


A real-time computational approach for human facial expression recognition based on landmark feature extraction

 <https://doi.org/10.56238/sevned2024.004-006>

Dennis P. Lopez¹, Felipe Z. Canal², Gustavo G. Scotton³, Eliane Pozzebon⁴ and Antonio C. Sobieranski⁵

ABSTRACT

Real-time human facial expression recognition plays a significant role in many application areas, including human-computer interaction, business intelligence, video surveillance, and robotics. Based on facial expressions, computers can interpret human feelings and psychological stages to provide more realistic approximations in real-world applications. This paper proposes a simple but effective solution for real-time Facial Emotion Recognition (FER), using a mask of the most relevant facial features as input data for a machine-learning approach. For this, a compact Convolutional Neural Network (CNN) classifier associated with a feature extraction layer was used to provide an end-to-end solution that can detect facial expressions from videos with good accuracy rates. The proposed approach was validated using a combination of different facial emotion datasets available in the literature, whose precision rates are considerably better than those provided by the state-of-the-art methods. Score rates of 96.83%, 98.58%, and 98.57% were obtained for the JAFFE, RaFD, and CK+ datasets, respectively, indicating that the presented approach is a promising solution for FER in real-time applications.

Keywords: Real-time facial expression recognition, Emotion classification, Facial patterns extraction, Convolutional Neural network.

¹ Department of Computing – DEC, Federal University of Santa Catarina, Brazil.
E-mail(s): dppazlopez@gmail.com

² Department of Computing – DEC, Federal University of Santa Catarina, Brazil.
Graduate Program in Information and Communication Technologies, Federal University of Santa Catarina, Brazil.
E-mail: felipe.canal@grad.ufsc.br

³ Department of Computing – DEC, Federal University of Santa Catarina, Brazil.
E-mail: gustavo.gino@outlook.com

⁴ Department of Computing – DEC, Federal University of Santa Catarina, Brazil.
Graduate Program in Information and Communication Technologies, Federal University of Santa Catarina, Brazil.
E-mail: eliane.pozzebon@ufsc.br

⁵ Department of Computing – DEC, Federal University of Santa Catarina, Brazil.
Graduate Program in Information and Communication Technologies, Federal University of Santa Catarina, Brazil.
E-mail: a.sobieranski@ufsc.br

INTRODUCTION

Emotion recognition from facial expressions is a relatively simple task for humans, being part of regular day-to-day non-verbal communication, but complex to be expressed analytically [1]. Humans are capable of inferring emotions through other means, such as body language, tone of voice, and spoken words. However, facial features significantly contribute to the expression of emotions since the face is the main focus of attention during communication. Many studies support the premise that some facial behaviors are universally linked to emotions, regardless of cultural and ethnic differences [2], and can be computationally reproduced.

Although natural for humans, Facial Emotion Recognition (FER) poses a significant challenge for machines since faces can have many attributes such as aging, shape, skin color, beards, and scars [3]. Additionally, faces may include some variability, such as glasses, uneven illumination, and occlusions, and can be visualized from different perspectives. All these challenges can be considered complex for a computational method to discriminate information from the faces and use only the relevant features needed to classify emotion patterns.

A general computational trend for the FER problem is to extract some features from the facial image and pass them through a classifier, thus abstracting its complexity and making predictions possible. Some of the most used feature extraction techniques are the classical approaches, also known as hand-crafted methods. Some examples are (i) the geometric-based features [4, 5], which measure the position and shape of different face components; and (ii) appearance features [5], which involve more complex details like edges and textures. There are also other methods used for specific feature extraction, like the Pyramid Histogram of Gradients (PHOG) [6], Local Phase Quantisation (LPQ) [6, 7], and Gabor Wavelets Features [8]. However, with the recent emergence of Deep Learning techniques, many methods based on Convolutional Neural Networks (CNNs) are becoming general problem solvers in real-world applications. Well-known networks such as AlexNet, GoogLeNet, VGG, and ResNet [9–11] have already been used in several applications and in the FER area.

With CNNs, an end-to-end solution can be provided directly, as the network can learn the most relevant features from the data. However, many samples and considerable computational effort are required to train the model correctly, which leads to better classification results and generalizability. This characteristic is the main difference compared to traditional hand-crafted methods, which are tailored explicitly in advance and designed by an expert.

Nonetheless, many of the FER approaches found in the literature depend mainly on the dataset used to train the models, which in most cases is a single one. Datasets containing face images or videos developed explicitly for FER are no novelty. The first studies date back to 1969 [12], and many facial expression datasets have emerged since then, varying their

acquisition environment, the number of recognizable expressions, and the subject age group, among other features. Some popular datasets include JAFFE, CK+, MMI, RaFD, and AffectNet [13–15]. A too-homogeneous or small dataset can cause overfitting and has little representation for some classes of data, and could lead to some unwanted bias. It can even have mislabeled data and make the learning process more complicated – it was reported that the percentage of correctly labeled data for the FER dataset was about 65% [16].

With these limitations in mind, this paper presents a simple but effective hybrid CNN solution explicitly designed for FER, capable of real-time video processing, and developed considering a mixed dataset. For this goal, the proposed method uses a simple geometric feature extraction technique, combining landmark descriptors from faces and using them to train and test CNN models. The purpose of using geometric features is their invariance to real-world limitations such as skin tones, lighting, and angles. Furthermore, during the training step, the input data (mixed from three databases) combined seven types of emotions: (i) neutral, (ii) happiness, (iii) sadness, (iv) surprise, (v) anger, (vi) fear, and (vii) disgust. The results showed precision rates of 96.83%, 98.58%, and 98.57% for the JAFFE, RaFD, and CK+ datasets, respectively, showing marginally better accuracy rates when compared to the methods found in the literature.

The remainder of this work is organized as follows: Section 2 discusses some highlighted works regarding the FER area. Section 3 describes the proposed method and its implementation aspects in detail, followed by Section 4, demonstrating the obtained results. Finally, the conclusions, discussion, and further works are presented in Section 5.

RELATED WORK

In the literature, it is possible to identify two clear trends regarding the FER area: (i) the classical approaches, also known as hand-crafted features selection methods; and (ii) the Convolutional Neural Network counterpart.

Classical approaches usually involve some technique to obtain a set of features and some metric or classifier to discriminate input vectors for decision-making. Typical examples of classical approaches are the well-known Support Vector Machines (SVMs), decision trees, random forests, or other non-connectionist machine learning techniques to classify the input data. Those methods are said to be hand-crafted since the expert/researcher generally designs the feature extraction method according to the particularity of the problem. Although this tailored approach design can produce a specific method very limited to a subset of input data (used to design the algorithm's model), it generally requires fewer input samples. Some examples of the classical approaches are:

- The method proposed by Happy and Routray [17] performs expression recognition by extracting features from selected facial patches around the face while using an SVM as the classifier. The authors reached 94.09% and 91.8% accuracy rates with this system in the CK+ and JAFFE datasets.
- Using a combination of the oriented FAST and rotated BRIEF features (ORB) and the Local Binary Patterns (LBP) to extract parts from facial expressions, Ben Niu et al. [18] proposed an approach without a CNN, achieving a precision of 88.5%, 93.2%, and 79.8% for the JAFEE, CK+, and MMI datasets, respectively.
- Abdulrahman and Eleyan [19] also proposed a FER approach based on the LBP and the Principal Component Analysis (PCA) algorithms while using an SVM classifier. They performed several experimental tests with the JAFFE and their newly introduced database MUFE (Mevlana University Facial Expression), achieving average results of 87% and 77%, respectively.
- In the paper presented by Salman et al. [20], a new extraction method is proposed based on a geometric approach, where six distances are calculated to measure the different parts of the face that better describe a facial expression, and a decision tree is applied to the JAFEE and COHEN databases. They obtained 89.20% and 90.61% recognition rates for each database.
- Pu et al. [21] proposed a novel framework for FER by recognizing AUs from image sequences using two-fold random forest classifiers. They achieved an accuracy of 96.38% for the CK+ database.

With the emergence of CNNs, several applications were proposed with this technique in mind as a general problem-solver. Their principal aspect is their ability to generalize problems using a generic feature extraction method, paving the way to learn the most relevant characteristics directly from the input data and using them in a neural network classifier. However, training a CNN requires large input data and considerable computational time to provide robust and accurate classifiers. Some works using the CNN counterpart are detailed below:

- Chen et al.'s [10] method calculates the motion of a facial mask for each emotion of the training dataset and uses the results to train a facial mask estimator. The images to be recognized are combined with their facial mask (obtained by passing the image through the estimator) and the classifier to convey the expected emotion. The results obtained showed an accuracy of 98.06%, 82.74%, and 61.52% for the CK+, MMI, and AffectNet datasets.

- Cugu et al. [22] introduced a less than 1MB network with 65K parameters capable of executing at 1851 frames per second on an Intel i7 CPU for lightweight and speed. However, this came with an accuracy penalty cost, achieving 84.8% on the CK+ dataset.
- Burkert et al. [23] proposed a deep CNN with a parallel Feature Extraction block (*FeatEx*), inspired by the GoogleNet network, as the central component. This block processes the input data into two parallel paths with different filter sizes to better capture the varying scales of a face within an image. Concatenating two *FeatEx* blocks and passing the result to a classifier, they achieved 99.6% and 98.63% accuracies in the CK+ and MMIdatasets, respectively.
- Using the recent method of attention networks, Meng et al. [24] created an end-to-end system capable of facial expression recognition from videos. The recognition consists of three distinct steps: (i) frame preprocessing (face alignment and others); (ii) feature extraction; and (iii) classification, which is the same procedure used in the method proposed in this paper. The accuracies reached by this network for the CK+ and AFEW 8.0 datasets were 99.69% and 51.18%.
- Minaee and Abdolrashidi [25] also used attention CNNs to create an end-to-end system that learns the relevant features and focuses on the essential parts of a facial expression. The model was trained using the FER2013, CK+, JAFFE, and FERF databases, reaching accuracies of 70.02%, 99.3%, 92.8%, and 98.0%, respectively.
- The work presented by Kai Wang [11] states that the difficulty in recognizing facial expressions is due to ambiguousness, low-quality facial images, and the subjectiveness of annotators. The solution proposed by the author is based on the *Self-Cure* Network (SCN), a simple but efficient network based on traditional CNNs. According to the author, this approach can suppress uncertainties and prevent CNNs from adjusting excessively to ambiguous facial images. They obtained an accuracy of 88.14% in RAF-DB, 60.23% in AffectNet, and 89.35% in FERPlus.

Although deep learning techniques can be used to have an end-to-end system that learns everything from the features to the classification, that might not always be the best solution. Using traditional features combined with neural networks can yield even better results in some cases. The recent resurgence in the field owes this, given the advances in computing capabilities and the repeated breaking of records using old (e.g., CNNs) and new (e.g., transformers) methods. These techniques can be used not just to create the classifier but also to learn the most relevant features.

Based on the works found in the literature, also summarized in Table 1, it is possible to see that the problems should revolve around the difficulty of recognizing the diversity and ambiguity of the face in the facial emotions. In our approach, we demonstrate that the problem is not related to the complexity of the neural network itself but to the quality of the input data passed to it. We developed a hybrid approach whose features are based on a robust geometric extraction technique that resolves the limitations presented by the authors and provides a fast and practical approach in the FER context, regardless of ethnicity, race, wrinkles, scars, beards, or glasses variations. Additionally, the proposed method is effective against ambiguous or undefined emotions, as will be discussed in Section 4.

Table 1 Comparison of the FER methods found in the literature.

Author	Year	Method	Database	Overall
Happy and Routray	2014	SVM	CK+ JAFPE	94.09% 91.80%
Burkert et al.	2015	CNN	CK+ MMI	99.60% 98.63%
Cugu et al.	2017	Viola-Jones and CNN	Oulu-CASIA CK+	62.69% 84.80%
Meng et al.	2019	CNN	CK+ AFEW 8.0	99.69% 51.18%
Chen et al.	2019	CNN	CK+ MMI AffectNet	98.06% 82.74% 61.52%
Minace and Abdolrashidi	2019	CNN	FER2013 CK+ JAFPE FERG	70.02% 99.30% 92.80% 98.00%
Kai Wang et al.	2020	SCN	RAF-DB AffectNet FERPlus	88.14% 60.23% 89.35%
Ben Niu et al.	2021	LBP and ORB	JAFPE CK+ MMI	88.50% 93.20% 79.80%

METHODOLOGY

The methodology described in this paper can be regarded as a hybrid approach since a Geometric Feature Extraction technique is used to obtain the most relevant features from facial images. These features are then passed through a CNN classifier designed specifically to classify human facial expressions into predetermined categories of emotions. The purpose of using a hybrid solution is to take advantage of its ability to quickly predict the results from a video feed in real-time and continuously.

A general procedure for the proposed methodology is illustrated in Figure 1. First, a data augmentation technique is applied to the input dataset to increase the number of

samples and provide a robust dataset to train the CNN model while avoiding the overfitting effect. Then, a preprocessing step is introduced to standardize the size and the alignment of the faces presented in the images. From there, a Feature Extraction technique is employed to obtain a set of geometric features from the facial image and convert them into a binary mask. Finally, an algorithm is used in the convolutional neural network model to predict facial expressions. These steps are described in detail in the following subsections.



Furthermore, the code implementation is based on the Python3 language

[26] due to its robustness for prototyping and testing, including libraries to support the presented method, such as *OpenCV* [27], *Keras* [28], *TensorFlow* (GPU version) [29], and *Dlib* [30].

INPUT IMAGE

The data augmentation technique applied to every input image consisted of rotating an image at a random angle (between 10 and 30 degrees) in the clock-wise and counterclockwise directions, producing three later mirrored images, and increasing by six times the original size of the dataset. The facial emotion datasets used to validate the proposed methodology are described below:

- **JAFFE** – The *Japanese Female Facial Expression* dataset (JAFFE) is a classical facial expression source from 1998 [13]. It comprises 253 images from 10 female Japanese models expressing seven common emotions: neutral, anger, disgust, fear, happiness, sadness, and surprise. The images are grayscale and have a resolution of 256x256 pixels. Although it is a simple dataset characterized by only a few images, it has been widely used academically worldwide.
- **RaFD** – The Radbound Faces Database (RaFD), presented in 2010, comprises a set of images from 67 models instructed to show eight different expressions (the seven shown in JAFFE plus another one for contempt) and to look in three directions (right, ahead, and left) [15]. For every expression, the subjects' photographs correspond to 5 camera angles (-90° , -45° , 0° , 45° , and 90°), totaling 120 images per model. This paper only used the front-facing images (0°) for the training and testing phases.
- **CK+** – The *Extended Cohn-Kanade* dataset (CK+) is an extension presented in

2010 as a replacement for the already popular at the time Cohn-Kanade dataset (CK) released in 2000 [14]. It comprises 593 images from 123 subjects showing eight emotions (the same as for RaFD) with a spatial image resolution of 640x490 pixels and labeled with the Facial Action Coding System (FACS), apart from the facial expression classification.

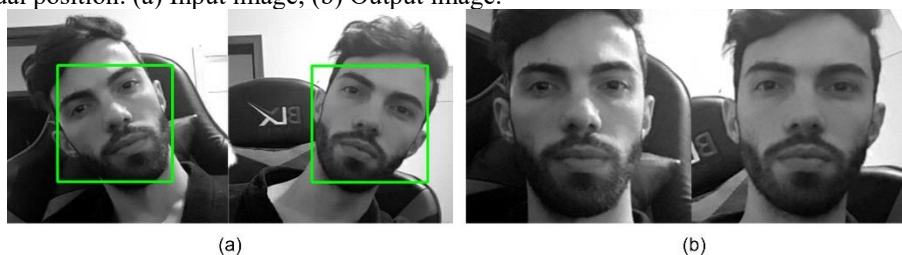
PREPROCESSING

A simple preprocessing stage is introduced to decrease dimensionality and to provide a set of regular images for the feature extraction phase. The result of this procedure is demonstrated in Figure 2, and it is mainly comprised by:

Color simplification – before any extraction, the input image is converted into grayscale, as this color scale has little significance on the facial expression, and the brightness variation is sufficient to detect the features;

- Face regularization – next, a face-centering process is performed by using the coordinates of the nose's center as an alignment reference point;
- Face repositioning – a rigid rotation is then performed to present the eyes in a horizontal line, leaving both eyes on the same y-axis level and ensuring identical aligned images for the dataset;
- Spatial resolution reduction – finally, the input image is resized to normalize the camera's distance (which causes the size of the face in the image to increase or decrease) and to ensure that the faces' sizes are equivalent.

Fig. 2 Regardless of the angle of the face to the camera, the algorithm can align, center, and resize the face to an approximately equal position: (a) Input image; (b) Output image.



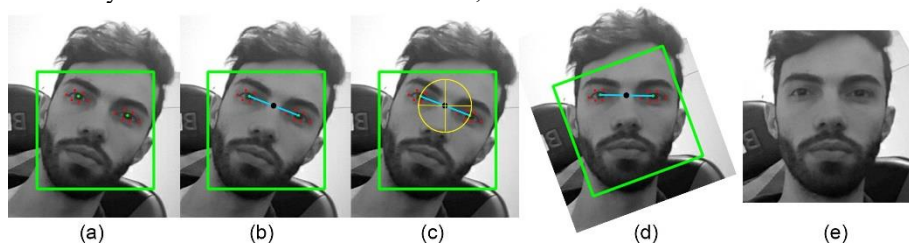
These steps were implemented with the help of the *Dlib* library, a toolkit designed explicitly for machine learning, image processing, and linear algebra applications. Initially, the face detector function [31], which detects a face based on the HOG method and a linear SVM, is used, followed by a proprietary algorithm to calculate the image's required adjustments and alignments. The functioning of this algorithm is detailed in the following subsection.

Alignment Algorithm

The most suitable way to align the face in the image corresponds to positioning the eyes in a horizontal plane register. For this, both eyes' centroids are initially calculated by considering each eye's average points (x and y) and dividing by their cardinality (n). This step is demonstrated in Figure 3a. Next, the difference between the left and right eyes (Δ) is computed in the x and y directions. From there, the eyes' center point is obtained, as shown in Figure 3b. Based on this information, the angle required to rotate the image, and ensure that the y -axis of both eyes is located in the same position, is obtained by computing the arc-tangent for the Δ values in the x and y directions. The result of this step is demonstrated in Figure 3c. Finally, to maintain a standardized dataset, the face image is resized, ensuring that all the images share the same size and, thus, the same input mask size needed by the neural network. This stage is achieved by using the portion of the distance between the eyes (Δ_x and Δ_y) and calculating a scale factor.

Moreover, the image rotation is performed by obtaining the transformation matrix (M) with the help of the `getRotationMatrix2D` and `warpAffine` functions from the *OpenCV* library. The result and the final output of the alignment algorithm are shown in Figures 3d and 3e, respectively.

Fig. 3 Preprocessing alignment algorithm workflow: (a) Location of each eye's centroid shown in green dots; (b) Distance between the two eyes' centroids shown in a blue line;



(c) Image rotation angle shown in a yellow circle; (d) Resulting face image obtained by the alignment algorithm (rotated to the computed angle); (e) Final output of the alignment algorithm (aligned and scaled into a default size).

GEOMETRIC FEATURE EXTRACTION

The Geometric Feature Extraction technique described in this section regards the jawline, mouth, eyes, eyebrows, and nose as the most relevant features during the extraction of facial expressions. These features are extracted from the preprocessed image with the help of the shape predictor function, a *Dlib* library function based on a set of pre-trained regression trees capable of estimating the location of 68 coordinate points needed for mapping facial structures [32]. A general overview of this technique is presented in Figure 4.

Fig. 4 General overview of the Geometric Feature Extraction technique: (a) Coordinate points captured by the geometric feature extractor; (b) Binary Mask created from the feature points; (c) Binary Mask with no jawline.

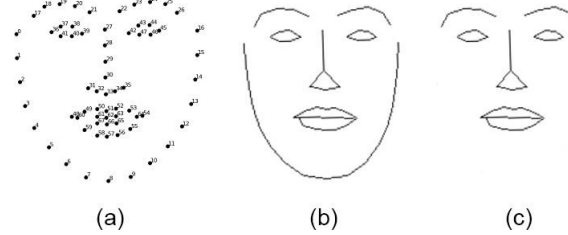
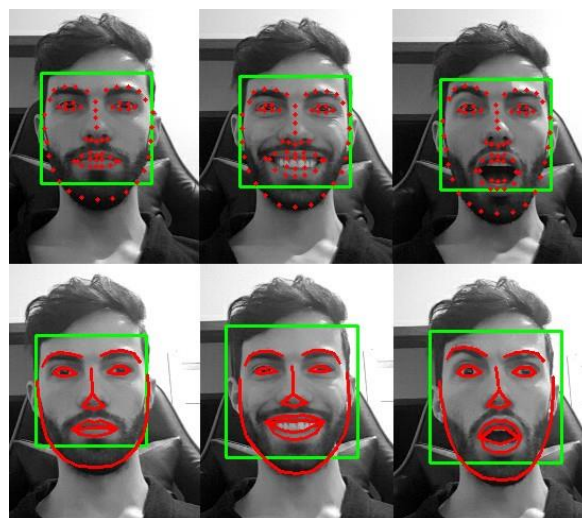


Figure 4a illustrates the 68 most representative points used by this approach. These points are then connected and transformed into a binary mask, Figure 4b, later used by the CNN classifier. This work also aims to compare the jawline's importance and impact during the classification process by evaluating a binary mask with no jawline, as depicted in Figure 4c.

From the image with the facial features, a new frame with a binary mask corresponding to the absolute white and black coloring is obtained by subtracting the pre-drawn image. As a result, for an image with one channel and 8-bit color depth, only values of 0 (black) and 255 (white) with no intermediate gray values can be obtained, simplifying the acquired set of features.

An application of the described Geometric Feature Extraction technique is demonstrated in Figure 5, where the 68 detected coordinate points and the eyes, nose, mouth, eyebrows, and jawline contours are highlighted. The features delimited by the red lines, drawn over the original image, are later processed with the image subtraction function to create the binary mask. Furthermore, the green square marks are used only for face recognition and are ignored when applying the subtraction function.

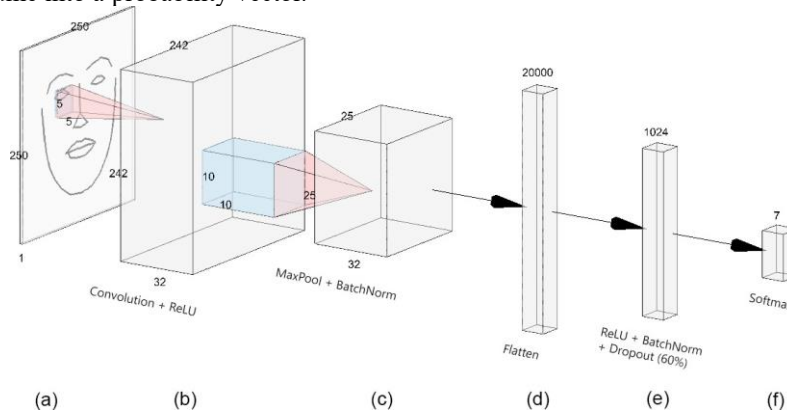
Fig. 5 Application of the Geometric Feature Extraction technique for three distinct expressions: neutral, happiness, and surprise, respectively.



CONVOLUTIONAL NEURAL NETWORK

The configuration and parameters of the proposed CNN architecture are illustrated in Figure 6. The input layer, as shown in Figure 6a, comprises a binary image of the most representative facial features and covers a planar domain of 250x250 pixels. Figure 6b depicts the Rectified Linear Unit activation function layer (ReLU) with network dimensions of 32x25x25 and activation of 10x10x25. Figure 6c displays the MaxPool+BatchNorm layer with network dimensions of 25x25x32. Figure 6d presents the Flatten layer, which reshapes the tensor to a single dimension and yields 20000 units. Finally, Figures 6e and 6f show the 60% dropout and decision output (*Softmax*) layers, which are discussed in detail in the subsequent subsection.

Fig. 6 Configuration of the proposed CCN architecture – The binary mask is inserted into a convolutional layer to extract the obtained features, followed by a max-pooling layer to reduce dimensionality. The fully connected layers then convert the feature volume into a probability vector.



Moreover, implementation details regarding the CNN used in the presented approach are also demonstrated by the source code in Figure 7, which describes the sequence used to build the CNN and the parameter values used as input configuration.

Fig. 7 Source code for the proposed CNN model, including its pipelining.

```
def cnn_modelo():
    modelo = Sequential()
    modelo.add(Conv2D(32, (5,5), input_shape=(250, 250, 1), activation='relu'))
    modelo.add(BatchNormalization())
    modelo.add(MaxPooling2D(pool_size=(10, 10), strides=(10, 10), padding='same'))
    modelo.add(Flatten())
    modelo.add(Dense(1024, activation='relu'))
    modelo.add(BatchNormalization())
    modelo.add(Dropout(0.6))
    modelo.add(Dense(7, activation='softmax'))
    sgd = optimizers.SGD(lr=1e-3)
    modelo.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
    callbacks_list = ModelCheckpoint("modelo/expressao.h5", monitor='val_accuracy',
    verbose=1, save_best_only=True, mode='auto')
    return modelo, callbacks_list
```

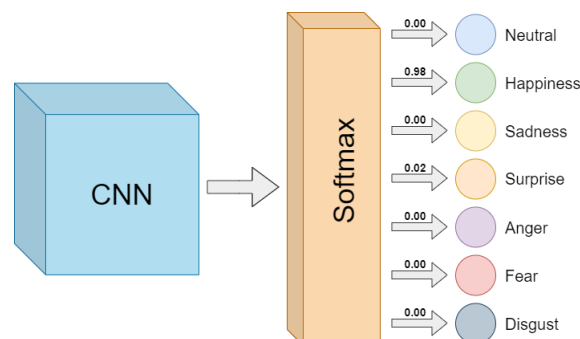
EXPRESSION CLASSIFIER

The final stage of the proposed methodology is related to the expression classifier layer. This layer follows the architecture introduced in Figure 6f, where the ReLU function activates the network and the first dense layer, followed by a 60% dropout and batch normalization procedure (BatchNorm). The final fully connected layer then employs the *Softmax* activation to build a probability distribution associated with one of the seven dataset emotions described in Section 3.1.

The reason for using the ReLU activation is based on self-acquired empirical evidence. This activation function produced the best results for the proposed CNN classifier compared to other activation functions. According to Ian Goodfellow et al. [33], the ReLU is the default recommendation regarding activation functions in modern neural network approaches, as it is appropriately designed to work with images and does not saturate in the positive region, tending to converge much faster than the sigmoid or hyperbolic tangent on images, as well as being not zero-centered.

The dropout stage is used to avoid overfitting since the individual nodes are removed from the network with a probability of 60%, maintaining a reduced network and forcing the nodes of a layer (neuron) to adapt and correct errors from previous layers, making the model more robust and reliable. Moreover, the BatchNorm procedure normalizes the activation of the previous layer, stabilizing and speeding up the neural network while improving the generalization error. Finally, the Softmax function calculates the probability of each input expression, and the sum of the probabilities for each of the seven expressions results in a value of 1. As illustrated in Figure 8, the expression with the highest probability value is the resulting emotion.

Fig. 8 The Softmax function results in the probability of a given facial expression. In this example, the CNN informs that the probability of the expression being happiness is 98%, although there is still a small chance (2%) to be the expression of surprise.



EXPERIMENTAL RESULTS

VALIDATION

Validating a CNN with the same data from the training set can be considered a logical procedural error. After all, a model that checks the images based on previous information would obtain good classification feedback, making the network extremely accurate. However, if the model's efficiency is measured based on data the network has never had access to, then in that case, the model will be completely limited (or useless in the worst case) in predicting new data in a broader set or during real-world applications.

This problem can be solved by conducting a supervised machine-learning phase where the dataset is split into two sets: (i) training and (ii) validation. This way, the network can use the first set to train its predictions and the second one to validate its accuracy, preventing the data used to calculate the accuracy from being seen by both sides (we call this invisible data).

Our approach split each dataset into 80% for training and 20% for validation during the initial supervised training stage. Subsequent training steps ran for 300 epochs using an SGD optimizer with a learning rate of 10^{-3} , a categorical cross-entropy as a loss function, and a batch size of 100. Table 2 shows the experimental results obtained by the presented approach in terms of accuracy compared to some related works found in the literature.

Table 2 Accuracy obtained by the presented method compared to other related works using the simple validation procedure (no cross-fold).

Method	Year	JAFFE	RaFD	CK+
BDBN	2014	93.00%		96.70%
Happy and Routray	2014	91.80%		94.09%
Hamester et al.	2015	95.80%		
DeXpression	2015			99.60%
Zavares et al.	2017		85.79%	88.58%
Mavani et al.	2017		95.71%	
MicroExpNet	2017			84.80%
TeacherExpNet	2017			97.60%
FAN	2019			99.69%
FMPN	2019			98.06%
Minaee and Abdolrashidi	2019	92.80%		98.00%
Kai Wang et al.	2020		88.14%	
Ben Niu et al.	2021	88.50%		93.20%
Our method		96.83%	98.58%	98.46%
Our method (no jawline)		96.03%	97.28%	98.57%

Although effective, this procedure can cause a false sense of overfitting correction that results from the validation set selection volatility and the assumption that the database is not homogeneous. Moreover, its effectiveness is related to the invisibility of the validation set towards the network trained by the training set. Therefore, the randomness applied to the dataset does not guarantee an increase in accuracy, and performing this procedure only once did not

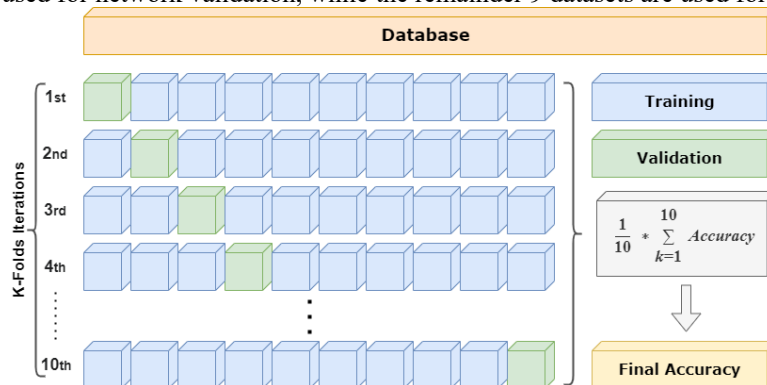
ensure sufficient precision to state whether the network analyzes invisible data efficiently.

CROSS-FOLD VALIDATION

To measure the accuracy of the network’s prediction ability, we used the cross-fold validation method [34], a simple and popular validation technique based on the number of groups (k) that a dataset needs to be divided to avoid biased results. Thus, each dataset was divided into ten subsets ($k = 10$), as this value usually results in predictions with modest variations and low polarization while obtaining approximately equal-sized subsets.

An example of the cross-fold validation method is illustrated in Figure 9, where the data is shuffled randomly and divided into ten subsets, and the model’s precision is obtained by averaging the accuracy values from each iteration, ensuring that the final result is a more accurate estimation in predicting the performance of real-world applications or with invisible data.

Fig. 9 Cross-fold validation method for $k = 10$ resulting in a random separation of 10 subsets of approximately equal sizes. The first subset is used for network validation, while the remainder 9 datasets are used for training.



The individual results of each iteration and the overall accuracy of the network for the two proposed methods (binary mask with and without jawline) are presented in Table 3. These results show that the variation between the two proposed methods is insufficient to state that the jawline provides valuable information for the proposed network in facial expression recognition.

Table 3 Comparison of the cross-fold validation results for the two proposed methods.

Method	JAFFE		RaFD		CK+	
	Normal	No-Jawline	Normal	No-Jawline	Normal	No-Jawline
Fold 1	96.54%	95.90%	97.88%	96.41%	97.93%	98.19%
Fold 2	97.06%	96.11%	98.21%	97.02%	97.74%	97.59%
Fold 3	96.92%	96.17%	98.07%	96.87%	96.91%	97.02%
Fold 4	96.14%	95.83%	98.20%	97.08%	98.49%	98.66%
Fold 5	95.36%	94.75%	98.36%	97.27%	98.58%	98.71%
Fold 6	97.05%	96.38%	97.93%	96.85%	98.07%	98.23%
Fold 7	97.22%	96.61%	98.27%	97.21%	96.95%	97.36%
Fold 8	95.49%	95.74%	97.97%	96.86%	98.39%	98.54%
Fold 9	95.17%	94.99%	98.14%	97.04%	97.43%	97.87%
Fold 10	96.61%	96.06%	97.93%	96.63%	97.82%	97.62%
Average	96.36%	95.85%	98.10%	96.92%	97.83%	97.98%

Similarly, a comparison of the final obtained results between the simple and cross-fold validation techniques is presented in Table 4. As expected, the cross-fold validation method reduced the accuracy of the network since it is a more conservative technique and computationally more expensive.

Table 4 Comparison of the obtained results using the simple and cross-fold validation techniques.

Validation Method	JAFFE	RaFD	CK+
Simple Normal	96.83%	98.58%	98.46%
No-Jawline	96.03%	97.28%	98.57%
Cross-fold Normal	96.36%	98.10%	97.83%
No-Jawline	95.85%	96.92%	97.98%

EXECUTION TIME AND PERFORMANCE

In this subsection, we discuss the method's performance in terms of execution time to provide an end-to-end solution regarding facial emotion prediction. We used the JAFFE dataset as input for the developed algorithm while performing a data augmentation technique to reach the data required in some performance tests, increasing the entries exponentially from 1 to 10000 images.

A comparison of the total time required for the end-to-end algorithm (including the image preprocessing step) and the prediction algorithm (ignoring the preprocessing step) is shown in Table 5. Furthermore, for visualization purposes, the total time of each iteration is also shown in FPS (frames per second) and PPS (prediction per second).

Table 5 Algorithm performance test based on processing time using an input data variation.

	Total processing time		Processing time per image		Metrics per second	
Images	End-to-End Prediction		End-to-End Prediction		FPS	PPS
e0	1s 077ms	1s 065ms	1s 77ms	1s 65ms	0.92	0.93
e1	1s 309ms	1s 189ms	130ms	118ms	7.63	8.40
e2	3s 737ms	2s 505ms	37ms	25ms	26.75	39.91
e3	27s 374ms	15s 291ms	27ms	15ms	36.53	65.39
e4	268s 808ms	143s 789ms	26ms	14ms	37.20	69.54

The setup configuration used for the performance tests was comprised of a desktop computer with a Windows 10 Pro operating system (version 20H2), an AMD Ryzen 7 3700X processor (8 cores with 3.59 GHz per core), 32 GB of RAM (3200 MHz), an NVIDIA GeForce RTX 2070 graphics card (8GB of dedicated video memory), and a Samsung 970 EVO Plus M.2 NVMe SSD.

CPU usage during image previews ranged from 7.8% to 9.1%, while GPU usage ranged from 0.3% to 1.1%. On the other hand, RAM usage was at most 2GB since the dedicated video memory was fully utilized. It is important to consider that these results were observed only for the process in which emotions were predicted, while the system and secondary processes were ignored.

These results show that a bottleneck is created during image loading and that the GPU processes in the batch execute faster than the CPU ones. A possible solution to this problem (not adopted in our approach) is based on distributing the image loading to a new thread to perform the preprocessing, causing the GPU to wait for more data during idle time and mitigating the higher CPU usage compared to GPU usage for the presented configuration.

Since the CPU loads more images at once (avoiding redundancy of operations), it makes the GPU access a greater amount of data, reducing idleness and consequently increasing the algorithm's performance. However, there is still a memory bottleneck for the GPU, being necessary to wait for the loading and unloading of new data from the dedicated memory of the GPU.

The results show that the proposed solution can be executed in real-time while continuously providing facial image decision-making with its corresponding emotion. In a continuous video feed provided by a webcam limited to 30fps, the algorithm was able to process 27 fps, where the fps loss could be related to the latency between the webcam input, loading data, and frame separation. This additional step is necessary to obtain the input image for the end-to-end network while performing an extra operation that was not necessary for the test performed with the JAFFE database, as the image was already captured, being only necessary

to load the image into the memory. The application of the proposed approach for a video sequence is demonstrated in Figure 10.

FINAL REMARKS

This work presented a real-time computational approach for facial image and video emotion recognition based on a geometric feature extraction technique associated with a CNN. The experimental results were compared against state-of-the-art methods, providing marginally better results for a mixed dataset. Accuracy rates using simple validation were 96.83%, 98.58%, and 98.57% for the JAFFE, RaFD, and CK+ datasets. On the other hand, through cross-validation, we obtained 96.36%, 98.10%, and 97.98% accuracy rates for the same datasets, indicating that the presented approach can be considered a promising solution for FER, especially for real-time applications.

Our method offers competitive results in the three datasets tested, achieving higher accuracies for the JAFFE and RaFD datasets. Using cross-fold validation, we can compare the accuracy between the normal and no-jawline method, whose conclusions are that using the jawline in the binary mask created can help the network to have an additional point of reference with the other facial landmarks. However, it does not directly influence the expression performed. Choosing to use or not use this landmark as input to the network.

Fig. 10 Facial emotions results for the proposed methodology: a) neutral, b) happiness, c) sadness, d) surprised, e) anger, f) fear, and g) disgust.





is a relatively insignificant decision in terms of precision. When comparing the results obtained with the simple and cross-fold validation, we see a decrease in accuracy in a range of approximately 1% since the cross-fold validation procedure tends to be more conservative with its percentages. Thus, it reinforces the thesis that the results obtained with this method are not a consequence of the overfitting effect.

In terms of execution time, the performance speed of the presented method is also encouraging, executing it in real-time with enough precision, paving the way for many real-world applications. Further works consider implementing some mechanism able to mitigate the time invariance of the system. We take each frame as its separate entity without regarding its relationship with the previous ones. This can cause some flickering effecting during the predictions, and taking into account temporal information related to the past few frames can be used to provide a soften classification and prediction.

DECLARATIONS

Conflict of interest: All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

DATA AVAILABILITY

The code generated during and/or analyzed during the current study is available in the GitHub repository <https://github.com/gustavogino/Facial-Expression-Recognition>. Furthermore, a demonstrative video of a real-time application can be found at <https://www.youtube.com/watch?v=fFOldbHtHQU>.

REFERENCES

1. Mellouk, W., Handouzi, W. (2020). Facial emotion recognition using deep learning: review and insights. *Procedia Computer Science*, 175, 689–694.
2. Ekman, P., Friesen, W.V. (1971). Constants across cultures in the face and emotion. *Journal of personality and social psychology*, 17(2), 124.
3. Orgeta, V. (2010). Effects of age and task difficulty on recognition of facial affect. *The journals of gerontology. Series B, Psychological sciences and social sciences*, 65B, 323–7. <https://doi.org/10.1093/geronb/gbq007>
4. Jung, H., Lee, S., Yim, J., Park, S., Kim, J. (2015). Joint fine-tuning in deep neural networks for facial expression recognition. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2983–2991.
5. Sun, N., Li, Q., Huan, R., Liu, J., Han, G. (2019). Deep spatial-temporal feature fusion for facial expression recognition in static images. *Pattern Recognition Letters*, 119, 49–61.
6. Dhall, A., Asthana, A., Goecke, R., Gedeon, T. (2011). Emotion recognition using PHOG and LPQ features. In: *Face and Gesture 2011*, pp. 878–883. IEEE.
7. Ojansivu, V., Heikkilä, J. (2008). Blur insensitive texture classification using local phase quantization. In: *International Conference on Image and Signal Processing*, pp. 236–243. Springer.
8. Zhang, Z. (1999). Feature-based facial expression recognition: Sensitivity analysis and experiments with a multilayer perceptron. *International Journal of Pattern Recognition and Artificial Intelligence*, 13(06), 893–911.
9. Barsoum, E., Zhang, C., Ferrer, C.C., Zhang, Z. (2016). Training deep networks for facial expression recognition with crowd-sourced label distribution. In: *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, pp. 279–283.
10. Chen, Y., Wang, J., Chen, S., Shi, Z., Cai, J. (2019). Facial motion prior networks for facial expression recognition. In: *2019 IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4. IEEE.
11. Wang, K., Peng, X., Yang, J., Lu, S., Qiao, Y. (2020). Suppressing uncertainties for large-scale facial expression recognition. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
12. Sakai, T., Nagao, M., Fujibayashi, S. (1969). Line extraction and pattern detection in a photograph. *Pattern recognition*, 1(3), 233–248.
13. Lyons, M., Kamachi, M., Gyoba, J. (1998). The Japanese Female Facial Expression (JAFPE) Dataset.
14. Lucey, P., Cohn, J.F., Kanade, T., Saragih, J., Ambadar, Z., Matthews, I. (2010). 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. IEEE.



15. Langner, O., Dotsch, R., Bijlstra, G., Wigboldus, D.H., Hawk, S.T., Van Knippenberg, A. (2010). Presentation and validation of the radboud faces database. *Cognition and emotion*, 24(8), 1377–1388.
16. Goodfellow, I.J., Erhan, D., Carrier, P.L., Courville, A., Mirza, M., Hamner, B., Cukierski, W., Tang, Y., Thaler, D., Lee, D.-H., et al. (2013). Challenges in representation learning: A report on three machine learning contests. In: *International Conference on Neural Information Processing*, pp. 117–124. Springer.
17. Happy, S., Routray, A. (2014). Automatic facial expression recognition using features of salient facial patches. *IEEE transactions on Affective Computing*, 6(1), 1–12.
18. Niu, B., Gao, Z., Guo, B. (2021). Facial expression recognition with LBP and ORB features. *Computational Intelligence and Neuroscience*, 2021.
19. Abdulrahman, M., Eleyan, A. (2015). Facial expression recognition using support vector machines. 2015 23rd Signal Processing and Communications Applications Conference, SIU 2015 - Proceedings, 276–279. <https://doi.org/10.1109/SIU.2015.7129813>
20. Salmam, F.Z., Madani, A., Kissi, M. (2016). Facial expression recognition using decision trees. *Proceedings - Computer Graphics, Imaging and Visualization: New Techniques and Trends, CGiV 2016*, 125–130. <https://doi.org/10.1109/CGiV.2016.33>
21. Pu, X., Fan, K., Chen, X., Ji, L., Zhou, Z. (2015). Facial expression recognition from image sequences using twofold random forest classifier. *Neurocomputing*, 168, 1173–1180. <https://doi.org/10.1016/J.NEUCOM.2015.05.005>
22. İlke Çıĝu, Şener, E., Akbaş, E. (2017). Microexpnet: An extremely small and fast model for expression recognition from face images. arXiv:1711.07011 [cs.CV]
23. Burkert, P., Trier, F., Afzal, M.Z., Dengel, A., Liwicki, M. (2015). Dexpression: Deep convolutional neural network for expression recognition. arXiv preprint arXiv:1509.05371.
24. Meng, D., Peng, X., Wang, K., Qiao, Y. (2019). Frame attention networks for facial expression recognition in videos. In: *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 3866–3870. IEEE.
25. Minaee, S., Abdolrashidi, A. (2019). Deep-emotion: Facial expression recognition using attentional convolutional network. arXiv preprint arXiv:1902.01019.
26. Van Rossum, G., Drake, F.L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.
27. Itseez. (2015). *Open Source Computer Vision Library*. GitHub. <https://github.com/itseez/opencv> Accessed 2021-06-26
28. Chollet, F., et al. (2015). *Keras*. GitHub. <https://github.com/fchollet/keras> Accessed 2021-06-26
29. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, Savannah, GA, pp. 265–283.



30. King, D.E. (2009). Dlib-ml: A Machine Learning Toolkit, 10, 1755–1758.
31. King, D. (2015). Face Detection with Python using OpenCV. Retrieved from http://dlib.net/face_detector.py.html. Accessed on: March 5, 2023.
32. Kazemi, V., Sullivan, J. (2014). One millisecond face alignment with an ensemble of regression trees. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1867–1874.
33. Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. MIT Press, Cambridge, Massachusetts. Retrieved from <http://www.deeplearningbook.org>
34. Refaeilzadeh, P., Tang, L., Liu, H. (2009). In: Liu, L., ÖZSU, M.T. (Eds.), Cross-Validation, pp. 532–538. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_565