in this study starting with data collection ending with the most important results.

A. Dataset

For this research study, the dataset has been collected by the iPhone-7 plus camera from 110 subjects (adults and teenagers) (71 males and 39 females). Most of them are undergoing physiotherapy sessions, and the second phase was collected from volunteers and different painful cases. The records were taken in a natural environment during four periods: A) Before the painful procedure to get the normal face B) Try to expose non-painful simulation to capture their expression. C) Painful simulation exposure In addition, in D) after 2 minutes of painful stimulation. So get 304 videos (154 with pain expressions and 150 without pain expressions). Then, by using the viola-jones algorithm, more than 6000 images of the face region were cropped; the larger number was for pain expression. To prevent the model from being biased for pain cases. Then, they removed similar frames from each video and collected just the key frames. Then 3174 face images were saved in jpg format and 24 bit as shown in Figure 2. After the face frame has been gotten, the preprocessing starts as a simple step that just involves resizing the face images by 244x244 pixels to fit with the input of the ResNet-50 model and DenseNet-121 model.

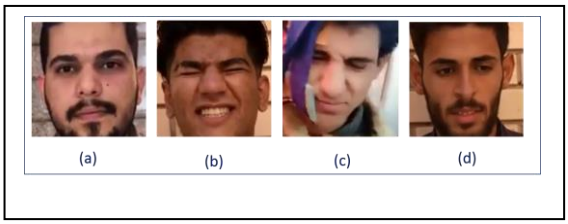

Figure. 2 Expression samples in a collection of images. (a) Explain no pain expression, While (b) and (c) pain expressions (d) rare pain expressions (almost no pain).

B. Experimental Setup

The experimental process starts with splitting the dataset by the ratio of 50:50 into two equal parts, called the training set and the testing set. Each set consists of 1587 images. By using Keras, training dataset images were augmented randomly through left-right flips, brightness, and saturation. To increase the data and overcome if there is any lack of enough samples to get better recognition and to avoid the overfitting problem that is popular in the Deep CNN model. However, there is no augmentation process on the testing dataset. Then the sets are fed into the pre-trained CNN models (ResNet-50 and

DOI: http://doi.org/10.24086/cocos2022/paper.686

DenseNet-121), which exploit by freezing their convolutional layers for extracting features at the end of the extracting process, getting a feature vector. These features are passed on to classification algorithms to distinguish between pain cases and no pain. The algorithms that are used as classifiers in this paper are SVM, KNN, and LR. Moreover, to complete the process of deep neural network building, a neural network classifier from scratch by using a (he-normal) initializer for dense layers with a seed of 42. Which is the best result technique with the Leaky-Relu activation function. The model has one input layer of 1024 inputs and 256 outputs. The process of each layer is called forward propagation, which is computed as (1). Then 256 will be an input of the dense layer (hidden layer) and the Xavier initializer (glorot with seed = 42) to initialize the weight parameter [22], which gives the best results with a sigmoid activation function. Then after reading the data, start the data pipeline with data shuffle = 1024 and batch size = 32. Global average pooling and binary cross-entropy loss function [23] as shown in (2) to compute the error (J (W)), and Adam optimizer [24] were used to update the weight by a backpropagation algorithm [25], as shown in (3). The epoch number is set to 25.

$$O^ij = b^ij + \sum_{i=1}^{N^{l-1}} conv(Wij^{l-1}Xi^{l-1}) \quad (1)$$

Where:

$O^ij =$ input

$Wij^{l-1} =$ kernel from ith neural at layer $i-1$

$Xi^{l-1} =$ output of ith neural at layer $i-1$

$b^ij =$ bias of jth neural at layer l

$$BCE\ (J\ (W)) = -\frac{1}{m}\sum_{i=1}^{m} Ylog(Ypred) + (1-Y)log(1-Ypred) \quad (2)$$

Where:

m = number of observation.

$Ypred =$ Prediction value

Y = Actual value

$$\frac{\partial}{\partial w^iij} J(W) = \Delta w^iij$$

$$w^iij(new) = w^iij(old) - \eta \Delta w^iij \quad (3)$$

To evaluate the results for each proposed system which consist of features extractor model and classification algorithm, four type of matric have been used for this mission which computing the accuracy of the model as shown in (4) when the higher accuracy, mean the lower error rate. Precision as shown in (5) which mean how many of predictions were correct, Recall as shown in (6) which means how many right hits were finding and F1-Score which evaluating performance of the model via the harmonic of precision and recall as shown in (7)

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (5)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (6)$$

$$\text{F1} - \text{Score} = 2 \times \frac{Precision \times Recall}{Precision + Recal} \quad (7)$$

A. Results and Discussion

All experiments were done on a personal laptop using the Jupiter Notebooks Keras framework [26] and the Tensor flow library with Python 3.7. Processor Intel inside Core i5, GPU 4 GHz, and memory (RAM) 8.00 GB. The results of the evaluation of proposed methods based on the pre-trained CNN models ResNet-50 and DenseNet-121 are presented in Table 3 and Table 4, respectively. These results illustrate that the models that exploit DenseNet-121 get the best performance. This is an intriguing observation. It shows that when the DenseNet-121 architecture was at its deepest, it was deeper than the ResNet-50. Therefore, the important thing to pay attention to is the deeper architecture that increases the performance of the model. In addition, as you can see from Figure 3, the loss rate decreases to less than 0.15 in the DenseNet-121 model because each layer has a direct connection to the output, in addition to the connection with other layers. This makes the model more stable than ResNet-50 models and then keeps more information. Therefore, when observing the results, it should pay attention to the most important thing: reusing the models as feature extractors gives higher accuracy than the previous research shown in Table 5, which depends on fine-tuning models.

Table 3. Evaluation results of proposed method based on ResNet-50 model.

Proposed method	ResNet50 +SVM	ResNet-50+KNN	ResNet-50+LR	ResNet50 +DNN
TP	640	750	620	790
TN	630	700	620	100
FP	150	41	170	690
FN	160	96	180	7
Accuracy (%)	80.15	91.36	77.98	56.8
Precision (%)	81	94.4496	78.48	59.17
Recall (%)	80	88.65	77.55	99.11
F1-score (%)	80.49	91.05	77.98	69.22

Table 4. Evaluation results of proposed method based on DenseNet-50 model.

Proposed method	DenseNet-121+SVM	DenseNet-121+KNN	DenseNet-121+LR	DenseNet-121+DNN
TP	760	760	770	760
TN	770	740	770	760
FP	34	31	27	32
FN	20	49	22	29
Accuracy (%)	96.59	94.95	96.91	96.15
Precision (%)	95.78	96	96.61	95.97
Recall (%)	97.47	93.82	97.22	96.34
F1-score (%)	96.5	94.99	96.89	96.16

DOI: http://doi.org/10.24086/cocos2022/paper.686

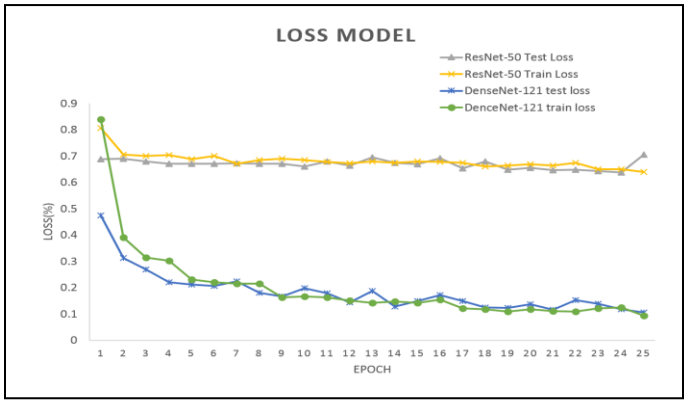

Figure 3. Comparison the train and test loss of ResNet-50 and DenseNet-121 models cross 25 epoch

Table 5. Explains methods and performances of the cited state-of-the-art researches

Reference	Approach (extracted feature with classification method)	Performance
Lucey et al 2011 [11]	Extracted Features: Shape, CAPPS, APP with SVM	83.9% AUC
Rodriguez et al [13]	Extracted Features VGG-16 with LSTM	83.% Accuracy
Chen et al 2017 [12]	Extracted Features: HOG-TOP, HOG with SVM	91 %Accuracy
Ilyas et al 2018 [14]	Extracted Features: spatio-temporal with LSTM	70% Accuracy
Bargshady et al 2019 [15]	Extracted Features: VGG-face with RNN	75% Accuracy
Bargshady et al 2020 [16]	Extracted Feature: VGG-face with EJH-CVV-BiLSTM	85% Accuracy
Huang et al 2021 [17]	Extracted Features: geometric information, spatial features and spatiotemporal with Hybrid network	0.76 MSE
Pooja et el 2021 [18]	Extracted Features: VGG-16	93%Accuracy
Othman et el 2021 [19]	Extracted Features: FAD with 2 CNN and RFc	51%Accuracy
The proposed model	Extracted Features:,DenseNet-121 with DNN	96.15% Accuracy

IV. Conclusions

Living in a speedily world like the one we are living in necessitates the use of technology, which allows us to simplify complicated tasks, organize our lives, and helps to improve the use of our time and efforts. The developments in technology helped in this paper by exploitation of the models, which were previously built to find a high-accuracy method for automatic pain recognition system. Taking into account the correct selection of the depth of the structure as well as the appropriate technique from the transfer learning technology for the task to be performed, as observed, using the DenseNet-121 model as a feature extractor, and with its large depth structure, enables one to get higher accuracy of 96%. Therefore, with this promising result, this work could be supported with future enhancement and encouraged to plan for working on another deep convolution neural network model like the Xception model by including head movement with facial expression and evaluating the score of pain, because with a high accuracy rate, an automatic pain recognition system is widely approximated to be applicable in a clinical environment.

References

[1] E. N. Saddam, S. Mutashar, and W. H. Ali, "A Study of Patient's Pain Assessment Based on Facial Expression: Issues and Challenges," Eng. Technol. J., vol. 39, no. 10, pp. 1514–1527, 2021.

[2] X.-D. Zhang, "Machine learning," in A Matrix Algebra Approach to Artificial Intelligence, Springer, 2020, pp. 223–440.

[3] G. Zamzmi, D. Goldgof, R. Kasturi, and Y. Sun, "Neonatal pain expression recognition using transfer learning," arXiv Prepr. arXiv1807.01631, 2018.

[4] R. U. Khan, X. Zhang, R. Kumar, and E. O. Aboagye, "Evaluating the performance of resnet model based on image recognition," in Proceedings of the 2018 International Conference on Computing and Artificial Intelligence, 2018, pp. 86–90.

[5] T. Zhang, R. Wang, J. Ding, X. Li, and B. Li, "Face recognition based on densely connected convolutional networks," in 2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM), 2018, pp. 1–6.

[6] O. Russakovsky et al., "Imagenet large scale visual recognition challenge," Int. J. Comput. Vis., vol. 115, no. 3, pp. 211–252, 2015.

[7] S. Brahnam, C.-F. Chuang, F. Y. Shih, and M. R. Slack, "SVM classification of neonatal facial images of pain," in International Workshop on Fuzzy Logic and Applications, 2005, pp. 121–128.

[8] Z. Deng, X. Zhu, D. Cheng, M. Zong, and S. Zhang, "Efficient kNN classification algorithm for big data," Neurocomputing, vol. 195, pp. 143–148, 2016.

[9] C. Zhou, L. Wang, Q. Zhang, and X. Wei, "Face recognition based on PCA and logistic regression analysis," Optik (Stuttg)., vol. 125, no. 20, pp. 5916–5919, 2014.

[10] B. B. Traore, B. Kamsu-Foguem, and F. Tangara, "Deep convolution neural network for image recognition," Ecol. Inform., vol. 48, pp. 257–268, 2018.

DOI: http://doi.org/10.24086/cocos2022/paper.686

[11] P. Lucey, J. F. Cohn, K. M. Prkachin, P. E. Solomon, and I. Matthews, "Painful data: The UNBC-McMaster shoulder pain expression archive database," in 2011 IEEE International Conference on Automatic Face & Gesture Recognition (FG), 2011, pp. 57–64.
[12] J. Chen, Z. Chi, and H. Fu, "A new framework with multiple tasks for detecting and locating pain events in video," Comput. Vis. Image Underst., vol. 155, pp. 113–123, 2017.
[13] P. Rodriguez et al., "Deep pain: Exploiting long short-term memory networks for facial expression classification," IEEE Trans. Cybern., 2017.
[14] C. M. A. Ilyas, M. A. Haque, M. Rehm, K. Nasrollahi, and T. B. Moeslund, "Facial expression recognition for traumatic brain injured patients," in International Conference on Computer Vision Theory and Applications, 2018, pp. 522–530.
[15] G. Bargshady, X. Zhou, R. C. Deo, J. Soar, F. Whittaker, and H. Wang, "Enhanced deep learning algorithm development to detect pain intensity from facial expression images," Expert Syst. Appl., vol. 149, p. 113305, 2020.
[16] G. Bargshady, J. Soar, X. Zhou, R. C. Deo, F. Whittaker, and H. Wang, "A joint deep neural network model for pain recognition from face," in 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS), 2019, pp. 52–56.
[17] Y. Huang, L. Qing, S. Xu, L. Wang, and Y. Peng, "HybNet: a hybrid network structure for pain intensity estimation," Vis. Comput., pp. 1–12, 2021.
[18] P. Prajod, D. Schiller, T. Huber, and E. André, "Do Deep Neural Networks Forget Facial Action Units?--Exploring the Effects of Transfer Learning in Health Related Facial Expression Recognition," arXiv Prepr. arXiv2104.07389, 2021.
[19] E. Othman, P. Werner, F. Saxen, A. Al-Hamadi, S. Gruss, and S. Walter, "Automatic vs. Human Recognition of Pain Intensity from Facial Expression on the X-ITE Pain Database," Sensors, vol. 21, no. 9, p. 3273, 2021.
[20] P. Marcelino, "Transfer learning from pre-trained models," Towar. Data Sci., 2018.
[21] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.
[22] S. K. Kumar, "On weight initialization in deep neural networks," arXiv Prepr. arXiv1704.08863, 2017.
[23] U. Ruby and V. Yendapalli, "Binary cross entropy with deep learning technique for image classification," Int. J. Adv. Trends Comput. Sci. Eng., vol. 9, no. 10, 2020.
[24] D. Yi, J. Ahn, and S. Ji, "An effective optimization method for machine learning based on ADAM," Appl. Sci., vol. 10, no. 3, p. 1073, 2020.
[25] R. Kishore and T. Kaur, "Backpropagation algorithm: an artificial neural network approach for pattern recognition," Int. J. Sci. Eng. Res., vol. 3, no. 6, pp. 6–9, 2012.
[26] N. Ketkar, "Introduction to keras," in Deep learning with Python, Springer, 2017, pp. 97–111.

DOI: http://doi.org/10.24086/cocos2022/paper.686