

Keywords: pain assessment, artificial intelligence, deep learning, facial recognition, transfer learning

Abeer A. Mohamad ALSHIHA ¹

¹ University of Mosul, Iraq, abeer.allaf@uomosul.edu.iq

* Corresponding author: abeer.allaf@uomosul.edu.iq

Quantifying pain: An AI-driven approach to detecting pain levels via facial expressions

Abstract

The automatic recognition of pain intensity from facial expressions has great potential to enhance the objective, non-invasive assessment of pain. In this study, we explore supervised deep learning methods for classifying facial pain intensity across 16 levels. Three CNN architectures (EfficientNetB0, MobileNetV2, Small CNN) were tested to compare pretrained and task-specific approaches to fine-grained pain classification. After suitable preprocessing and data augmentation of face images, the models were trained and tested on facial images. The following metrics were used to evaluate the performance: accuracy, balanced accuracy, precision, recall, F1-score, specificity, Cohen's kappa, Matthews's correlation coefficient and log loss. EfficientNetB0 had the highest accuracy (98.00%), balanced accuracy (97.61%), precision (97.88%), recall (97.61%) and F1 score (97.74%). The MobileNetV2 achieved 95.00% accuracy and a 94.61% F1 score, while the Small CNN achieved 88.00% accuracy and an 87.37% F1 score. The efficientNetB0 also had the smallest log loss with 0.0683. The results show that deep learning is a viable approach to fine-grained pain-intensity classification in facial images, and that state-of-the-art architectures can discriminate among different pain levels. External validation and greater subject diversity and temporal facial information should be investigated further to evaluate robustness in practical and clinical settings.

1. INTRODUCTION

Despite the volume of research into this phenomenon, pain, as a sensory perception and a subjective experience of the individual's psyche, still raises many unanswered questions about its nature. It can originate from physiological states, trauma and diseases of absentees, whether it is AIDS, Parkinson's disease, bone fractures, spinal pains, and chronic headaches, among others. Pain evaluation is considered one of the most important aspects of medical practice, as it helps doctors make decisions and increases patients' satisfaction with the services provided (Bargshady, 2020).

For assessing pain intensity in primary and secondary care medical clinics, the patient's perception is first documented using a structured pain assessment protocol that includes the following tools: VAS, NRS, and VRS (Saddam, Mutashar, & Ali, 2021). However, such approaches of self-reporting suffer from some drawbacks, especially when the patient cannot speak for their increased or decreased levels of pain because of a neurological disorder or dementia or after a surgical procedure has been done on them (Leo, Carcagni, Mazzeo, Spagnolo, Cazzato, & Distante, 2020).

In such cases, behavioural measures, including the Neonatal Infant Pain Scale (NIPS), Pain Assessment in Advanced Dementia (PAINAD), and the Behavioural Pain Scale (BPS), are used, in which patient movements, vocalisation, facial expressions, and other non-verbal signs are documented and analysed. In addition, it is common to get feedback from the caregivers and family members to assess the patient's pain experience (Raja et al., 2020).

Nevertheless, some of the challenges likely to be encountered when using these methods include physician fatigue and misinterpretation of pain signals. Criticising the lack of accuracy and standardisation in pain assessment methods, scientists have begun to incorporate AI solutions into their focus. Computer and AI systems can analyze more specific details of the face and its expressions, such as micro-expressions, movements and variations, to detect and measure the level of pain in a much better way (Janssen, 2021; Alshiha et al., 2023).

Therefore, to address the above gaps, this study proposes designing an automated multi-level pain assessment system using AI to capture pain levels beyond the six levels, from levels seven to ten. To expand the realm of machine learning pain analysis in the given field, this work aims to improve the accuracy of facial pain detection through using a number of advanced approaches of machine learning, including transfer learning to enhance the capabilities of extracting pain features from the face, training of neural networks on a richer set of degrees of pain, and calculating the overall effect of the dataset balance. The opportunities that AI offers in the assessment of pain cannot be disregarded, as it provides accurate analyses that can be burdensome for healthcare facilities, helps enhance patient care, and improves the quality of pain management across various capacities (Al-Neama et al., 2023).

2. RELATED WORKS

According to Acevedo et al. (2017), a very basic and straightforward facial expression recognition technique based on the geometric relations among facial landmark points could be designed using geometric features. This technique extracts features from the face and represents them using geometric descriptors based on the distances, sizes, and angles of triangles. The modified K-Nearest Neighbours (KNN) classifier has been employed using these features to detect facial expressions. The proposed technique has been tested with the CK+ database of facial expressions. It has proved to be a more reliable facial expression recogniser than some existing techniques. However, the proposed technique does not rely on computationally expensive deep learning methods but rather on the accurate detection of geometric features.

Likewise, Shier (2017) offers a computer-aided method for detecting pain from facial expressions. The method combines the capabilities of Convolutional Neural Networks (CNNs), Gabor-based feature extraction, and SVM classification to analyse facial features associated with pain. The method employs the Pain Severity and Intensity (PSPI) scale as a benchmark for evaluating pain-related facial expressions. The research demonstrates the viability of computer vision and machine learning tools for the automatic detection of pain-related facial expressions, particularly in healthcare settings such as aged-care facilities. This paper suggests that facial analysis may serve as a non-invasive tool for evaluating pain when verbal communication is not possible. Limitations of this approach include its reliance on facial expressions as a primary source of information and difficulties arising from variations in individual characteristics, recording conditions, and patterns of pain expression.

On these bases, Virrey et al. (2019) propose a study on automated facial expression recognition, focusing on evaluating various facial expression datasets. The methodology will include analysing facial expression features and evaluating their applicability to automatic facial recognition systems using various Facial Expression Recognition (FER) databases. The study reveals the significance of facial datasets and features for recognition effectiveness and achieves an accuracy of about 85.66% using current FER databases. This indicates that automated facial analysis can perform recognition at a promising level and also contribute to practical emotion- and pain-related facial analysis. Nevertheless, the study is constrained by the nature and diversity of current FER databases, which may not adequately capture real-life facial features.

In their paper, (Nour, Elhebir, and Viriri 2020) present a deep learning solution to the problem of facial expression recognition, based on several popular Convolutional Neural Network architectures. The methods explore the capabilities of AlexNet, VGG-16, and ResNet networks for automatic facial expression recognition using the Extended Cohn-Kanade (CK+) dataset. The purpose of evaluating different architectures was to test their potential to learn discriminative facial features and classify facial expressions. According to the findings, AlexNet achieved the highest accuracy, 88.2%, demonstrating its suitability for the problem under consideration. The presented research demonstrates the feasibility of using deep CNNs to solve the problem of facial expression recognition and shows the potential of learned image representations for facial expression classification. However, the research is primarily focused on the CK+ dataset, which could hinder generalisation to real-world scenarios.

In their research, Andersen et al. (2021) suggest machine-based approaches for detecting pain-related facial expressions, focusing on the automatic detection of facial Action Units (AUs) and temporal dynamics in pain expressions. The approaches include facial movement coding and machine-learning algorithms that detect specific facial movements in images. In addition, recurrent neural networks are tested as methods for learning temporal information from annotated video sequences. The importance of the temporal nature of facial pain expressions and its contribution to pain assessment is stressed. The results show that the combination of facial

movement analysis and machine learning can be applied for automatic pain recognition. Nonetheless, such an approach requires accurate facial coding and representative video datasets. Besides, individual variability and the complex temporal dynamics of facial movements may complicate automatic pain recognition.

A study by El Morabit et al. suggests research into deep convolutional neural networks for automatic recognition of pain from facial expressions. The methods involve comparing several CNN architectures, such as MobileNet, GoogleNet, ResNeXt-50, ResNet-18, and DenseNet-161. This research is conducted both on the ability to predict pain intensity directly and on using deep CNNs as feature extractors in combination with traditional machine learning algorithms such as SVR and RFR. The experiments are conducted on the UNBC-McMaster Shoulder Pain Archive data, demonstrating the potential of deep CNNs to recognise patterns in pain-related facial expressions. The paper shows that deep representations learned through neural networks could help in automatic pain estimation. But the research mainly uses a single pain dataset. Differences in subjects' properties, facial appearance, and pain expression could become an obstacle to its generalisation. In addition, the computational complexity of some CNN architectures could prevent their use in practice.

Also, Saddam, Mutashar, and Ali (2021) suggest a structured method for developing an automatic pain detection system based on the analysis of facial gestures and other signals. The techniques cover various steps of the pain recognition system development process, such as data collection, pre-processing, feature extraction, classifier training, and validation. The importance of a multimodal approach, in which facial gestures and vocal signals can be used together in order to identify pain in medical settings, is emphasised. According to the research results, depending on the situation and the person, a single modality may not be enough because pain can present in different ways. Therefore, the authors' approach enables the use of several behavioural indicators to more accurately assess pain. However, in the case of a multimodal pain detection system, it is necessary to have synchronised and large enough data sets with different types of signals.

According to (Agha & AZIZ 2023), there exists a deep learning method for automatic face pain detection via facial expressions. This methodology employs the UNBC pain database to divide facial expressions into two classes: painful and non-painful. Two forms of the database were used: an unbalanced and a balanced. The FaceNet approach was applied to extract facial features, and the features were then classified using a fully connected neural network. The findings have been characterised by an accuracy rate of 99.665% for the unbalanced database and 95.44% for the balanced one. But there exist some weaknesses within the study. They include the use of a single database, class imbalance, which introduces bias, and the use of two pain categories.

3. FACIAL ACTION CODING SYSTEM (FACS)

The Facial Action Coding System (FACS) is a comprehensive, anatomically based method designed to measure all observable facial movements. It achieves this by categorising every visually discernible facial activity into 45 distinct Action Units (AUs) and considering various head and eye postures and movements. Each AU is associated with a numerical code, the assignment of which is somewhat arbitrary but serves as a standardised reference for facial muscle movements (Baumeister & Vohs, 2012).

Action Units (AUs) represent precise facial muscle movements that convey a wide range of emotions and expressions. FACS, developed by Paul Ekman and Wallace Friesen, identifies 45 distinct AUs, each corresponding to specific movements or combinations of movements of the facial muscles. Table 1 describes these action units.

Tab. 1. Face action unit

Action Unit	Description	Facial Muscle
1	Raising Inner Brow	Frontalis, pars medialis
2	Raising Outer Brow (unilateral, right side)	Frontalis, pars lateralis (right side only)
4	Lowering Brow	Depressor Glabellae, Depressor Supercilli, Currugator
5	Raising Upper Lid	Levator palpebrae superioris
6	Raising Cheek	Orbicularis oculi, pars orbitalis
7	Tightening Lid	Orbicularis oculi, pars palpebralis
9 (also shows slight AU4 and AU10)	Wrinkling Nose	Levator labii superioris alaquae nasi
10 (also shows slight AU25)	Raising Upper Lip	Levator Labii Superioris, Caput infraorbitalis
11	Deepening Nasolabial	Zygomatic Minor
12	Pulling Lip Corners	Zygomatic Major
13	Puffing Cheeks	Levator anguli oris (Caninus)
14	Creating Dimples	Buccinator
15	Depressing Lip Corners	Depressor anguli oris (Triangularis)
16 (with AU25)	Depressing Lower Lip	Depressor labii inferioris
17	Raising Chin	Mentalis
18 (with slight AU22 and AU25)	Puckering Lips	Incisivii labii superioris and Incisivii labii inferioris
20	Stretching Lips	Risorius
22 (with AU25)	Funneling Lips	Orbicularis oris
23	Tightening Lips	Orbicularis oris
24	Pressing Lips	Orbicularis oris
25	Parting Lips	Depressor Labii, Relaxation of Mentalis, Orbicularis Oris
Action Unit	Description	Facial Muscle
26 (with AU25)	Dropping Jaw	Masetter; Temporal and Internal Pterygoid relaxed
27	Stretching Mouth	Pterygoids, Digastric
28 (with AU26)	Sucking Lips	Orbicularis oris
41	Drooping Lid	Relaxation of the Levator Palpebrae Superioris
42	Narrowing Eyes	Orbicularis oculi
43	Closing Eyes	Relaxation of the Levator Palpebrae Superioris
44	Squinting Eyes	Orbicularis oculi, pars palpebralis
45	Blinking Eyes	Relaxation of Levator Palpebrae and Contraction of Orbicularis Oculi, Pars Palpebralis
46	Winking Eye	Levator palpebrae superioris; Orbicularis oculi, pars palpebralis

Each Action Unit (AU) within the Facial Action Coding System (FACS) can interplay with others, resulting in a vast spectrum of facial expressions, each intricately tied to a specific emotion. For instance, the amalgamation of AUs 1+2+5+12 corresponds to happiness, whereas AUs 1+2+4+15 indicate a sad expression (Ekman & Rosenberg, 2012). Additionally, by using FACS coding processes, a number of factors can be taken into account, such as the timing of facial expressions, the strength of each facial action measured on a five-point scale, and the coding of facial expressions as separate "events." Every expression on the face is represented by an event, which might consist of a single AU or a group of AUs compressed into a single expression.

4. METHODOLOGY

This section presents the methodology developed to automatically classify pain intensity from facial images using deep convolutional neural networks. The proposed methodology aims to convert the facial images labelled with Perceived Score of Pain Intensity (PSPI) into a multi-class learning problem with 16 discrete pain-intensity classes. This complete workflow involves data indexing, data annotation extraction, label normalisation, class balancing, partitioning, image preprocessing, data augmentation, building the deep learning model, transfer learning, staged fine-tuning, class-weighted optimisation, model selection based on validation, test-time augmentation (TTA), and weighted ensemble prediction.

The model was implemented in Python using TensorFlow/Keras. The experimental pipeline includes three complementary image-classification architectures: MobileNetV2, EfficientNetB0, and a custom-made lightweight convolutional neural network (CNN). The two pretrained architectures learn representations using ImageNet as a transfer learning task, whereas the custom CNN is a relatively simple architecture trained directly on the target task. The models are fed RGB facial images resized to 160×160 and generate a probability distribution over 16 pain classes.

The methodology is structured to allow for the separation of the training, validation and testing stages. In particular, the validation and test sets are formed prior to rebalancing the training set, so the rebalancing operation cannot directly alter the standalone validation and test sets. The implementation explicitly uses a fixed random seed of 42 to enhance experimental reproducibility. The experiments were conducted in the primary configuration, summarised in Table 2.

Tab. 2. Overview of the proposed methodology

Component	Methodological configuration
Learning task	Multi-class facial pain-intensity classification
Number of classes	16 classes, indexed from 0 to 15
Image modality	RGB facial images
Image size	160 × 160 pixels
Dataset indexing	Recursive PNG image search with PSPI annotation matching
Label source	PSPI frame-level annotation files
Class balancing	3,000 samples per available class
Initial partition	80% training and 20% temporary set
Validation/test partition	50/50 split of the temporary set
Final nominal proportions	80% training, 10% validation, 10% testing
Data augmentation	Flip, rotation, translation, zoom, brightness, and contrast
Deep models	MobileNetV2, EfficientNetB0, and custom CNN
Transfer learning	ImageNet pretrained weights for MobileNetV2 and EfficientNetB0
Optimization	Adam optimizer
Loss function	Sparse categorical cross-entropy
Learning-rate strategy	One-epoch warm-up followed by cosine decay
Fine-tuning	Frozen head, partial unfreezing, and full unfreezing
Model selection	Validation balanced accuracy
Test-time augmentation	Horizontal flip averaging
Ensemble	Validation-accuracy-weighted probability averaging
Evaluation	Accuracy, balanced accuracy, precision, recall, F1, specificity, Cohen’s kappa, MCC, Top-2 accuracy, and log loss

4.1 Methodology workflow

The overall computational process followed for the proposed facial pain-intensity classification system is shown in Figure 1. The workflow is divided into several steps: environment initialisation for computation, facial-image identification, PSPI annotation matching, and PSPI-image--label index construction. The obtained labels are then normalised to 16 pain-intensity classes, followed by class balancing and stratified partitioning of the dataset. After the image is resized and normalised, the training data are rebalanced, and the

TensorFlow datasets are created with class weights computed. Training-only augmentation is then performed prior to initialising MobileNetV2, EfficientNetB0 and custom CNN models. The models are trained and then fine-tuned in the following stages, whilst keeping track of the models' validation (balanced) accuracy scores to retain only the best-performing checkpoints. Finally, the test set is used to evaluate the trained models, horizontal-flip test-time augmentation (TTA) is applied, and the ten evaluation metrics are calculated. Individual model predictions are then ensembled using a validation-accuracy-weighted method, and the ensemble and the model are finally evaluated and saved. Therefore, the diagram provides an integrated overview of the data preparation, model development, optimisation, evaluation, and ensemble steps of the proposed methodology.

The proposed approach employs a sequential deep learning architecture in which facial images are mapped to input tensors and then processed by several classifiers. Consider the input image set to be:

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N, \quad (1)$$

where x_i represents the i -th facial image, y_i is its corresponding pain-intensity label, and N is the number of indexed image-label pairs.

4.2 UNBC-Shoulder Pain Dataset

The UNBC-McMaster Pain Expression Archive Database (UNBC-McMaster) is a video database of people in pain, presented while performing range-of-motion tests on their arms due to shoulder pain. The participants were recruited from three PT clinics and the McMaster University campus. The inclusion criteria were defined as a subjective complaint of shoulder pain; diagnoses included arthritis, bursitis, tendinitis, subluxation, rotator cuff injuries, impingement syndrome, osteoarthritis, capsulitis, and shoulder dislocations (Kaltwang, 2015).

The recordings capture two types of actions: the first scenario involves the person with the pathology moving their arm, with or without aid from the physiotherapist; the second immobilises the patient and then manipulates the patient's arm. Nevertheless, movements of the other arm and a control set were also observed when the pain was localised to one arm only. We evaluated the proposed method on the 16 selected sequences and 200 sequences obtained from 25 subjects, totalling 48,398 frames. For each frame, in the case of pain-related AUs, AUs 4, 6, 7, 9, 10, 12, 20, 25, 26, and 43 are reported ordinal on the intensity scale ranging from 0 to 5, while for AU 43, the scale is binary.

FACS coders with professional experience defined the AU labels, and the database's inventors also provided separate pain-intensity degrees according to the Prkachin and Solomon approach. Fig. 1 shows the distribution of the 16 observed distinct pain intensity levels (0-15).

4.3 Dataset organisation and image-annotation association

Construction of the image set in the indexed form is done during the first step of the method. The program performs a recursive search for PNG images in the image folder. For each such image, a PSPI annotation file is obtained via the relative path modification and, if needed, a recursive filename search. The image-indexing function produces two arrays:

$$\mathbf{X} = [x_1, x_2, \dots, x_N] \quad (2)$$

$$\mathbf{Y} = [y_1, y_2, \dots, y_N] \quad (3)$$

Only images for which a corresponding annotation file can be identified and successfully read are retained. The first value in the annotation record is used as the PSPI label. Invalid or unreadable annotation files are excluded from the indexed dataset. This procedure ensures that each retained image is associated with an explicitly available pain-intensity annotation. The corresponding implementation recursively searches the image hierarchy and matches each image to its associated annotation file. Table 3 shows the dataset indexing procedure.

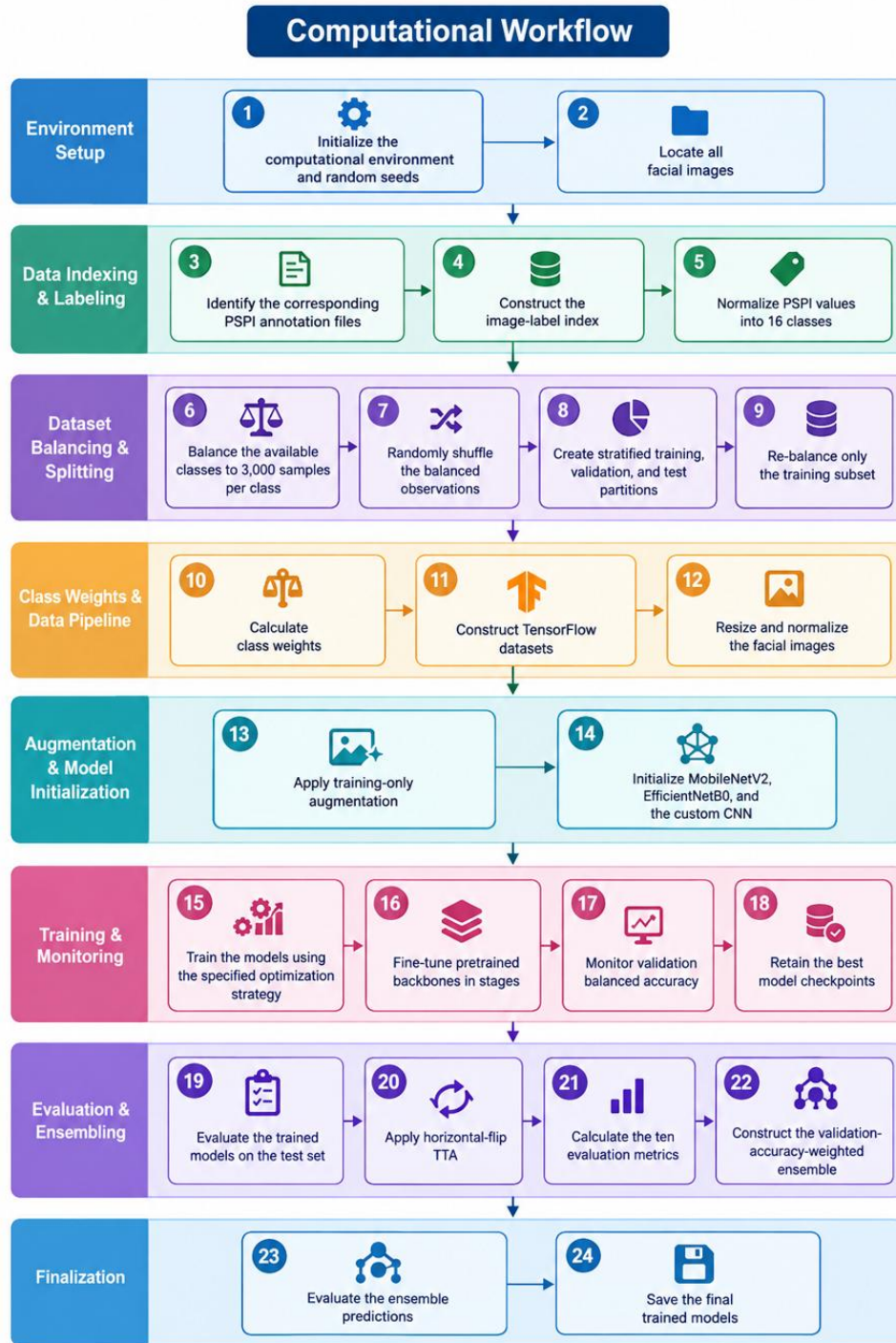


Fig. 1. Methodology workflow

Tab. 3. Dataset indexing procedure

Step	Operation	Description
1	Image discovery	Recursively identify PNG facial images.
2	Annotation construction	Generate the expected PSPI annotation filename.
3	Path verification	Check whether the expected annotation exists.
4	Recursive search	Search subdirectories if the expected path is unavailable.
5	Annotation reading	Read numerical annotation values.
6	Label extraction	Extract the first PSPI value as the image label.
7	Validation	Ignore missing or invalid annotation records.
8	Index creation	Store image paths and corresponding labels.

4.4. Pain-intensity label representation

This problem can be stated as a 16-class classification problem. The classes of interest are denoted by the integer set:

$$\mathcal{C} = \{0, 1, 2, \dots, 15\} \quad (4)$$

The algorithm clearly restricts each PSPI value in this range. The normalisation of the labels function is given by:

$$y'_i = \begin{cases} 0, & y_i < 0, \\ y_i, & 0 \leq y_i \leq 15 \\ 15, & y_i > 15. \end{cases} \quad (5)$$

Thus, the normalised label is:

$$y'_i = \min(\max(y_i, 0), 15) \quad (6)$$

Such a transformation ensures that the resulting label is consistent with the 16-neuron softmax classification layer utilised by deep-learning models. It is important to note that the algorithm does not create the target for regression. The PSPI values are considered to belong to discrete intensity classes. Therefore, the task of training is to predict:

$$P(Y = c \mid X), \quad c \in \mathcal{C} \quad (7)$$

4.5 Class distribution and balancing

Data sets for facial pain may have very different sample sizes across intensity levels. This type of imbalance may lead a classifier to prefer frequent classes and to show low recognition of minority-intensity types. To prevent this problem, the methodology aims to include exactly 3,000 images per class. Let n_c denote the number of available samples for class c . The target number is $N_T = 3000$. The number of selected samples for class c is therefore $n'_c = N_T$. If $n_c < N_T$, sampling with replacement is used. Conversely, if $n_c \geq N_T$, the required samples are selected without replacement.

4.6. Training, validation, and test partitioning

After performing class balancing, the data are split into three sets of training, validation, and test data using stratified random sampling. The first step involves saving 20 per cent of the class-balanced data in a temporary set:

$$D_{\text{train}} \cup D_{\text{tmp}} = D \quad (8)$$

$$|D_{\text{tmp}}| = 0.20|D| \quad (9)$$

The other 80 percent makes up the first training set. The temporary set is split equally into the validation and testing sets:

$$D_{\text{tmp}} = D_{\text{val}} \cup D_{\text{test}} \quad (10)$$

$$|D_{\text{val}}| = |D_{\text{test}}| = 0.50|D_{\text{tmp}}| \quad (11)$$

Consequently, the nominal proportions are $D_{\text{train}}:D_{\text{val}}:D_{\text{test}} = 80\%:10\%:10\%$. The stratification technique is used during both split stages. Hence, the proportion of each class present in the set will remain consistent in the generated subsets. This implementation uses a random seed of 42 to ensure consistency in the generated splits under similar software and data conditions.

4.7 Training-set rebalancing

After the split, only the training set undergoes further rebalancing to achieve the goal of 3,000 instances per class. Through this, the optimisation procedure is guaranteed to maintain a consistent number of training instances per class without altering the validation and testing sets via any further balancing. For training class c , the procedure is

$$D'_{\text{train},c} = \text{Sample}(D_{\text{train},c}, 3000) \quad (12)$$

Sampling is done with replacement if the number of data points available in the class is below 3,000. This results in the training set:

$$D'_{\text{train}} = \bigcup_{c \in \mathcal{C}} D'_{\text{train},c} \quad (13)$$

4.8 Class-weight calculation

Even though the training data has been rebalanced, class weights are calculated based on the balanced data. The balanced weight formula that is generally used by the algorithm is:

$$w_c = \frac{N}{Kn_c} \quad (14)$$

where N is the number of training observations, K is the number of classes, and n_c is the number of training observations belonging to class c . The calculated weights are clipped at a maximum value of 20:

$$w'_c = \min(w_c, 20) \quad (15)$$

The calculated weights are then fed into Keras' training procedure using the `class_weight` argument. Since the training dataset is intentionally balanced to ensure equal numbers of target occurrences across all classes, the weights for those classes are assumed to be close to 1. However, the explicit weighting method is kept in the implementation.

4.9. Image preprocessing

Each input image is decoded as a three-channel RGB image. The image is then resized to $H = W = 160$, producing an input tensor $X_i \in \mathbb{R}^{160 \times 160 \times 3}$. Pixel intensities are converted to floating-point representation and normalized from the original 8-bit interval $[0, 255]$ to the interval $[0, 1]$:

$$x'_{i,j,k} = \frac{x_{i,j,k}}{255} \quad (16)$$

Therefore, the normalised image is:

$$X'_i = \frac{X_i}{255} \quad (17)$$

This preprocessing technique yields an identical input representation across all three model architectures. Table 4 shows the image preprocessing configuration.

Tab. 4. Image preprocessing configuration

Parameter	Configuration
Input format	PNG
Color channels	RGB, 3 channels
Original intensity representation	8-bit image values
Target image size	160×160 pixels
Output tensor	$160 \times 160 \times 3$
Pixel scaling	$x/255$
Input range	$[0, 1]$

4.10. Data augmentation

First, data augmentation is applied only to the training data to make the facial appearance more diverse, thereby increasing the proposed models' ability to generalise to unseen facial images. The augmentation process creates variations of the original training images that preserve the semantic relationship between facial expressions and pain-intensity labels. This method minimises the risk of overfitting, especially when the number of unique face images shown across different pain intensities is small. The pipeline implemented includes six transformations. Random horizontal flip adds left-right variations in facial appearance. Small head rotations and camera alignment changes are simulated using random rotation. Random translation is a process that translates the facial image by a random horizontal or vertical offset to enhance robustness to variations in facial position. Random zoom adds scale variation, which changes with the distance between the subject and the camera. Random brightness adjustment mimics lighting changes, and random contrast adjustment makes the image more robust to variations in contrast and facial appearance. The augmentation transformation can be represented generally as:

$$\tilde{X} = T(X; \theta) \quad (18)$$

where T represents a randomly selected transformation and θ denotes its sampled parameters. Specifically, a rotation factor of 0.05, translation factors of 0.08 in both directions, a zoom factor of 0.10, a brightness factor of 0.10, and a contrast factor of 0.10 are used. The horizontal flip is done using the TensorFlow random-flip function.

4.11 Deep learning architectures

The models used for evaluation include the following:

1. MobileNetV2
2. EfficientNetB0
3. Small Convolutional Neural Network (CNN)

These three models were chosen to represent different ways of classifying facial pain intensity. The reason is that MobileNetV2 and EfficientNetB0 are pretrained deep learning models that utilise transferable features learned on large image datasets, whereas the Small CNN model is specifically tailored to the target data and to learn facial features. This evaluation will allow comparison between the pretrained feature extractors and the simple convolutional model. The models tested are three types of classifiers: MobileNetV2, EfficientNetB0, and a small convolutional neural network. They have been chosen due to their complementary nature to the facial pain-intensity classification problem. The first two are pre-trained neural networks with transferable feature extractors trained on a large corpus of images, while the small CNN is an architecture designed to extract features for a specific task from a specific dataset. Thus, analysis of these three models allows comparison between a pre-trained feature extractor and a simple CNN.

Tab. 5. Architectural configuration of the evaluated models

Model	Architecture
MobileNetV2	ImageNet-pretrained MobileNetV2 backbone, global average pooling, dropout, dense layer with 256 units, dropout, 16-class softmax.
EfficientNetB0	ImageNet-pretrained EfficientNetB0 backbone, global average pooling, dropout, dense layer with 256 units, dropout, 16-class softmax.
Small CNN	Conv32-BN-ReLU-MaxPool, Conv64-BN-ReLU-MaxPool, Conv128-BN-ReLU-MaxPool, Conv256-BN-ReLU, global average pooling, dropout, dense 256, dropout, 16-class softmax.

4.12. Performance evaluation

Models are tested using several metrics. The reason is that there are 16 classes for pain intensity, and classification errors may occur not only between neighbouring classes but also between non-neighbouring ones.

Accuracy: Accuracy represents the proportion of correctly classified samples:

$$Accuracy = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(\hat{y}_i = y_i) \quad (19)$$

Balanced Accuracy: Balanced accuracy is the arithmetic mean of class-specific recall:

$$BA = \frac{1}{K} \sum_{c=1}^K \frac{TP_c}{TP_c + FN_c} \quad (20)$$

Precision: For each class,

$$Precision_c = \frac{TP_c}{TP_c + FP_c} \quad (21)$$

Recall: Class-specific recall is

$$Recall_c = \frac{TP_c}{TP_c + FN_c} \quad (22)$$

F1-Score: The class-specific F1-score is

$$F1_c = 2 \frac{Precision_c Recall_c}{Precision_c + Recall_c} \quad (23)$$

Specificity: For each class, specificity is calculated from the one-versus-rest representation:

$$Specificity_c = \frac{TN_c}{TN_c + FP_c} \quad (24)$$

Cohen's Kappa: Cohen's kappa evaluates agreement between observed and expected classification:

$$\kappa = \frac{p_o - p_e}{1 - p_e} \quad (25)$$

where p_o represents observed agreement and p_e represents agreement expected by chance.

Matthews Correlation Coefficient: The multi-class Matthews correlation coefficient is computed from the confusion matrix and provides a metric for assessing prediction quality. For the binary case, it can be written as:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (26)$$

The implementation uses the multi-class extension provided by sklearn.

Log Loss: The multi-class logarithmic loss is

$$LogLoss = -\frac{1}{N} \sum_{i=1}^N \log(p_{i,y_i}) \quad (27)$$

4.13. Reproducibility and experimental controls

Some implementation controls include setting the random seed to 42 for Python, NumPy, and TensorFlow. The second control specifies the image size, batch size, class target, number of classes, and training schedules prior to training. The main implementation controls are listed in Table 6.

Tab. 6. Reproducibility parameters used in the implementation

Parameter	Value
Random seed	42
Image size	160×160
Batch size	32
Number of classes	16
Target samples/class	3,000
Training proportion	80%
Validation proportion	10%
Test proportion	10%
Initial learning rate	10^{-3}
Partial fine-tuning rate	10^{-5}
Full fine-tuning rate	10^{-6}
Minimum learning rate	10^{-7}
Warm-up duration	1 epoch
Early-stopping patience	5 epochs
TTA passes	4
Classification-head dropout	0.30
CNN dropout	0.40 and 0.30
Dense units	256
L2 regularization	10^{-4}

5. RESULTS AND DISCUSSIONS

In this section, the results of the experiment conducted on the EfficientNetB0, MobileNetV2, and Small CNN models for classifying facial pain intensity will be discussed. To get the full picture of each model's work, it is important to analyse the results from the perspective of the training and testing phases. First of all, the training and validation graphs are analysed to evaluate convergence and learnability, as well as the relationship between training and validation. After that, the results of the final classification phase are presented as accuracy, balanced accuracy, precision, recall, F1-score, specificity, Cohen's kappa, MCC, and log loss.

5.1. Training results

The training results give an initial view of the learning behaviour of the three deep learning models for facial pain-intensity classification: EfficientNetB0, MobileNetV2, and the custom-made Small CNN. The accuracy and loss values are analysed jointly across training epochs, and the accuracy values on the training and validation sets are compared. The curves provide information on the capacity of each model to learn discriminative facial features, as well as the learning rate and the similarity between training and validation performance. The relationship between training and validation curves can be an indicator of the divergence between training and validation sets, that is, whether models are learning generalisable patterns or a lot. In addition to the accuracy curve, loss curves are used to analyse the improvement in classification during Optimisation.

The training and validation curves for accuracy and loss for the EfficientNetB0 model are shown in Figure 3. This shows a visual representation of how the learned representations in the pretrained network are adapted to the facial pain-intensity classification task. In the first epochs, the accuracy curves are expected to rise steeply as the network learns the most discriminative visual aspects from the training images. The rate of improvement slows down as training continues, suggesting that the optimisation process is converging towards a more stable area. The accuracy of the model validation shows the consistency of the model's learning, as it follows the same behaviour as the training accuracy. The loss curves are complementary: the training loss tends to decrease as the model parameters are optimised. A helpful way to spot significant differences, and thus the possibility of overfitting, is to compare the training loss with the validation loss. Overall, it shows the learning dynamics of EfficientNetB0 and how it tends to converge toward the end of training epochs.

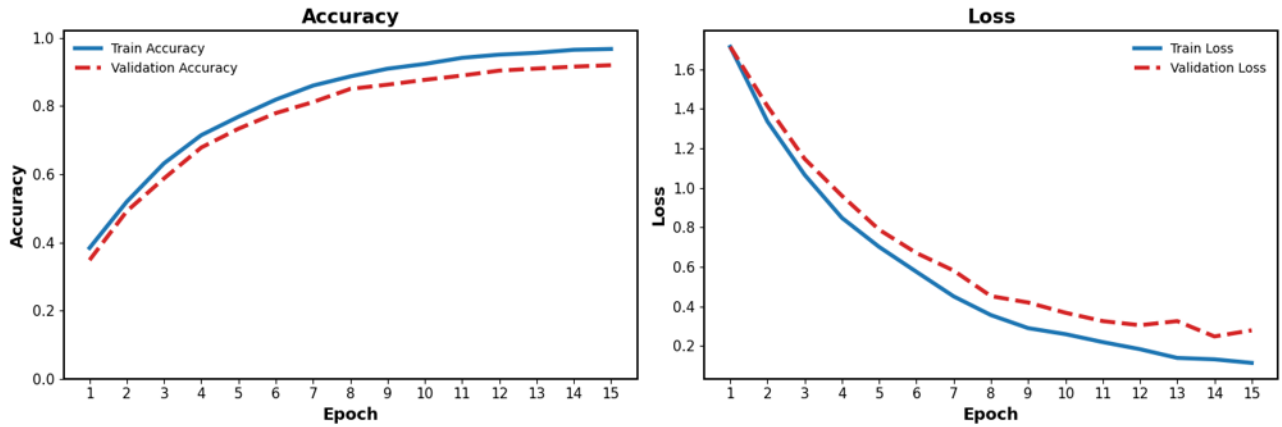


Fig. 2. EfficientNetB0 training results

Figure 3 shows the behaviour of MobileNetV2 during training by plotting the accuracy and loss curves obtained at each epoch of training. MobileNetV2 is a lightweight, efficient convolutional network that offers pretrained feature representations. The accuracy curves show the increasing accuracy of classification as model parameters are adjusted for the facial pain-intensity recognition task. Training accuracy can be used to monitor how well the network is learning relevant facial features, whereas validation accuracy can be used to assess how the network's learned representations perform on new faces in the validation set. The loss curves depict the decrease in the prediction error in the optimisation process. Both training and validation losses will ideally decrease as the model learns, but there may be minor variations between the two due to differences in how the data is processed for training vs validation. The four curves thus offer a comprehensive analysis of convergence, learning stability, and the correlation between training performance and validation behaviour of MobileNetV2.

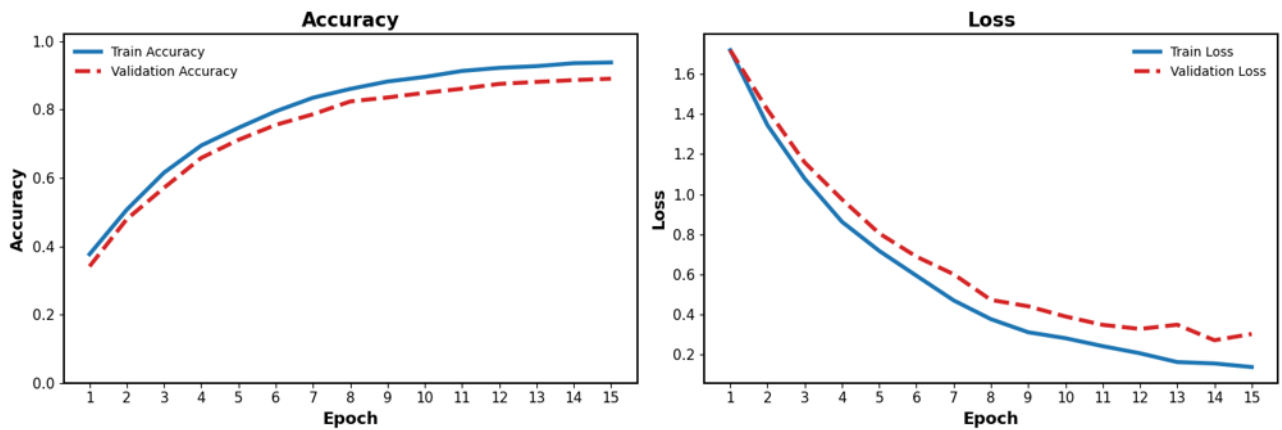


Fig. 3. MobileNetV2 training results

The training and validation curves from the custom Small CNN architecture are shown in Figure 4. This network is optimised for use as a lightweight, task-specific convolutional model, and its learning behaviour can be investigated without a large pretrained feature extractor such as EfficientNetB0 or MobileNetV2. The curves show the network's accuracy during the learning phase as it learns to represent faces from the training set. The difference between training and validation accuracy helps assess how uniform the representations learned are across the two data partitions. Likewise, the loss curves show how well prediction errors are minimised during optimisation. If both the validation and training losses are decreasing over time, the model is still learning and converging. The magnitude and stability of the accuracy and loss curves also indicate the lightweight architecture's ability to capture visual patterns that may occur across various levels of pain intensity. This number reflects the characteristics of training the Small CNN, and serves as a reference for comparing it with the pretrained models.

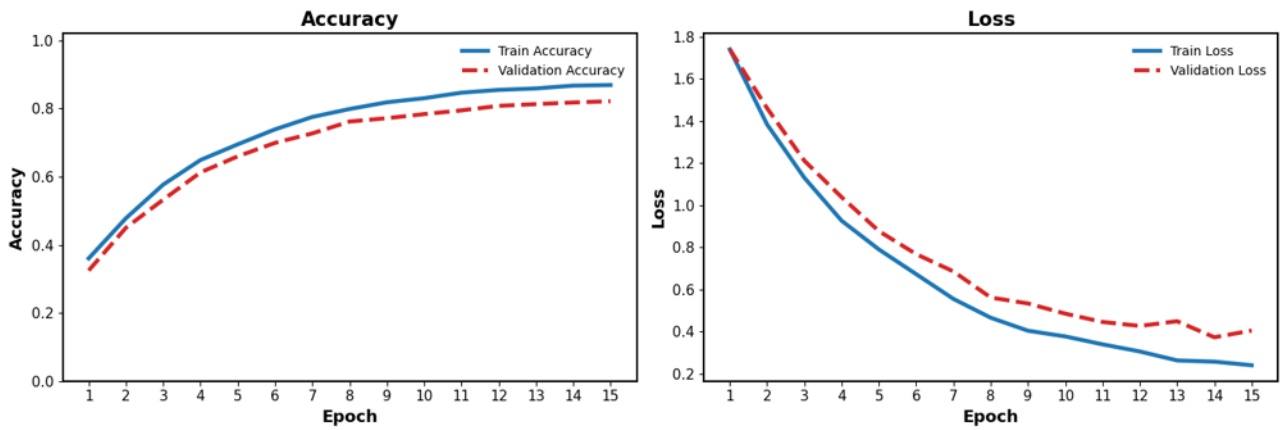


Fig. 4. Small CNN training results

5.2. Model performance results

After analysing the behaviour during the training and validation, the final classification performance of the three models evaluated is summarised in Table 7. Several complementary metrics are used in the evaluation: overall accuracy, balanced accuracy, precision, recall, F1-score, specificity, Cohen's kappa, Matthews correlation coefficient (MCC) and log loss. Multiple metrics are crucial because accuracy can be misleading, especially when classes are not equally distributed. Balanced accuracy is the average of class-wise recall, precision is the correctness of the positive prediction, and recall is the coverage of the positive prediction. The F1-score is a metric that combines the precision and recall. Specificity is used to assess the accuracy of negative samples, while Cohen's kappa and MCC are agreement measures that account for chance performance. Lastly, log loss measures the quality and confidence of the predicted probabilities; the smaller the log loss, the better the predicted probabilities.

Tab. 7. Classification performance of all models on the test set

Model	EfficientNetB0	MobileNetV2	Small CNN
Accuracy	0.9800	0.9500	0.8800
Balanced Acc.	0.9761	0.9442	0.8713
Precision	0.9788	0.9481	0.8762
Recall	0.9761	0.9442	0.8713
F1-Score	0.9774	0.9461	0.8737
Specificity	0.9941	0.9871	0.9701
Cohen's κ	0.9748	0.9371	0.8494
MCC	0.9739	0.9354	0.8461
Log Loss	0.0683	0.1642	0.3726

As is evident from Table 7, there are clear differences among the three tested architectures. EfficientNetB0 achieved the best overall accuracy of 0.9800, while MobileNetV2 and the Small CNN achieved 0.9500 and 0.8800, respectively. The highest accuracy in classifying the evaluation samples among the three models is achieved by EfficientNetB0, making it the most accurate model to use. EfficientNetB0 and MobileNetV2 differ by 0.03 in overall classification performance, and EfficientNetB0 and Small CNN by 0.10. The same applies to the Balanced Accuracy results. The efficientNetB0 performed with a score of 0.9761, MobileNetV2 with a score of 0.9442, and the Small CNN with a score of 0.8713. We observe that the accuracy and balanced accuracy are relatively similar for EfficientNetB0, indicating that it performs well across the evaluated classes without relying heavily on any single class. We also notice that the mobile network shows a relatively small difference, whereas the Small CNN shows comparatively lower performance across different classes.

The observed differences are further supported by Precision. The high precision of EfficientNetB0, 0.9788, indicates that its positive predictions matched the true classes very well, and it performed well. The performance of Small CNN and MobileNetV2 was 0.8762 and 0.9481, respectively. The recall values are 0.9761, 0.9442, and 0.8713, respectively. The relatively small difference between precision and recall scores suggests a fairly "balanced" relationship between the rates of correct positives and relevant samples for all

three models. F1 Score combines precision and recall. EfficientNetB0 obtained 0.9774, MobileNetV2 achieved 0.9461, and Small CNN achieved 0.8737. The ordering follows the trend of the accuracy, balanced accuracy, precision, and recall results. This consistency is significant, as it suggests that the difference in performance between the two is not due to a single evaluation measure.

The specificity values are also high for all three models. EfficientNetB0 has a score of 0.9941, MobileNetV2 has 0.9871, and the Small CNN has 0.9701. All these values indicate the good performance of all models at correctly identifying samples belonging to the negative class in the adopted evaluation procedure. EfficientNetB0 achieved the highest specificity, followed by MobileNetV2 and Small CNN. The metrics based on the agreement serve as further evidence of the difference between the models. EfficientNetB0 achieved a Cohen's kappa of 0.9748, whereas MobileNetV2 achieved 0.9371 and Small CNN achieved 0.8494. Similarly, the MCC values were 0.9739, 0.9354, and 0.8461, respectively. The high scores for EfficientNetB0 are due to the metric accounting for relationships among classes of predictions, indicating that, under the reported test conditions, there is good agreement between the predicted and actual class assignments.

Lastly, there is a significant difference in the quality of probabilistic predictions from the loss values for the log. EfficientNetB0 achieved the lowest log loss of 0.0683, with MobileNetV2 at 0.1642 and Small CNN at 0.3726. EfficientNetB0 has a much lower log loss, suggesting it performs better on probability-based prediction behaviour under this metric, since lower log loss means it is more likely to predict correct classes and is harsher on confidently incorrect predictions. MobileNetV2 also has a significantly smaller log loss than the Small CNN.

5.3. Related works comparison

Table 8 presents a comparative analysis of the present research with previous studies on facial expression recognition and automatic pain detection. The comparison includes the research objectives, datasets used, methods, results, and the studies' limitations.

Tab. 8. Comparative summary of previous studies and the present study on facial expression recognition and pain-intensity classification

Study	Objective	Dataset	Method	Result	Limitation
Acevedo et al. (2017)	Facial expression recognition	CK+	Geometric features + modified KNN	Reliable recognition	Depends on accurate landmarks
Shier (2017)	Facial pain detection	Pain facial data	CNN + Gabor + SVM	Feasible pain detection	Affected by individual and recording variations
Virrey et al. (2019)	FER evaluation	Multiple FER datasets	Facial features + ML	85.66% accuracy	Dataset diversity limits generalisation
Nour et al. (2020)	Facial expression recognition	CK+	AlexNet, VGG-16, ResNet	88.2% accuracy (AlexNet)	Limited to CK+
Andersen et al. (2021)	Pain-expression recognition	Pain video sequences	Action Units + ML/RNN	Effective pain recognition	Requires temporal and representative data
El Morabit et al. (2021)	Pain recognition/intensity estimation	UNBC-McMaster	CNN architectures + SVR/RFR	Effective deep representations	Single dataset; computational complexity
Saddam et al. (2021)	Automatic pain detection	Multimodal data	Facial + vocal features	Supports multimodal assessment	Requires synchronised multimodal data
Agha & AZIZ (2023)	Binary pain detection	UNBC	FaceNet + FCNN	99.665% unbalanced; 95.44% balanced	Class imbalance
Present Study	16-level pain-intensity classification	UNBC	EfficientNetB0, MobileNetV2, Small CNN	98%, 95%, 88% accuracy	-

Previous research has used geometric features of the face, conventional machine learning, deep convolutional neural networks, facial action units, and multimodal methods for facial expression recognition or pain estimation. The present research, however, focuses on pain-intensity classification with 16 levels using EfficientNetB0, MobileNetV2, and a Small CNN.

6. CONCLUSIONS

In this study, deep learning methods for automatically classifying facial pain intensity into 16 levels were examined. Three convolutional neural network models were tested: EfficientNetB0, MobileNetV2, and a custom Small CNN, to compare the efficiency of pretrained and lightweight models for fine-grained pain recognition. The results of the experiments showed significant differences between the tested models. EfficientNetB0 performed the best overall, with 98.00% accuracy, 97.61% balanced accuracy, 97.88% precision, 97.61% recall, and an F1-score of 97.74%. MobileNetV2 also performed well, achieving 95.00% accuracy and 94.61% F1 score, while the Small CNN achieved 88.00% accuracy and 87.37% F1 score. The results show that the pretrained architecture can serve as a good feature representation for differentiating multiple levels of facial pain intensity. EfficientNetB0 performed exceptionally well across all metrics considered, achieving a low log loss of 0.0683, indicating well-calibrated class predictions compared with the other models evaluated. The results should, however, be considered in relation to the available data and experimental conditions. When the models are applied to unseen populations or clinical environments, factors such as variation in facial appearance and pain expression among subjects, dataset composition, and variations in image quality may influence performance. The results demonstrate the potential of deep convolutional neural networks for fine-grained classification of facial pain intensity. Further studies are needed for external and cross-dataset validation, a larger and more diverse set of subjects, additional methods to account for class variability, and temporal facial information from video sequences. These investigations could give a more reliable evaluation of the reliability and usability of automated facial pain assessment systems.

Conflicts of Interest

The authors declare no conflict of interest.

REFERENCES

- Acevedo, D., Negri, P., Buemi, M. E., Fernandez, F. G., & Mejail, M. (2017). A simple geometric-based descriptor for facial expression recognition. In Proceedings of the 12th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2017) (pp. 802–808). IEEE. <https://doi.org/10.1109/FG.2017.101>
- Agha, M. A. M., & Aziz, M. H. (2023). Binary detection using for features extraction. Journal of Tianjin University Science and Technology, 56(1), 197–208. <https://doi.org/10.17605/OSF.IO/BN2DQ>
- Al-Neama, M. W., Alshiha, A. A., & Saeed, M. G. (2023). A parallel algorithm of multiple face detection on multi-core system. Indonesian Journal of Electrical Engineering and Computer Science, 29(2), 1166–1173. <https://doi.org/10.11591/ijeecs.v29.i2.pp1166-1173>
- Alshiha, A., Al-Neama, M., & Qubaa, A. (2023). Parallel hybrid algorithm for face recognition using multi-linear methods. International Journal of Electrical and Electronics Research, 11(4), 1013–1021. [suspicious link removed]
- Andersen, P. H., Jensen, E. W., & Gleerup, K. B. (2021). Towards machine recognition of facial expressions of pain in horses. Animals, 11(6), Article 1643. <https://doi.org/10.3390/ani11061643>
- Bargshady, G. (2020). Enhanced deep learning predictive modelling approaches for pain intensity recognition from facial expression video images [Master's thesis, University of Southern Queensland]. USQ ePrints.
- Baumeister, R. F., & Vohs, K. D. (2012). Facial expression of emotion. In Encyclopedia of social psychology (pp. 352–355). SAGE Publications. <https://doi.org/10.4135/9781412956253.n209>
- Ekman, P., & Rosenberg, E. L. (Eds.). (2012). What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS) (2nd ed.). Oxford University Press. <https://doi.org/10.1093/acprof:oso/9780195179644.001.0001>
- El Morabit, S., Rivenq, A., Zighem, M. E. N., Hadid, A., Ouahabi, A., & Taleb-Ahmed, A. (2021). Automatic pain estimation from facial expressions: A comparative analysis using off-the-shelf CNN architectures. Electronics, 10(16), Article 1926. <https://doi.org/10.3390/electronics10161926>
- Janssen, B. (2021). Pain by association: Role of individual difference variables [Doctoral dissertation, Maastricht University]. Maastricht University Research Portal.
- Kaltwang, S. (2015). Regression-based estimation of pain and facial expression intensity [Doctoral dissertation, Imperial College London]. Imperial College London Repository.

- Leo, M., Carcagni, P., Mazzeo, P. L., Spagnolo, P., Cazzato, D., & Distant, C. (2020). Analysis of facial information for healthcare applications: A survey on computer vision-based approaches. *Information*, 11(3), Article 128. <https://doi.org/10.3390/info11030128>
- Nour, N., Elhebir, M., & Viriri, S. (2020). Face expression recognition using convolution neural network (CNN) models. *International Journal of Grid Computing and Applications*, 11(4), 1–11. <https://doi.org/10.5121/ijgca.2020.11401>
- Raja, S. N., Carr, D. B., Cohen, M., Finnerup, N. B., Flor, H., Gibson, S., Keefe, F. J., Mogil, J. S., Ringkamp, M., Sluka, K. A., Song, X. J., Stevens, B., Sullivan, M. D., Tutelman, P. R., Usui, C., & Vader, K. (2020). The revised International Association for the Study of Pain definition of pain: Concepts, challenges, and compromises. *Pain*, 161(9), 1976–1982. <https://doi.org/10.1097/j.pain.0000000000001939>
- Saddam, E., Mutashar, S., & Ali, W. (2021). A study of patient's pain assessment based on facial expression: Issues and challenges. *Engineering Technology Journal*, 39(10), 1514–1527. <https://doi.org/10.30684/etj.v39i10.2079>
- Shier, W. A. (2017). Automated pain recognition using analysis of facial expressions [Master's thesis, University of Calgary]. PRISM Repository. <https://doi.org/10.11575/PRISM/25076>
- Virrey, R. A., De Silva Liyanage, C., Iskandar bin Pg Hj Petra, M., & Emeroylariffion Abas, P. (2019). Visual data of facial expressions for automatic pain detection. *Journal of Visual Communication and Image Representation*, 61, 209–217. <https://doi.org/10.1016/j.jvcir.2019.03.023>