Deep learning based face beauty prediction via dynamic robust losses and ensemble regression

Fares Bougourzi, F. Dornaika, Abdelmalik Taleb-Ahmed

Knowledge-Based Systems 242 (2022) 108246

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Deep learning based face beauty prediction via dynamic robust losses and ensemble regression

F. Bougourzi [b], F. Dornaika [a,c,d,*], A. Taleb-Ahmed [e]

[a] School of Computer and Information Engineering, Henan University, Kaifeng, China
[b] Institute of Applied Sciences and Intelligent Systems, National Research Council of Italy, 73100 Lecce, Italy
[c] University of the Basque Country UPV/EHU, San Sebastian, Spain
[d] IKERBASQUE, Basque Foundation for Science, Bilbao, Spain
[e] Univ. Polytechnique Hauts-de-France, Univ. Lille, CNRS, Centrale Lille, UMR 8520 - IEMN, F-59313 Valenciennes, France

ARTICLE INFO

Article history:
Received 12 October 2021
Received in revised form 20 December 2021
Accepted 19 January 2022
Available online 31 January 2022

Keywords:
Facial beauty prediction
Convolutional neural network
Deep learning
Ensemble regression
Robust loss functions

ABSTRACT

In the last decade, several studies have shown that facial attractiveness can be learned by machines. In this paper, we address Facial Beauty Prediction from static images. The paper contains three main contributions. First, we propose a two-branch architecture (REX-INCEP) based on merging the architecture of two already trained networks to deal with the complicated high-level features associated with the FBP problem. Second, we introduce the use of a dynamic law to control the behaviour of the following robust loss functions during training: ParamSmoothL1, Huber and Tukey. Third, we propose an ensemble regression based on Convolutional Neural Networks (CNNs). In this ensemble, we use both the basic networks and our proposed network (REX-INCEP). The proposed individual CNN regressors are trained with different loss functions, namely MSE, dynamic ParamSmoothL1, dynamic Huber and dynamic Tukey. Our approach is evaluated on the SCUT-FBP5500 database using the two evaluation scenarios provided by the database creators: 60%–40% split and five-fold cross-validation. In both evaluation scenarios, our approach outperforms the state of the art on several metrics. These comparisons highlight the effectiveness of the proposed solutions for FBP. They also show that the proposed dynamic robust losses lead to more flexible and accurate estimators.

1. Introduction

For centuries philosophers, artists, and scientists have tried to discover the mystery of beauty [1]. In fact, the beauty of the face is gaining more and more interest due to the rapid development of plastic surgery and cosmetic industry [2]. In recent years, facial beauty estimation and classification has become an interesting research topic in computer vision and machine learning due to its growing applications [3,4]. Applications for facial beauty estimation and prediction include: Cosmetic recommendations [5], scheduling of aesthetic surgeries [6], facial beautification [7], and Social Networks Services (SNS) (such as Facebook, Instagram, and dating websites) [8]. In addition, automatic facial beauty prediction (FBP) may find application when attractiveness is a basic requirement, such as in advertising, magazine covers, and screening applicants for certain jobs, such as entertainment and modelling [6].

Despite the considerable progress in estimating and predicting the beauty of faces, more labelled data is needed for training deep CNNs. To deal with the data limitation, some researchers used active data augmentation. In addition, others used pre-trained models trained on the ImageNet database [9]. These pre-trained models are capable of extracting high-level features. In this paper, we propose a system that exploits the diversity of learners. We present two main proposals. First, we propose to combine two different CNN architectures into a single architecture (called the two-branch architecture) that is trained end-to-end. Second, we propose to build an ensemble of regressions where the final prediction is given by the average of all predictions. The latter solution does not need to be trained on new validation sets. More specifically, we propose ensemble regressions using one-branch architectures (ResneXt-50 and Inception-v3) and our proposed two-branch architecture (REX-INCEP) trained with different loss functions. Four loss functions are used in our approach, namely MSE, dynamic ParamSmoothL1, dynamic Huber and dynamic Tukey. In summary, the main contributions of this paper are as follows:

- We propose ParamSmoothL1 regression loss function ParamSmoothL1. Moreover, we introduce a dynamic law that changes the parameters of the robust loss function during training. For this purpose, we use the cosine law

* Corresponding author at: University of the Basque Country UPV/EHU, San Sebastian, Spain.
E-mail address: fadi.dornaika@ehu.eus (F. Dornaika).

https://doi.org/10.1016/j.knosys.2022.108246

with the following robust loss functions: ParamSmoothL1, Huber and Tukey. This can solve the problem of complexity in finding the best loss function parameter.

- We propose two branches network (REX-INCEP) for face beauty estimation based on ResneXt-50 and Inception-v3 architectures. The main advantage of our REX-INCEP architecture is its ability to learn FBP features at a high level by using ResneXt and Inception blocks simultaneously, which proved its efficiency compared to seven CNN architectures. Moreover, our REX-INCEP architecture provides the right trade-off between the performance and the number of parameters for facial beauty prediction.
- We propose an ensemble regression for facial beauty estimation by merging the predicted scores of networks with one branch networks (ResneXt-50 and Inception-v3) and two branches network (REX-INCEP) trained with four loss functions (MSE, dynamic ParamSmoothL1, dynamic Huber, and dynamic Tukey). Although the individual regression models are trained with the same fixed hyperparameters, the resulting ensemble regression provides the most accurate estimates compared to the individual models and to state-of-the-art solutions. We have made our codes and pre-processed faces publicly available using our face alignment scheme at https://github.com/faresbougourzi/CNN-ER_for_FBP. (Last accessed on November, 29th 2021)

This paper is organized as follows: Section 2 presents some related work on facial beauty prediction. In Section 3, we illustrate the backbone CNN architectures used, the proposed approach, and the proposed dynamic robust losses. Section 4 includes: the description of the database and evaluation metrics used, the experimental settings, and the 60%–40% split and five fold cross-validation experiments. Section 4 is concluded by the comparison with state-of-the-art methods. Finally, Section 5 concludes the paper.

2. Related work

Image-based estimation of the beauty of faces is a new problem in computer vision. The first database that treats FBP as a regression task is from 2015 [10]. The methods used in the literature to predict and estimate facial beauty are either hand-crafted methods [11–18] or deep learning methods [12,19–21]. The hand-crafted methods can be classified as geometry-based or appearance-based methods [16]. In [15], P. Aarabi et al. developed an automatic facial beauty rating system based on the relationships between facial features (face, eyes, eyebrows and mouth) with the K-nearest neighbour algorithm to learn a beauty mapping. D. Zhang et al. used tens of thousands of female and male faces and assigned them to a human face shape subspace. They then used a quantitative method to analyse the effect of geometric facial features on human facial beauty using a similarity transformation invariant shape distance measure. In [16], H. Yan proposed a new CSOR (Cost-Sensitive Ordinal Regression) to measure the importance of samples in different classes. They applied their CSOR to four types of features, namely intensity, LBP [22], SIFT [23], and LE [24]. L. Liang et al. [12] used geometric features (extracted 18-dimensional ratio features from faces) and appearance features (40 Gabor feature maps) with shallow predictors, which are linear regression (LR), Gaussian regression (GR), and support vector regression (SVR). These methods were tested using the SCUT-FBP5500 database.

In recent years, deep learning architectures have been widely used to evaluate the beauty of faces. In [12], L. Liang et al. presented their face beauty database (SCUT-FBP5500) with two evaluation protocols (60%–40% split and 5-fold cross validation). They tested three CNN architectures (Alexnet [25], Resnet-18 [26] and ResneXt-50 [27]). Their results show that the ResneXt-50 architecture outperformed the other two deep architectures (Alexnet and Resnet-18). Moreover, the deep architectures performed better than the hand-crafted features they used with different shallow regressors. K. Cao et al. used a residual-in-residual (RIR) block to build a deeper network with multi-level skip connections to produce better gradient transmission flow. In addition, they used both channel-wise and space-wise attention mechanisms to find the inherent correlation between feature maps. Their approach was tested on the SCUT -FBP5500 [12] database and showed good performance. In [21], L. Lin et al. propose an R^3CNN architecture consisting of two main components. The first component is a regression component that contains two identical regression subnets that consistently map each face image to a beauty value. The second component is a ranking component that uses the Siamese network to learn a pairwise ranking to guide the regression of the beauty prediction. Their architecture showed promising results on SCUT-FBP [10] and the SCUT-FBP5500 [12] databases. In addition to supervised learning, semi-supervised learning shows promising results for face beauty estimation [28,29]. In [29], F. Dornaika et al. presented a multi-layered local discriminant embedding algorithm that integrates feature selection as the main step. Feature selection captures the most relevant and discriminative features of an input face image or face descriptor.

3. Methodology

In this section, we will present the used CNN architectures and our proposed approach and the proposed dynamic robust losses.

3.1. Backbone CNN architectures

Since deep learning architecture "Alexnet" [25] won the ImageNet challenge in 2012, numerous CNN architectures have been proposed. In our work, we will use two popular architectures (ResneXt-50 [27] and Inception-v3 [30]) as building blocks for our solution. It is worth noting that other backbone architectures can also be used. In our proposal, we use the above pre-trained models trained on the ImageNet challenge database [9]. To keep the paper self-contained, this section briefly introduces the CNNs ResneXt-50 and Inception-v3, which were used as backbone architectures in our proposed solution.

ResneXt-50 architecture. The architecture of ResneXt-50 is presented in [27], which is based on the ResneXt module (Fig. 1). The ResneXt module performs a series of transformations, each based on a low-dimensional embedding and sharing the same topology. The results of all transformations are combined by summation.

Inception-v3 architecture. The Inception-v3 architecture is presented in [30], which is based on the Inception module presented in [31]. The main idea of the inception architecture is to combine different convolutional layers with different kernel sizes and pooling layers in one inception module, as shown in Fig. 2.

3.2. Our approach

Our global approach is shown in Fig. 3. The output score is the average of multiple scores, which means that we use an ensemble of multiple regression models. In our implementation, we use six models. There are two main contributions in this ensemble: (i) the deep network with two branches (REX-INCEP) (see Section 3.4) and (ii) the dynamic robust loss functions (see Section 3.5).

The first two scores are predicted by the trained deep networks ResneXt-50 and Inception-v3 using the MSE loss function

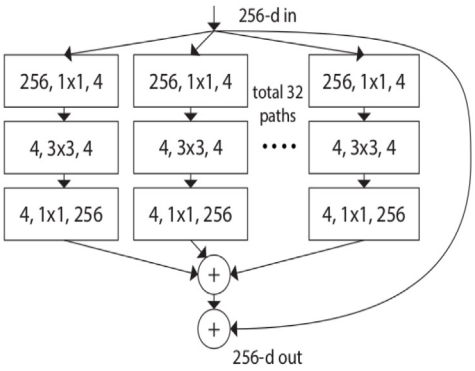

Fig. 1. A ResneXt Module with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels) [27].

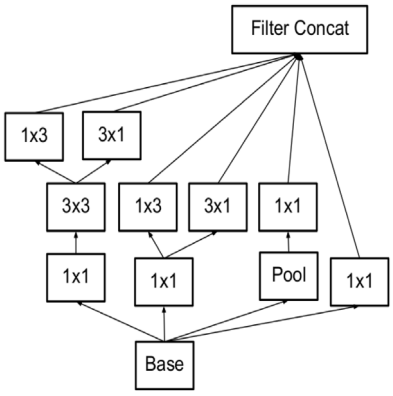

Fig. 2. The Third Inception Module used in Inception-v3 architecture [30].

and the dynamic Huber loss function, respectively. The remaining four scores are estimated after training the proposed two-branch deep network (REX-INCEP) using four loss functions: MSE, dynamic ParamSmoothL1, dynamic Huber, and dynamic Tukey. The two-branch deep network consists of ResneXt-50 and inception-v3, which are merged into a single architecture. As will be seen in the experimental section, using the two contributions without the ensemble results in performance that is better than that of the state-of-the-art methods. Using the ensemble shown in Fig. 3 will further improve the results.

3.3. Face preprocessing

In the face preprocessing phase, we used the scheme proposed in [32,33]. The process is shown in Fig. 4. There are three steps to crop the face region from the raw face image. First, we use the provided face features to align the eyes by performing a 2D rotation of the face based on the eye coordinates. After this 2D rotation of the image and the detected points, the three farthest points in the left, right and bottom directions are selected as the three boundaries of the face. We denote the distance from the lower boundary to the position of the eyes as d_1. The upper boundary of the face is set with a distance d_2 from the eyes, which is set as $d_2 = 0.6\ d_1$. Finally, obtain the face ROI, by cropping the face using the four boundaries and resizing the obtained box image to a fixed size that depends on the input size of the corresponding network.

3.4. Two branches architecture

In recent years, many successful deep architectures have been proposed for many computer vision tasks.

To train two architectures simultaneously, we propose two branches architecture to exploit the different capabilities of the networks. Since FBP data is limited, we propose to exploit the low-level and high-level feature extraction capability of two powerful architectures simultaneously. Fig. 5 summarizes our proposed architecture with two branches. The first and second branches are the ResneXt-50 and Inception-v3 architectures, respectively, with the decision levels removed. In our proposed architecture with two branches, we added the layer FC1 that maps the output of the ResneXt-50 branch (vector of dimension 2048) to 1024 neurons. Similarly, we added layer FC2, which maps the output of the Inception-v3 branch (vector of dimension 2048) to 1024 neurons. FC1 and FC2 were concatenated into a single vector FC, which is followed by the FC2 layer that performs the regression. Note that the weights of both branches are the weights of the pre-trained ResneXt-50 and Inception-v3 models (trained on the ImageNet challenge database [9].), while the FC1, FC2 and FC3 layers are randomly initialized. Our proposed network with two branches is called REX-INCEP architecture. In the training phase, we will fine-tune this architecture for FBP.

3.5. Loss functions: the use of dynamic robust losses

During convolutional network training, the loss function measures the error (the loss) between the ground truth and the estimated values. The CNNs aim to minimize the loss based on the gradients of the loss function used to update the weights of the network. In this section, we will describe the loss functions used in our experiments. We emphasize that three of them are robust loss functions. We will also introduce a dynamic law that adjusts the parameters of the robust losses during training. The losses are computed for the batch size N, y_i denotes the ground truth score of the ith image, and $\hat{y}_i$ denotes the estimated value corresponding to the ith image.

3.5.1. L_1 Loss function

L_1 is one of the most commonly used loss functions. The most important property of the L_1 loss function is its robustness to outliers. For N batch size, L_1 loss function is defined by:

$$L_{L_1} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{1}$$

3.5.2. Mean Squared Error (MSE) loss function

MSE is also known as L_2 loss function, it is more sensitive to outliers compared to L_1. The MSE loss function should be used when the target data are normally distributed around a mean and when it is important to penalize outliers particularly heavily. For N predictions, the MSE loss function is defined by:

$$L_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2}$$

3.5.3. Dynamic parameterized SmoothL1 (ParamSmoothL1) loss function

The loss function SmoothL1 creates a criterion that uses a quadratic term if the absolute element-wise error falls below 1, and an L_1 term otherwise. It is less sensitive to outliers than the MSE loss function, and in some cases prevents exploding gradients [34]. The SmoothL1 loss function of N images is defined by:

$$L_{SmoothL1} = \frac{1}{N}\sum_{i=1}^{N} z_i \tag{3}$$

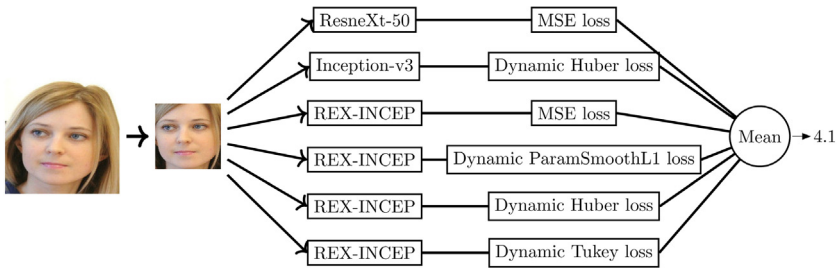

Fig. 3. General structure of the proposed approach (CNN-ER).

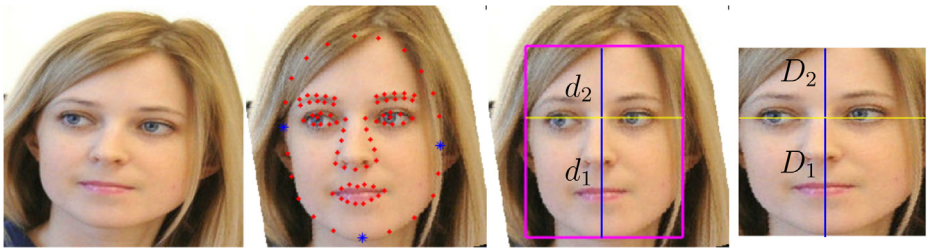

Fig. 4. Face Region of Interest. The left image is an original image from the database SCUT-FBP5500 [12]. The second image is the rotated face with its 86 detected landmarks used to estimate the three face boundaries (right, left and bottom). These boundaries correspond to the three points $*$ marked in blue. The third image shows how the upper boundary of the face is determined. It is located at a distance $d_2 = 0.6\ d_1$ from the vertical position of the eyes. The fourth image shows the cropped and rescaled face image with 224 pixels. Note that the distances D_1 and D_2 are constant for all cropped faces.

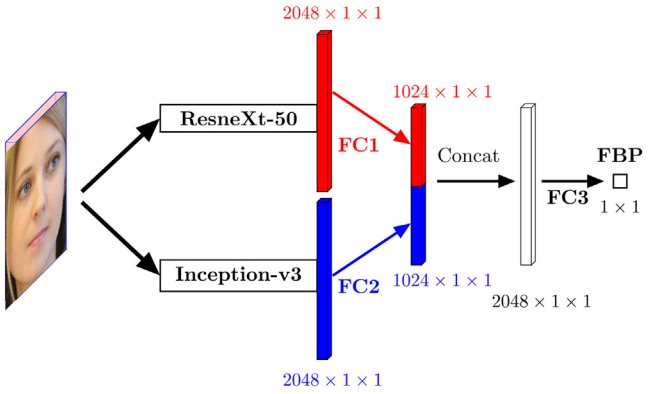

Fig. 5. Our proposed two branches network REX-INCEP.

where N is the batch size and z_i is given by:

$$z_i = \begin{cases} 0.5\,(y_i - \hat{y}_i)^2, & if\ |y_i - \hat{y}_i| < 1 \\ |y_i - \hat{y}_i| - 0.5, & \text{otherwise} \end{cases} \tag{4}$$

Since the threshold may vary from one task to another, we proposed a Parameterized SmoothL1 loss function defined as follows:

$$L_{Para\_SmoothL1} = \frac{1}{N}\sum_{i=1}^{N} z_i \tag{5}$$

where N is the batch size and z_i is given by:

$$z_i = \begin{cases} 0.5\,(y_i - \hat{y}_i)^2, & if\ |y_i - \hat{y}_i| \le \alpha \\ |y_i - \hat{y}_i| + 0.5\,\alpha^2 - \alpha, & \text{otherwise} \end{cases} \tag{6}$$

where α is tunable parameter. Our proposed dynamic robust loss functions are based on the following observation. During the training of ConvNets, the robust loss functions can be adjusted as the training progresses. Namely, during training, the model evolves and the outlier examples may vary. In the early stages of training, the model is usually neither very stable nor accurate enough to handle the outlier examples. Therefore, it is recommended to use the quadratic function of loss. At the end of the training, the model may be more or less accurate to deal with the outliers. Therefore, it is recommended to use the robust loss function where the range of non-outlier errors is relatively small. This means that the parameter of the robust loss function starts with a maximum value and decreases monotonically as the training progresses. From a practical point of view, it is extremely difficult to know the best value for α in advance. However, the variation interval $[\alpha_{min}, \alpha_{max}]$ can be known in advance. Therefore, to make the robust loss function more adaptive to the training progress, we propose a dynamic parameter α. This parameter follows a cosine law as a function of the epoch number. The current value of α is given by:

$$\alpha_{cur} = \alpha_{min} + \frac{1}{2}(\alpha_{max} - \alpha_{min})\left(1 + \cos\left(\frac{e_{cur}}{n_e}\,\pi\right)\right) \tag{7}$$

where α_{cur} is the value of α at the current epoch (e_{cur}). The latter varies between 1 and the total number of epochs (n_e). α_{max} and α_{min} are the maximum and minimum of the α value. In this paper, we denote the proposed dynamic Parameterized SmoothL1 by dynamic ParamSmoothL1. Fig. 6 shows the values of α using the proposed law (Eq. (7)) as a function of epoch number. Here α_{max} and α_{min} are fixed at 0.7 and 0.3, respectively. Our dynamic law was inspired by the dynamic law used to control the learning rate in stochastic gradient descent methods [35].

3.5.4. Dynamic Huber loss function

Similar to ParamSmoothL1, Huber is another loss function that is less sensitive to outliers in the data than the L_2loss function L_2. For N training images, the Huber loss function is defined by [36]:

$$L_{Huber} = \frac{1}{N}\sum_{i=1}^{N} z_i \tag{8}$$

where N is the batch size and z_i is defined by:

$$z_i = \begin{cases} 0.5\,(y_i - \hat{y}_i)^2, & if\ |y_i - \hat{y}_i| \le \beta \\ \beta\,|y_i - \hat{y}_i| - 0.5\,\beta^2, & \text{otherwise} \end{cases} \tag{9}$$

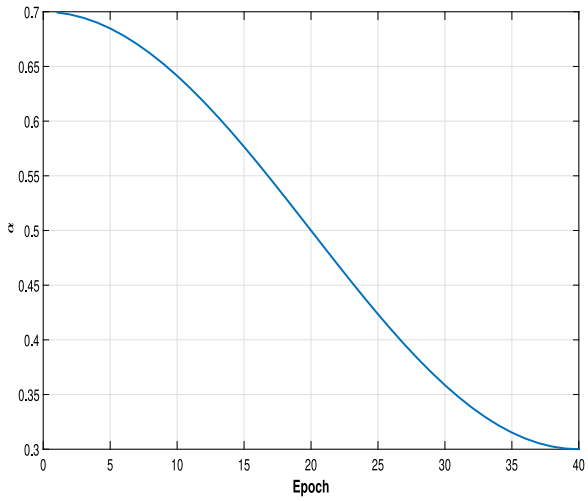

Fig. 6. Dynamic ParamSmoothL1 with α that decreases from 0.7 to 0.3.

where β is a controlled parameter. Fig. 7, shows a visualization of the Huber loss function with four β values (0.7, 0.5, 0.3 and 0.1) and L_2 loss function.

Similar to the ParamSmoothL1 loss, we propose to use the dynamic β given by the following the equation:

$$\beta_{cur} = \beta_{min} + \frac{1}{2}(\beta_{max} - \beta_{min})\left(1 + \cos\left(\frac{e_{cur}}{n_e}\pi\right)\right) \tag{10}$$

where β_{cur} is the value of β in the current epoch (e_{cur}), where e_{cur} increases from 1 to the total number of epochs (n_e). β_{max} and β_{min} are the maximum and minimum of the β value.

3.5.5. Dynamic Tukey loss function

The Tukey loss function [37] has the property of suppressing the influence of outliers during backpropagation by reducing the magnitude of their gradient towards zero. Another interesting property of this loss function is the soft constraints it imposes between inliers and outliers [38]. The Tukey loss function is defined by:

$$L_{Tukey} = \frac{1}{N}\sum_{i=1}^{N} z_i \tag{11}$$

where N is the batch size and z_i is given by:

$$z_i = \begin{cases} \frac{c^2}{6}\left[1 - \left(1 - \left(\frac{|y_i - \hat{y}_i|}{c}\right)^2\right)^3\right], & if\ |y_i - \hat{y}_i| \le c \\ \frac{c^2}{6}, & \text{otherwise} \end{cases} \tag{12}$$

where c is an adjustable parameter. Similar to ParamSmoothL1 and Huber losses, we propose to use dynamic c during training through the equation:

$$c_{cur} = c_{min} + \frac{1}{2}(c_{max} - c_{min})\left(1 + \cos\left(\frac{e_{cur}}{n_e}\pi\right)\right) \tag{13}$$

where c_{cur} is the value of c at the current epoch (e_{cur}), where e_{cur} increases from 1 to the total number of epochs (n_e). c_{max} and c_{min} are the maximum and minimum of the c value.

4. Performance evaluation

4.1. Database and evaluation protocols

To evaluate the performance of our approach, we used the database SCUT-FBP5500 [12]. It consists of 5500 frontal faces of subjects with different attributes: Age (from 15 to 60), gender (male/female), and ethnicity (Asian/Caucasian). Each face image was given a beauty score in the range [1–5] by 60 volunteers. In addition, each face image contains 86 facial features. Figs. 8(a), 8(b), 8(d) and 8(c) show some face samples with their corresponding beauty ratings. The creators of the SCUT-FBP5500 database provided two evaluation scenarios [12]. In the first scenario, the data were split into a training split and a test split (60%–40%). In the second scenario, the data was split into 5 folds to perform a five-fold cross-validation. In our analyses, we will use both scenarios.

4.2. Evaluation metrics

To evaluate the performance of each model, four evaluation metrics are used, namely: mean absolute error (MAE), root mean square error (RMSE), Pearson correlation coefficient (PC) and the ϵ error. Consider $Y = (y_1, y_2, \ldots, y_n)$ the ground-truth scores of the tested n images and $\hat{Y} = (\hat{y_1}, \hat{y_2}, \ldots, \hat{y_n})$ as their corresponding estimated scores. Where n represents the number of face images tested. The evaluation metrics are defined as follows:

Mean Absolute Error (MAE): MAE is defined by:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{14}$$

MAE is a scale-dependent accuracy measurement, i.e. MAE uses the same scale as the data being measured.

Root Mean Square Error (RMSE): RMSE is defined by:

$$\text{RMSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{15}$$

The RMSE is another scale-dependent accuracy measure. Unlike MAE, the effect of any error on the RMSE is proportional to the squared error; therefore, larger errors have a disproportionately large effect on the final RMSE. Consequently, the RMSE is sensitive to outliers.

Pearson Correlation coefficient (PC): PC was developed by Karl Pearson [39] and it is defined by:

$$\text{PC} = \frac{\sum_{i=1}^{n}(y_i - \overline{y_i})(\hat{y}_i - \overline{\hat{y}_i})}{\sqrt{\sum_{i=1}^{n}(y_i - \overline{y_i})^2}\sqrt{\sum_{i=1}^{n}(\hat{y}_i - \overline{\hat{y}_i})^2}} \tag{16}$$

where $\overline{y_i}$ and $\overline{\hat{y}_i}$ are the means of the ground-truth scores and the estimated scores, respectively. PC has a value between +1 and -1, it is a statistic that measures the linear correlation between two variables Y and $\hat{Y}$. A value of +1 means a completely positive linear correlation, 0 means no linear correlation, and −1 means a completely negative linear correlation.

ϵ-error: ϵ-error is defined by:

$$\epsilon\text{-error} = \frac{1}{n}\sum_{i=1}^{n}\left(1 - exp\left(\frac{(y_i - \hat{y}_i)^2}{2\,\sigma_i^2}\right)\right) \tag{17}$$

where σ_i is the standard deviation of the scores of all raters of image i. The value of the ϵ error is the accumulation of the errors of the individual images i based on the term $\epsilon\text{-error}_i = 1 - exp\left(\frac{(y_i - \hat{y}_i)^2}{2\,\sigma_i^2}\right)$. When the absolute error of image i approaches zero (i.e., $y_i = \hat{y}_i$), $\epsilon\text{-error}_i$ is zero. On the other hand, when the absolute error is large, ϵ-error takes into account the uncertainty of the rate represented by σ_i^2. More precisely, the division by the term σ_i^2 contributes less to the value of the ϵ error when the uncertainty of the rate is large and vice versa.

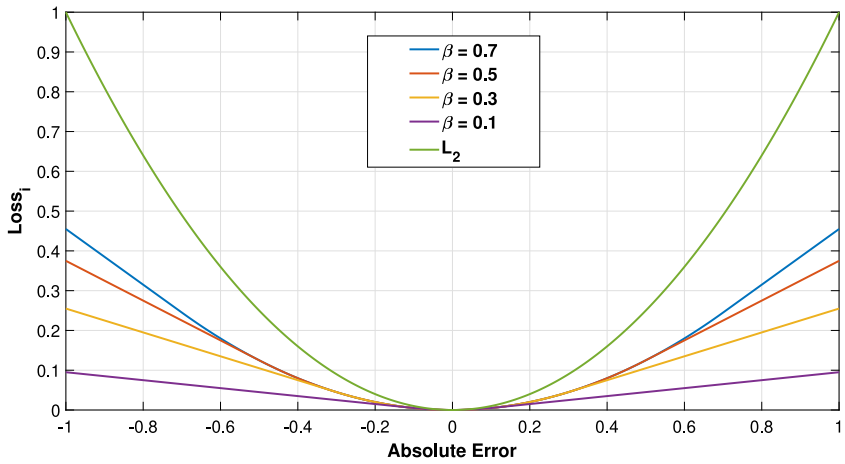

Fig. 7. Visualization of two loss functions: L_2 and Huber with four β values (0.7, 0.5, 0.3 and 0.1).

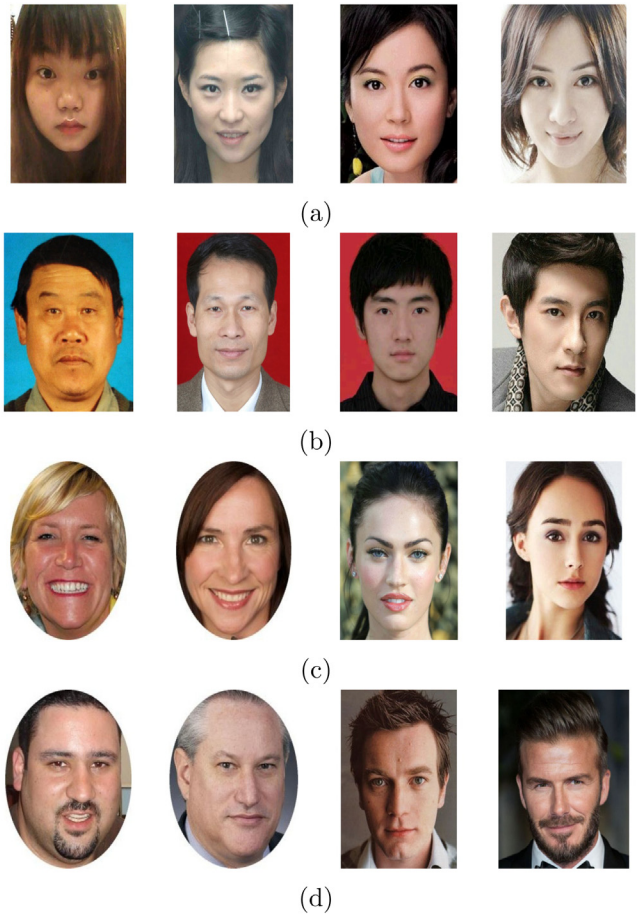

Fig. 8. Facial beauty samples from the SCUT-FBP5500 database, (a) Female Asian samples and the corresponding scores are (from left to right): 1.88, 3.00, 3.93, and 4.28. (b) Male Asian samples and the corresponding scores are (from left to right): 1.73, 2.48, 3.53, and 4.43. (c) Female Caucasian samples and the corresponding scores are (from left to right): 1.93, 2.87, 3.63, and 4.7. (d) Male Caucasian samples and the corresponding scores are (from left to right): 1.88, 2.67, 3.27, and 4.43.

4.3. Experimental setup

All experiments were performed on Pytorch [40] with an NVIDIA Geforce GTX 1060 6 GB GPU. All networks are trained for 40 epochs using Adam optimizer [41] and batch size of 15. The initial learning rate is 1e−4 for 20 epochs, then the learning rate decreases to 1e−5 for the next 10 epochs, and for the last 10 epochs the learning rate decreases to 1e−6. Active data augmentation is performed by rotating the input face by an angle between [−5, 5]. For all experiments, the reported results correspond to the best PC of the test data during the training/testing of the 40 epochs.

4.4. Experimental results on the 60%–40% split scenario

In this section, we limit the study to the provided 60%–40% split.

4.4.1. Raw input vs the proposed face preprocessing

To investigate the effectiveness of our proposed face preprocessing method, we used ResneXt-50 and Inception-v3 with loss function MSE to train FBP on two input image scenarios (the raw image and the cropped face with our preprocessing method). The obtained results are summarized in Table 1. From these results, it can be seen that our preprocessing scheme improves the results for both ResneXt-50 and Inception-v3 architectures. In other words, our proposed face alignment scheme can support the training of CNN architectures by discarding the background features and prioritizing the face features.

4.4.2. FBP using CNN architectures

In this section, we compare the performance of seven CNN architectures (VGG-16 [42], Resnet-50 [44], Resnet-101 [44], Resnet-152 [44], Wide-Resnet [43], Inception-v3 [30], ResneXt-50 [27]) using the standard MSE loss function. The results are summarized in Table 2. Based on these results, we can conclude that Inception-v3 and ResneXt-50 perform the best compared to the other CNN architectures. Moreover, these two CNN architectures have a smaller number of parameters than VGG-16, Resnet-101, Resnet-152, Wide-Resnet and similar to Resnet-50, which proves the ability of Inception-v3 and ResneXt-50 to learn high-level features for FBP with an intermediate number of parameters.

Given the observed efficiency of the Inception-v3 and ResneXt-50 architectures in both performance and number of trainable parameters (Table 2), we propose to combine these two CNN architectures (REX-INCEP). From Table 2, we conclude that our proposed REX-INCEP achieves better performance than all CNN architectures. Moreover, the number of trainable parameters of our proposed REX-INCEP is similar to Resnet-101, Resnet-152 and Wide-Resnet and less than VGG-16. These advantages prove the

Table 1
Face beauty prediction using ResneXt-50 and Inception-v3 networks with MSE loss function and two input image scenarios (The raw image and the detected face with our preprocessing scheme).

CNN architecture	Pre-processing	PC ↑	MAE ↓	RMSE ↓	ϵ-error ↓
ResneXt-50	Raw image	0.9119	0.2126	0.2845	0.0853
ResneXt-50	Face detection	0.9146	0.2092	0.2763	0.0802
Inception-v3	Raw image	0.9108	0.2150	0.2831	0.0873
Inception-v3	Face detection	0.9112	0.2147	0.2814	0.0833

Table 2
Comparison between seven CNN architectures (VGG-16, Resnet-50, Resnet-101, Resnet-152, Wide-Resnet, Inception-v3 and ResneXt-50) and our proposed REX-INCEP approach for Facial Beauty Prediction with MSE loss function.

CNN architecture	PC ↑	MAE ↓	RMSE ↓	ϵ-error ↓	N of params
VGG-16 [42]	0.9025	0.2229	0.2932	0.0891	134 M
Wide-Resnet [43]	0.9066	0.2176	0.2889	0.0851	66 M
Resnet-50 [44]	0.9087	0.2155	0.2850	0.0849	23 M
Resnet-152 [44]	0.9069	0.2182	0.2880	0.0860	58 M
Resnet-101 [44]	0.9095	0.2157	0.2852	0.0848	44 M
Inception-v3 [30]	0.9139	0.2125	0.2779	0.0819	25 M
ResneXt-50 [27]	0.9146	0.2092	0.2763	0.0802	22 M
REX-INCEP (Our architecture)	**0.9159**	**0.2071**	**0.2739**	**0.0790**	52 M

efficiency of our proposed REX-INCEP for FBP compared to CNN architectures.

As shown in Fig. 5, our proposed REX-INCEP has two branches. In the first branch, our proposed REX-INCEP architecture is able to learn high-level features for FBP by using a combination of splitting, transformation and aggregation mechanisms through the ResneXt block. In the second branch, our proposed REX-INCEP architecture is able to learn high-level features for FBP by combining different convolutional layers with different kernel sizes and pooling layers through the Inception blocks. The main advantage of our REX-INCEP architecture is its ability to learn high-level FBP features using ResneXt and Inception blocks simultaneously, which proved its efficiency compared to seven CNN architectures. From the results in Table 2, we conclude that our REX-INCEP architecture provides the right tradeoff between the performance and the number of parameters for facial beauty prediction.

4.4.3. Dynamic vs fixed loss parameter

In this section, we compare the performance of Face Beauty Prediction with dynamic and fixed loss functions. In this set of experiments, we choose a CNN architecture and a parametric robust loss function. We then compare the performance of two variants of this parametric robust loss function: (i) a loss function that assumes a fixed parameter, and (ii) a loss function that assumes a dynamic parameter using the cosine law. Specifically, we use the ResneXt-50 network and the following parametric loss functions: ParamSmoothL1, Huber and Tukey loss functions. To provide a fair comparison, the interval of parameter variation associated with the dynamic scheme is also used by the fixed parameter loss function. This is achieved by repeating the training and testing with several fixed values from the same interval.

We compare the performance obtained with the dynamic scheme with the average performance associated with the spanned fixed values within this interval. Table 3 summarizes the results obtained. For the loss of ParamSmoothL1, the interval of α is fixed to [0.7–0.3]. The fixed parameter scheme spans the following values {0.7, 0.6, 0.5, 0.4, 0.3}. Based on the results of the loss function ParamSmoothL1, we can see that the performance of the dynamic scheme is better than the average performance obtained with the fixed loss. Similar to ParamSmoothL1, the dynamic Huber loss function with a β parameter in the interval [0.7–0.3] achieved better performance than the mean performance obtained by the fixed β values {0.7, 0.6, 0.5, 0.4, 0.3}. For Tukey loss, the interval of parameter c was set to [2-1.5] and the fixed c values are {2, 1.7, 1.5}. Similar to ParamSmoothL1 and the Huber loss function, the dynamic Tukey loss provided better performance than the Tukey loss with a fixed c value. Moreover, the dynamic Tukey loss function adopting the [2-1] achieved better performance than the dynamic Tukey loss function adopting the interval [2-1.5]. In our approach, the intervals for the α, β and c parameters of the three loss functions are [0.7–0.3] [0.7–0.3] and [2-1], respectively.

4.4.4. Two branches vs one branch using five loss functions

In this section, we compare the performance of single-branch networks (ResneXt-50 and Inception-v3) and that of the proposed two branches network (REX-INCEP). Table 4 shows the performances obtained with the ResneXt-50 network when five loss functions were used. From the results, it can be seen that the loss function MSE gives the best performance. Table 5 contains the results for the Inception-v3 network when five loss functions were used. From these results, we can conclude that the dynamic Huber loss function gives the best performance.

Table 6 shows the results of our proposed two branches network (REX-INCEP) when four loss functions (MSE, dynamic ParamSmoothL1, dynamic Huber and dynamic Tukey losses) are used. Among the results obtained, dynamic ParamSmoothL1 achieved the best performance. Moreover, for a given loss function, the performance of the two-branch solution was better than that of the one-branch solution. The exception is the dynamic Huber loss. However, this difference in performance is very small. Comparing the results of one-branch networks and two-branch networks, we find that the two-branch network achieves high performance for all loss functions, with ParamSmoothL1 being the best. In contrast, the one-branch networks achieved competitive performance only for the MSE loss with the ResneXt-50 network and the dynamic Huber loss with the Inception-v3 network. This proves the effectiveness of the proposed REX-INCEP network, which effectively fuses and transforms the features generated by each architecture.

4.4.5. CNN ensemble

To increase the performance of FBP, we will use an ensemble of trained CNN architectures and use different loss functions. In this scenario, the final score is set to the average of the facial beauty scores provided by different models. In this group of experiments, six models are adopted: the two trained networks with one branch and the best loss functions (ResneXt-50 with MSE and Inception-v3 with dynamic Huber) and four

Table 3
Comparison between dynamic and fixed parameters of loss functions ParamSmoothL1, Huber and Tukey using ResneXt-50 network.

Loss function	Parameter	PC ↑	MAE ↓	RMSE ↓	ϵ-error ↓
ParamSmoothL1	α = 0.7	0.9122	0.2127	0.2799	0.0814
	α = 0.6	0.9141	0.2098	0.2772	0.0803
	α = 0.5	0.9132	0.2110	0.2780	0.0815
	α = 0.4	0.9101	0.2146	0.2825	0.0833
	α = 0.3	0.9116	0.2150	0.2810	0.0831
	Mean	0.9123	0.2126	0.2797	0.0819
	dynamic α (0.7–0.3)	**0.9140**	**0.2104**	**0.2777**	**0.0805**
Huber	β = 0.7	0.9126	0.2114	0.2796	0.0812
	β = 0.6	0.9130	0.2107	0.2780	0.0804
	β = 0.5	0.9144	0.2111	0.2770	0.0808
	β = 0.4	0.9124	0.2122	0.2783	0.0839
	β = 0.3	0.9110	0.2155	0.2811	0.0845
	Mean	0.9127	0.2122	0.2788	0.0822
	dynamic β (0.7–0.3)	**0.9141**	**0.2116**	**0.2777**	**0.0814**
Tukey	c = 2.0	0.9128	0.2155	0.2810	0.0837
	c = 1.7	0.9116	0.2133	0.2805	0.0821
	c = 1.5	0.9126	0.2129	0.2808	0.0824
	Mean	0.9123	0.2139	0.2808	0.0827
	dynamic c (2–1.5)	0.9127	0.2120	0.2801	0.0819
	dynamic c (2–1)	**0.9133**	**0.2100**	**0.2780**	**0.0802**

Table 4
Facial Beauty Prediction using ResneXt-50 Network with five loss functions (L_1, MSE, dynamic ParamSmoothL1, dynamic Huber and dynamic Tukey losses).

Loss function	PC ↑	MAE ↓	RMSE ↓	ϵ-error ↓
L_1	0.9126	0.2113	0.2783	0.0810
MSE	**0.9146**	**0.2092**	**0.2763**	**0.0802**
Dyn. ParamSmoothL1 (0.7–0.3)	0.9140	0.2104	0.2777	0.0805
Dyn. Huber (0.7–0.3)	0.9141	0.2116	0.2777	0.0814
Dyn. Tukey (2–1)	0.9133	0.2100	0.2780	0.0802

Table 5
Facial Beauty Prediction using Inception-v3 Network with five loss functions (L_1, MSE, dynamic SmoothL1, dynamic Huber and dynamic Tukey).

Loss function	PC ↑	L_1 ↓	RMSE ↓	ϵ-error ↓
L_1	0.9103	0.2152	0.2832	0.0848
MSE	0.9112	0.2147	0.2814	0.0833
Dyn. ParamSmoothL1 (0.7–0.3)	0.9118	0.2129	0.2805	0.0836
Dyn. Huber (0.7–0.3)	**0.9139**	**0.2125**	**0.2779**	**0.0819**
Dyn. Tukey (2–1)	0.9124	0.2138	0.2794	0.0802

Table 6
Facial Beauty Prediction using the proposed two branches Network (REX-INCEP) with four loss functions (MSE, dynamic ParamSmoothL1, dynamic Huber and dynamic Tukey losses).

Loss function	PC ↑	MAE ↓	RMSE ↓	ϵ-error ↓
MSE	0.9159	0.2071	0.2739	0.0790
Dyn. ParamSmoothL1 (0.7–0.3)	**0.9165**	**0.2065**	**0.2736**	**0.0789**
Dyn. Huber (0.7–0.3)	0.9138	0.2113	0.2785	0.0818
Dyn. Tukey (2–1)	0.9149	0.2105	0.2766	0.0806

trained two branches networks with four loss functions (REX-INCEP with MSE, dynamic ParamSmoothL1, dynamic Huber and dynamic Tukey). The solution based on the six models is called CNN Ensemble Regression (CNN-ER) and the fusion scheme refers to the averaging of the predicted scores from more than one model. For each model, we selected the final model after training it with 40 epochs. The results are summarized in Table 7. This table presents three different ensemble solutions. According to the results depicted in this table, we find the mean scores of the models with two branches perform better than the mean scores of the models with one branch. Moreover, the mean scores of the models with one and two branches (all six models) outperformed the mean scores of the models with one and two branches. We also note that the mean scores of the one branch networks perform better than one and two branches networks (Tables 4, 5, and 6). This proves the effectiveness of our proposed CNN ensemble method. It is worth noting that the individual models were trained only on the training set with a fixed number of epochs (40 epochs).

4.5. Experimental results using the five fold cross-validation scenario

In this section, we will use the provided five folds to perform the cross-validation experiments. Table 8 contains the results obtained with each fold, as well as the average over the five folds using the networks with one branch (ResneXt-50 with MSE loss function and Inception-v3 with dynamic Huber loss function) and four networks with two branches (REX-INCEP with MSE, dynamic ParamSmoothL1, dynamic Huber and dynamic Tukey loss function). It is worth noting that the presented result for each fold corresponds to the best result obtained by PC over the test

Table 7
Facial Beauty Prediction using the proposed CNN ensemble of different trained models on 60%–40% data split.

Fusion scheme	PC ↑	MAE ↓	RMSE ↓	ϵ-error ↓
One branch (Mixture 2 models)	0.9190	0.2054	0.2705	0.0777
Two branches (Mixture 4 models)	0.9194	0.2043	0.2699	0.0771
CNN-ER (Mixture 6 models)	**0.9207**	**0.2032**	**0.2683**	**0.0764**

Table 8
Five-fold cross-validation of facial beauty prediction using networks with one branch (ResneXt-50 with MSE loss and Inception-v3 with dynamic Huber loss) and two branches (REX-INCEP with MSE, dynamic ParamSmoothL1, dynamic Huber and dynamic Tukey losses).

Architecture	Fold	PC ↑	MAE ↓	RMSE ↓	ϵ-error ↓
Inception-v3 with dynamic Huber loss function	Fold 1	0.9149	0.2142	0.2779	0.0799
	Fold 2	0.9138	0.2074	0.2791	0.0798
	Fold 3	0.9203	0.2122	0.2801	0.0800
	Fold 4	0.9226	0.2072	0.2684	0.0783
	Fold 5	0.9202	0.2056	0.2706	0.0777
	Mean	**0.9184**	**0.2093**	**0.2752**	**0.0791**
ResneXt-50 with MSE loss function	Fold 1	0.9176	0.2072	0.2744	0.0765
	Fold 2	0.9114	0.2163	0.2865	0.0845
	Fold 3	0.9204	0.2106	0.2759	0.0783
	Fold 4	0.9228	0.2066	0.2686	0.0767
	Fold 5	0.9204	0.2053	0.2702	0.0760
	Mean	**0.9185**	**0.2092**	**0.2751**	**0.0784**
REX-INCEP with MSE loss function	Fold 1	0.9190	0.2081	0.2722	0.0772
	Fold 2	0.9172	0.2068	0.2755	0.0783
	Fold 3	0.9212	0.2085	0.2748	0.0783
	Fold 4	0.9252	0.2045	0.2654	0.0767
	Fold 5	0.9213	0.2049	0.2691	0.0756
	Mean	**0.9208**	**0.2066**	**0.2714**	**0.0772**
REX-INCEP with dynamic ParamSmoothL1 loss function	Fold 1	0.9202	0.2070	0.2718	0.0763
	Fold 2	0.9167	0.2056	0.2766	0.0774
	Fold 3	0.9206	0.2095	0.2763	0.0780
	Fold 4	0.9253	0.2053	0.2634	0.0759
	Fold 5	0.9238	0.2011	0.2639	0.0742
	Mean	**0.9213**	**0.2057**	**0.2704**	**0.0764**
REX-INCEP with dynamic Huber loss function	Fold 1	0.9196	0.2078	0.2722	0.0767
	Fold 2	0.9167	0.2033	0.2742	0.0771
	Fold 3	0.9238	0.2084	0.2717	0.0775
	Fold 4	0.9282	0.1992	0.2605	0.0720
	Fold 5	0.9227	0.2041	0.2666	0.0755
	Mean	**0.9222**	**0.2046**	**0.2690**	**0.0758**
REX-INCEP with dynamic Tukey loss function	Fold 1	0.9178	0.2079	0.2738	0.0766
	Fold 2	0.9166	0.2082	0.2782	0.0790
	Fold 3	0.9222	0.2089	0.2722	0.0773
	Fold 4	0.9242	0.2036	0.2648	0.0755
	Fold 5	0.9205	0.2076	0.2702	0.0770
	Mean	**0.9203**	**0.2072**	**0.2718**	**0.0771**

data during the training of 40 epochs. The cross-validation results can provide a better comparison between the networks and the loss functions. Comparing the networks, we can find that the trained networks with two branches outperform the networks with one branch. This is consistent with the conclusion found in the 60%–40% split scenario.

Moreover, we can observe that ResneXt-50 with the MSE loss function and Inception-v3 with the dynamic Huber loss function achieve similar results. Based on the results obtained with the two branch solutions, we can conclude that the dynamic Huber loss function achieves the best performance. On the other hand, the dynamic loss function ParamSmoothL1 outperforms the dynamic loss functions Tukey and MSE, which obtained similar results.

Table 9 shows the performances achieved by CNN ensembles. Similar to the ensemble experiments in the previous scenario, we consider an ensemble of one branch networks, an ensemble of two branch networks, and the mixture ensemble of all networks. Comparing the results of the 8 and 9, we can make the following observations:

Table 9
Five folds cross-validation of Facial Beauty Prediction using the proposed CNN ensemble of different trained models.

Fusion scheme	Fold	PC ↑	MAE ↓	RMSE ↓	ϵ-error ↓
One branch (2 models)	Fold 1	0.9205	0.2065	0.2703	0.0753
	Fold 2	0.9170	0.2060	0.2763	0.0788
	Fold 3	0.9245	0.2029	0.2691	0.0745
	Fold 4	0.9263	0.2015	0.2630	0.0747
	Fold 5	0.9234	0.2010	0.2652	0.0736
	Mean	**0.9223**	**0.2036**	**0.2688**	**0.0754**
Two branches (4 models)	Fold 1	0.9223	0.2035	0.2683	0.0743
	Fold 2	0.9202	0.2018	0.2713	0.0757
	Fold 3	0.9252	0.2053	0.2698	0.0751
	Fold 4	0.9289	0.1995	0.2586	0.0730
	Fold 5	0.9253	0.1994	0.2622	0.0725
	Mean	**0.9244**	**0.2019**	**0.2660**	**0.0741**
CNN-ER (Mixture 6 models)	Fold 1	0.9232	0.2026	0.2667	0.0735
	Fold 2	0.9204	0.2016	0.2710	0.0756
	Fold 3	0.9264	0.2029	0.2675	0.0738
	Fold 4	0.9292	0.1990	0.2583	0.0727
	Fold 5	0.9257	0.1984	0.2615	0.0720
	Mean	**0.9250**	**0.2009**	**0.2650**	**0.0735**

Table 10
Comparison with the state-of-the-arts methods using the 60%–40% split.

Method	PC ↑	MAE ↓	RMSE ↓
LR [12]	0.5948	0.4289	0.5531
GR [12]	0.6738	0.3914	0.5085
SVR [12]	0.6668	0.3898	0.5132
Alexnet [12]	0.8298	0.2938	0.3819
Resnet-18 [12]	0.8513	0.2818	0.3703
ResneXt-50 [12]	0.8777	0.2518	0.3325
CNN with SCA [19]	0.8780	0.2517	0.3320
Dynamic ParamSmoothL1* (Ours)	0.9165	0.2065	0.2736
CNN-ER (Ours)	**0.9207**	**0.2032**	**0.2683**

Dynamic ParamSmoothL* is our REX-INCEP network that was trained using the dynamic ParamSmoothL1 loss function.

- From the results of the one-branch ensemble, it can be seen that the fusion scheme outperforms the individual one-branch networks (ResneXt-50 with MSE loss function and Inception-v3 with dynamic Huber loss function).
- For the two-branch ensemble results, the fusion scheme outperforms all the results obtained by the single two-branch networks (REX-INCEP with MSE, dynamic ParamSmoothL1, dynamic Huber and dynamic Tukey loss functions).
- From the results of the mixture ensemble (all six models), it is clear that the fusion scheme outperforms not only the one- and two-branch networks, but also their fused models.

The above observations prove the effectiveness of the proposed fusion scheme. This also shows the efficiency of using two branch networks with different loss functions.

4.6. Comparison with state-of-the-art methods

In this section, we compare our proposed methods with the state-of-the-art methods in both scenarios: 60%–40% split and five-fold cross-validation.

Table 10 shows a comparison between our method and the state-of-the-art methods using the 60%–40% split. The comparison shows that our approach (CNN-ER) outperforms the state-

Table 11
Comparison with the State-of-the-Arts methods using the five-fold cross-validation scenario.

PC ↑	1	2	3	4	5	Mean
Alexnet [12]	0.8667	0.8645	0.8615	0.8678	0.8566	0.8634
Resnet-18 [12]	0.8847	0.8792	0.8929	0.8932	0.9004	0.8900
ResneXt-50 [12]	0.8985	0.8932	0.9016	0.899	0.9064	0.8997
CNN with SCA [19]	0.8990	0.8939	0.9020	0.8999	0.9067	0.9003
PI-CNN [45][a]	–	–	–	–	–	0.8978
CNN + LDL [46][a]	–	–	–	–	–	0.9031
ResNet-18 based AaNet [20]	–	–	–	–	–	0.9055
ResneXt-50-R^{3}CNN [21]	0.9143	0.9066	0.9136	0.9146	0.9217	0.9142
Dynamic ParamSmoothL1* (Ours)	0.9202	0.9167	0.9206	0.9253	0.9238	0.9213
CNN-ER (Ours)	**0.9232**	**0.9204**	**0.9264**	**0.9292**	**0.9257**	**0.9250**
MAE ↓	1	2	3	4	5	Mean
Alexnet [12]	0.2633	0.2605	0.2681	0.2609	0.2728	0.2651
Resnet-18 [12]	0.2480	0.2459	0.243	0.2383	0.2383	0.2419
ResneXt-50 [12]	0.2306	0.2285	0.226	0.2349	0.2258	0.2291
CNN with SCA [19]	0.2300	0.2284	0.2257	0.2345	0.2251	0.2287
PI-CNN [45][a]	–	–	–	–	–	0.2267
CNN + LDL [46][a]	–	–	–	–	–	0.2201
ResNet-18 based AaNet [20]	–	–	–	–	–	0.2236
ResneXt-50-R^{3}CNN [21]	0.2109	0.2152	0.2126	0.2130	0.2085	0.2120
Dynamic ParamSmoothL1* (Ours)	0.2070	0.2056	0.2095	0.2053	0.2011	0.2057
CNN-ER (Ours)	**0.2026**	**0.2016**	**0.2029**	**0.1990**	**0.1984**	**0.2009**
RMSE ↓	1	2	3	4	5	Mean
Alexnet [12]	0.3408	0.3449	0.3538	0.3438	0.3576	0.3481
Resnet-18 [12]	0.3258	0.3286	0.3184	0.3107	0.2994	0.3166
ResneXt-50 [12]	0.3025	0.3084	0.3016	0.3044	0.2918	0.3017
CNN with SCA [19]	0.3020	0.3081	0.3013	0.3039	0.2916	0.3014
PI-CNN [45][a]	–	–	–	–	–	0.3016
CNN + LDL [46][a]	–	–	–	–	–	0.2940
ResNet-18 based AaNet [20]	–	–	–	–	–	0.2954
ResneXt-50-R^{3}CNN [21]	0.2767	0.2895	0.2837	0.2804	0.2701	0.2800
Dynamic ParamSmoothL1* (Ours)	0.2718	0.2766	0.2763	0.2634	0.2639	0.2704
CNN-ER (Ours)	**0.2667**	**0.2710**	**0.2675**	**0.2583**	**0.2615**	**0.2650**

[a]The authors of [21] used ResNeXt-50 as a backbone network to re-implement the [45,46] methods on the newly created SCUT-FBP5500 dataset.
Dynamic ParamSmoothL* is our REX-INCEP network trained with the dynamic loss function ParamSmoothL1.

of-the-art methods in the three evaluation metrics (PC, MAE and RMSE). In addition to CNN-ER, we compare the results of the proposed two branches network (REX-INCEP) trained with the proposed dynamic loss function ParamSmoothL1 with the state-of-the-art methods. This comparison shows that the proposed REX-INCEP with the dynamic ParamSmoothL1 loss function performs better than the state-of-the-art methods in the three evaluation metrics (PC, MAE and RMSE). This proves that the superiority of our approach over the state-of-the-art methods is not only due to the ensemble of models, but both the proposed two branches network and the dynamic loss functions play a crucial role in achieving such performance.

Table 11 shows a comparison between our method and state-of-the-art methods using the five-fold cross-validation experiments and their average. Three evaluation metrics (PC, MAE and RMSE) are used for this comparison. The comparison shows that our approach performs better than the state-of-the-art methods, both in terms of average performance and performance of individual folds for all the evaluation metrics used. Similar to the comparison for the 60%–40% split, the proposed REX-INCEP with the dynamic loss function ParamSmoothL1 is shown to perform better than the state-of-the-art methods in all three evaluations.

metrics (PC, MAE and RMSE). This confirms that both the proposed two branches network and the dynamic loss functions play a crucial role in outperforming the state-of-the-art methods. The comparisons of the two scenarios (60%–40% split and five fold cross-validation) demonstrate the efficiency of our proposed approach. This tends to confirm that both the proposed two branches network and the dynamic loss functions played a crucial role in outperforming the State-of-the-Art methods. The comparisons of both scenarios (60%–40% split and five folds cross-validation) prove the efficiency of our proposed approach.

5. Conclusion

In this paper, we address the evaluation of the beauty of faces in facial images using Deep Learning. First, we propose a two-branch architecture (REX-INCEP) based on merging the architecture of two already trained networks. Second, we introduce a dynamic law to control the behaviour of the robust regression loss functions during training, making the robust losses adaptive and dynamic. Third, we propose an ensemble regression based on Convolutional Neural Networks (CNN-ER).

In the ensemble method, the CNNs ResneXt-50, Inception-v3 and the proposed REX-INCEP architectures are used. The latter is a two branches CNN architecture that combines the ResneXt-50 and Inception-v3 architectures through FC layers. The main advantage of our REX-INCEP architecture is the ability to learn high-level FBP features simultaneously with ResneXt and Inception blocks.

In addition to using CNN architectures, several loss functions are used, namely MSE and the proposed dynamic ParamSmoothL1, dynamic Huber and dynamic Tukey. For the dynamic loss functions (ParamSmoothL1, Huber and Tukey), a cosine law is proposed to reduce the robust loss parameter during training. The dynamic schemes have been shown to be very efficient, both in terms of performance and in avoiding the grid search for the best value, which has a high computational cost.

Our approach CNN-ER averages the predictions of six models, namely: the two trained one-branch networks (ResneXt-50

with MSE loss function and Inception-v3 with dynamic Huber loss function) and four trained two-branch networks with four loss functions (REX-INCEP with MSE, dynamic ParamSmoothL1, dynamic Huber and dynamic Tukey). The obtained results show the superiority of the proposed REX-INCEP over ResneXt-50 and Inception-v3 networks. Moreover, the proposed approach (CNN-ER) outperforms not only single and two branches networks but also their fused models. The proposed architecture REX-INCEP and CNN-ER outperformed many CNN baselines as well as many published state-of-the-art solutions. This superior performance was achieved in the two evaluation protocols related to the SCUT-FBP5500 dataset: 60%–40% and five cross-validations using the three evaluation metrics (PC, MAE and RMSE).

We have also found that using the proposed dynamic robust loss functions generally leads to better estimates. In addition, we found that the best loss function may depend on the data used and the CNN architecture.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was partially funded by the University of the Basque Country, GUI19/027.

References

[1] K. Dion, E. Berscheid, E. Walster, What is beautiful is good, J. Pers. Soc. Psychol. 24 (3) (1972) 285.
[2] J. Gan, L. Xiang, Y. Zhai, C. Mai, G. He, J. Zeng, Z. Bai, R.D. Labati, V. Piuri, F. Scotti, 2M BeautyNet: Facial beauty prediction based on multi-task transfer learning, IEEE Access 8 (2020) 20245–20256.
[3] Y. Eisenthal, G. Dror, E. Ruppin, Facial attractiveness: Beauty and the machine, Neural Comput. 18 (1) (2006) 119–142.
[4] X. Liu, T. Li, H. Peng, I.C. Ouyang, T. Kim, R. Wang, Understanding beauty via deep facial features, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, CVPRW, 2019, pp. 246–256.
[5] T. Alashkar, S. Jiang, Y. Fu, Rule-based facial makeup recommendation system, in: 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition, FG 2017, IEEE, 2017, pp. 325–330.
[6] A. Laurentini, A. Bottino, Computer analysis of face beauty: A survey, Comput. Vis. Image Underst. 125 (2014) 184–199.
[7] L. Liang, L. Jin, X. Li, Facial skin beautification using adaptive region-aware masks, IEEE Trans. Cybern. 44 (12) (2014) 2600–2612.
[8] L. Xu, H. Fan, J. Xiang, Hierarchical multi-task network for race, gender and facial attractiveness recognition, in: 2019 IEEE International Conference on Image Processing, ICIP, IEEE, 2019, pp. 3861–3865.
[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.
[10] D. Xie, L. Liang, L. Jin, J. Xu, M. Li, SCUT-FBP: A benchmark dataset for facial beauty perception, in: 2015 IEEE International Conference on Systems, Man, and Cybernetics, IEEE, 2015, pp. 1821–1826.
[11] L. Xu, J. Xiang, X. Yuan, Transferring rich deep features for facial beauty prediction, 2018, arXiv preprint arXiv:1803.07253.
[12] L. Liang, L. Lin, L. Jin, D. Xie, M. Li, SCUT-FBP5500: A diverse benchmark dataset for multi-paradigm facial beauty prediction, in: 2018 24th International Conference on Pattern Recognition, ICPR, 2018.
[13] D. Gray, K. Yu, W. Xu, Y. Gong, Predicting facial beauty without landmarks, in: European Conference on Computer Vision, Springer, 2010, pp. 434–447.
[14] D. Zhang, Q. Zhao, F. Chen, Quantitative analysis of human facial beauty using geometric features, Pattern Recognit. 44 (4) (2011) 940–950.
[15] P. Aarabi, D. Hughes, K. Mohajer, M. Emami, The automatic measurement of facial beauty, in: 2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat. No. 01CH37236), vol. 4, IEEE, 2001, pp. 2644–2647.
[16] H. Yan, Cost-sensitive ordinal regression for fully automatic facial beauty assessment, Neurocomputing 129 (2014) 334–342.
[17] W.-C. Chiang, H.-H. Lin, C.-S. Huang, L.-J. Lo, S.-Y. Wan, The cluster assessment of facial attractiveness using fuzzy neural network classifier based on 3D moiré features, Pattern Recognit. 47 (3) (2014) 1249–1260.
[18] J. Fan, K.P. Chau, X. Wan, L. Zhai, E. Lau, Prediction of facial attractiveness from facial proportions, Pattern Recognit. 45 (6) (2012) 2326–2334.
[19] K. Cao, K.-n. Choi, H. Jung, L. Duan, Deep learning for facial beauty prediction, Information 11 (8) (2020) 391.
[20] L. Lin, L. Liang, L. Jin, W. Chen, Attribute-aware convolutional neural networks for facial beauty prediction, in: IJCAI, 2019, pp. 847–853.
[21] L. Lin, L. Liang, L. Jin, Regression guided by relative ranking using convolutional neural network (R3CNN) for facial beauty prediction, IEEE Trans. Affect. Comput. (2019).
[22] T. Ahonen, A. Hadid, M. Pietikainen, Face description with local binary patterns: Application to face recognition, IEEE Trans. Pattern Anal. Mach. Intell. 28 (12) (2006) 2037–2041.
[23] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[24] Z. Cao, Q. Yin, X. Tang, J. Sun, Face recognition with learning-based descriptor, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 2707–2714.
[25] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[26] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[27] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1492–1500.
[28] F. Dornaika, K. Wang, I. Arganda-Carreras, A. Elorza, A. Moujahid, Toward graph-based semi-supervised face beauty prediction, Expert Syst. Appl. 142 (2020) 112990.
[29] F. Dornaika, A. Moujahid, K. Wang, X. Feng, Efficient deep discriminant embedding: Application to face beauty prediction and classification, Eng. Appl. Artif. Intell. 95 (2020) 103831.
[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.
[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[32] F. Bougourzi, K. Mokrani, Y. Ruichek, F. Dornaika, A. Ouafi, A. Taleb-Ahmed, Fusion of transformed shallow features for facial expression recognition, IET Image Process. 13 (9) (2019) 1479–1489.
[33] F. Bougourzi, F. Dornaika, K. Mokrani, A. Taleb-Ahmed, Y. Ruichek, Fusing transformed deep and shallow features (FTDS) for image-based facial expression recognition, Expert Syst. Appl. 156 (2020) 113459.
[34] R. Girshick, Fast R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.
[35] I. Loshchilov, F. Hutter, SGDR: Stochastic gradient descent with warm restarts, in: International Conference on Learning Representation, 2017.
[36] P.J. Huber, Robust estimation of a location parameter, in: Breakthroughs in Statistics, Springer, 1992, pp. 492–518.
[37] M.J. Black, A. Rangarajan, On the unification of line processes, outlier rejection, and robust statistics with applications in early vision, Int. J. Comput. Vis. 19 (1) (1996) 57–91.
[38] V. Belagiannis, C. Rupprecht, G. Carneiro, N. Navab, Robust optimization for deep regression, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2830–2838.
[39] K. Pearson, VII. note on regression and inheritance in the case of two parents, Proc. R. Soc. Lond. 58 (347–352) (1895) 240–242.
[40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, 2019, pp. 8026–8037.
[41] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[42] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2015, arXiv:1409.1556.
[43] S. Zagoruyko, N. Komodakis, Wide residual networks, 2017, arXiv:1605.07146.
[44] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016.
[45] J. Xu, L. Jin, L. Liang, Z. Feng, D. Xie, H. Mao, Facial attractiveness prediction using psychologically inspired convolutional neural network (PI-CNN), in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2017, pp. 1657–1661.
[46] Y.-Y. Fan, S. Liu, B. Li, Z. Guo, A. Samal, J. Wan, S.Z. Li, Label distribution-based facial attractiveness computation by deep residual learning, IEEE Trans. Multimed. 20 (8) (2017) 2196–2208.