Author' s Accepted Manuscript, AAM

DeepFlash: Turning a flash selfie into a studio portrait ☆

Nicola Capece [a,*], Francesco Banterle [b], Paolo Cignoni [b], Fabio Ganovelli [b], Roberto Scopigno [b], Ugo Erra [c]

[a] University of Basilicata, Potenza, Italy
[b] ISTI-CNR, Pisa, Italy
[c] University of Basilicata, Potenza, Italy

ARTICLE INFO

Keywords:
Image enhancement
Machine learning algorithms
Deep learning
Computational photography
Image processing

ABSTRACT

We present a method for turning a flash selfie taken with a smartphone into a photograph as if it was taken in a studio setting with uniform lighting. Our method uses a convolutional neural network trained on a set of pairs of photographs acquired in an ad-hoc acquisition campaign. Each pair consists of one photograph of a subject's face taken with the camera flash enabled and another one of the same subject in the same pose illuminated using a photographic studio-lighting setup. We show how our method can amend defects introduced by a close-up camera flash, such as specular highlights, shadows, skin shine, and flattened images.

1. Introduction

With the steady improvement of built-in digital cameras, pictures taken on smartphones and computer tablets are becoming increasingly predominant on the Internet, even on web-based services dedicated to quality photography such as Flickr, 500px, and Instagram. Although smartphones can take pictures that are comparable to those of digital reflex cameras in favorable lighting conditions, they perform poorly in low light conditions. This is mainly due to the size of their sensors, which is a constraint difficult to overcome within the small room on a smartphone. Therefore, taking pictures in low light often triggers the camera flash, which is typically a low-power LED flash mounted side by side with the camera lens that produces several artifacts.

One of the most common type of photograph taken with a smartphone is the so-called selfie, which is a picture of one's face taken by holding the phone in the hand or by using a "selfie stick". Low-light flash photographs and selfies are an unfavorable combination that produces images with specular highlights, sharp shadows, and flat and unnatural skin tones. Therefore, researchers have recently started to develop several correction techniques: re-lighting and enhancement of images with non-uniform lighting [1,2], some of them have focused mainly on the images of faces [3–6]. In this paper, we explore the possibility of taking smartphone flash selfies and employing a convolutional neural network (CNN) to turn them into studio portraits. Researchers have extensively used CNNs to improve pictures, for example for creating high dynamic range images from single exposure [7], colorization [8], super-resolution [9], and so on, as will be discussed in Section 2.3. However, our problem is especially challenging for at least three reasons. Firstly, it concerns an effect that has both local and global discriminant features such as highlight and skin tone, respectively. Secondly, we want to imitate a process that humans are very good at performing; i.e., to picture what an image would look like if flash was not used. Finally, both previous points apply to the domain of human faces, on which humans are extremely good to spot any kind of inconsistencies. We leverage the fact that, by their nature, smartphone flash selfies share many common traits and make a fairly well-defined subdomain of photographs: they are front or three-quarter single-face portraits, taken from less than one meter away with a single flash located with the camera lenses. Our approach consists of training a CNN with a series of pairs of portraits, where one is taken with the smartphone flash and one with photographic studio illumination. The two photographs of the same pair are taken as simultaneously as possible, so that the pose of the subject is the same. Although the flash no-flash problem applies to a wider application domain, the collection of images useful for this aim is still a challenge today, considering the huge amount of data required [10] (i.e., the order of hundreds of thousand). As explained by Sun et al. [11], focusing on a specific category for a deep learning algorithm is very important, whereas adding random categories could reduce the performance. For this reason, the choice of the restricted dataset and task to improve the only selfie leads to a more specialized deep learning algorithm and

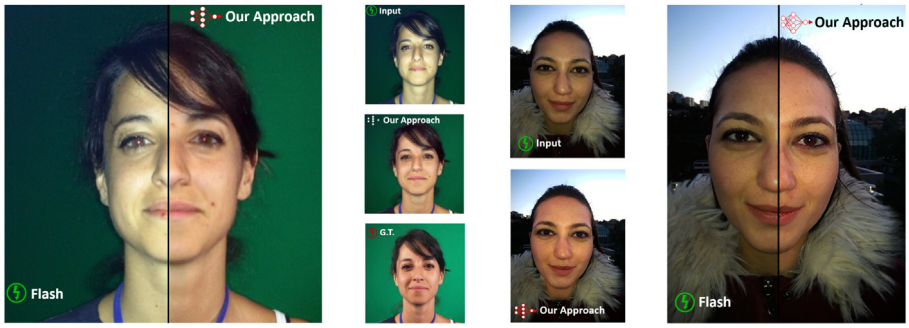

Fig. 1. Two examples from our results. The split images show a comparison between the input and the output of our algorithm. In the central column the input, output, and ground truth images.

results in higher performance. Moreover, one of our contributions was to build a specific dataset, which size represents a lower bound useful to train CNN to perform this task.

The remainder of this paper is structured as follows: In Section 2, we review the literature most relevant to our problem. In Section 3, we introduce the network we designed and the way it is used to solve our problem. In Section 4, we describe the experimental setup used to create the training dataset. In Section 5, we show and discuss the results obtained with our method, in Section 6, we comment on its limitations and finally we draw our conclusion and address future work in Section 7.

2. Related work

In this section, we review works related to our work and photography per se; that is, without a focus on acquisition for computer vision/graphics tasks (e.g., color acquisition for three-dimensional (3D) meshes).

2.1. Flash photography

In two contemporary works, Petschnigg et al. [12] and Eisemann and Durand [13] proposed the idea of using a flash photograph with low ISO (i.e., low noise) to transfer the ambiance of the available lighting into a non-flash photograph of the same subjects/scene, to reduce noise. This is because non-flash photographs taken in dark environments suffer from high noise to avoid blur. Other works [14] have developed this idea further by removing over/under-illumination at a given flash intensity, reflections, highlights, and attenuation over depth.

2.2. Deep learning

Deep learning (DL) [15] is a framework inspired by biology that enables computational models (made of nonlinear projection) to learn high-level representation from data, by back-propagating errors [16] with respect to the model parameters. The most important advantage of DL is that it avoids feature engineering that is automatically inferred from raw data. One class of DL architecture of particular interest is the CNN class, namely, DL models that mimic how the visual cortex works. Krizhevsky et al.'s work [17] on CNNs has sparked a renaissance in the field of DL. Furthermore, this work has pushed forward the state of the art in many imaging tasks. This has been made possible because of three key factors: the increasing computing power of modern graphics processing units (GPUs), the availability of huge datasets, and novel and powerful algorithms and frameworks for CNNs. To improve their learning invariance, CNNs are typically trained on augmented data [18]; as will be shown later, data augmentation was found to be essential in our experimentation.

2.3. Deep learning and computational photography

Recently, CNNs have been applied to numerous computer vision, imaging, and computer graphics tasks [19–24]. Furthermore, they are becoming extremely popular, and novel architectures and algorithms are continually popping up overnight.

In this section, we review works with similar aims to ours, namely, starting from a single image and attempting to improve its quality or aesthetics.

To improve the editing of selfies, Shen et al. [25] extended the FCN-8s framework [26] to automatically segment portraits. This allows a user to automatically edit/augment portraits. For example, users can change the background, stylize the selfie, enhance the depth-of-field, etc. Zhang et al. [27] proposed an end-to-end learning approach for single-image reflection separation with perceptual losses and a customized exclusion loss. Their method can be used to remove or reduce unwanted reflections in pictures.

Eilertsen et al. [7] introduced a U-Net architecture [28] for expanding the dynamic range of low dynamic range images to obtain high dynamic range images. Similarly, Chen et al. [29] showed that U-Nets can be used successfully to debayer images captured at low-light conditions and high ISO, which typically exhibit noise. They extensively studied different approaches on real-world low light with real noise. For example, they tested a variety of architectures, loss functions (e.g., L1 (least absolute deviations), L2 (least square errors), and the structural similarity index (SSIM)), and color inputs.

Aksoy et al. [30] presented a large-scale collection of pairs of images with ambient light and flashlight of the same scene. The images were obtained with the aid of casual photographers by using their smartphone cameras, and consequently, the dataset covers a wide variety of scenes. The dataset was provided in order to be studied in future works for high-level tasks such as semantic segmentation or depth estimation. Unlike their dataset, whose objective is to provide matching between two images under uncontrolled lighting conditions, our dataset aims to change the lighting scheme by turning from flash lighting to a controlled photograph studio light.

To the best of our knowledge, there have not been proposed approaches for removing flash artifacts such as hard shadows and highlights from selfies as the work presented in this paper.

3. Our approach

We designed a regression model targeted to the restricted domain of human faces. We adopted a supervised approach where a CNN is trained by feeding pair of flash and no flash portraits. Although the main idea is straightforward, there are several details that need to be addressed in the design of the network, the training procedure, the way the problem is encoded in the network, and finally the loss function. All these aspects are discussed in the following sections.

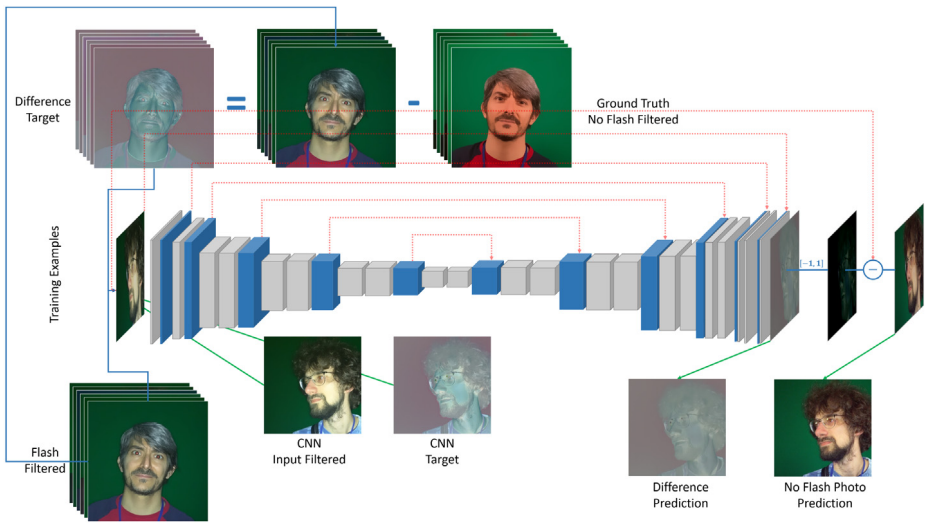

Fig. 2. Our neural network architecture for transforming a flash image into a non-flash image. The first 13 blocks represent the VGG-16's convolutional layers, which perform the image encoding. The second part reconstructs the output image and has several convolutional and deconvolutional layers. From the blue blocks of the VGG-16, the shortcut connections start, which are linked with their counterparts in the decoder. The CNN input is an image taken with the smartphone flash, and the ground truth is an image taken using simulated ambient light, on both of which the bilateral filter is applied. The target image is the difference between the input and ground truth image, normalized in a range between 0 and 1. The network prediction is the searched difference, which is denormalized in the range $[-1, 1]$ and then subtracted from the non-filtered input. The final output prediction is an image without flash highlights.

3.1. Deep neural network

Our CNN is an encoder–decoder structured neural network that consists of two sub-networks: the first network takes as input a flash image and performs the encoding to create a deep feature map representation; the second network takes as input the encoder's output and recreates the image without the flash defects. The input image encoding is performed by the well-known Visual Geometry Group's VGG-16[31] network. This is a CNN widely used for object detection and classification tasks, achieving 92.7% accuracy in results on 1000 classes of objects in ImageNet (a dataset of about 14 million images).

The network is composed of 16 layers, 13 of which are convolutional layers, and 3 are fully connected layers. For our purpose, we used only the convolutional layers, which are divided into five separate groups: the first two groups and the last three are composed, respectively, of two and three layers (Fig. 2). Each single CNN layer performs three operations in sequence: (i) A set of linear activations is produced using several parallel convolutions by a 3×3 kernel size, a stride value of 1, and a padding value set to "SAME" to preserve the spatial resolution between two convolutions; (ii) linear activation is performed through a nonlinear activation function called ReLU (Rectified Linear Unit) [32], which is defined as $ReLU(x) = max(0, x)$, and ensures that the gradients remain large and consistent; (iii) a max pooling [33] operation takes the output of the convolutional layer as input and reduces its size to match the input size of the next layer through a downsampling operation. In particular, max pooling takes the largest value from a window moved over the output of the convolutional layer. In this way, the extracted values from the pooling operation do not vary [34], and the output is invariant to small translations of the input.

We developed the decoder component, which performs the decoding task, based on Eilertsen et al.'s approach [7]. In particular, the input of the network is the output from the last VGG-16 convolutional layer after a further convolution operation. As the change of parameters of the CNN in the training phase changes the output distribution of each layer, it is necessary to normalize this distribution to produce an output that will be valid as the input for the next layer. For this reason, batch normalization [35] is performed after each convolution, forcing the input of the activation function to have mean zero and unit variance.

After each batch normalization, the obtained tensor, called the activation tensor, crosses the next activation function, LeakyReLU [36], which is a modified version of ReLU introducing a nonzero gradient for the next inputs. To obtain optimal CNN performance in the training phase, the slope parameter α is set to 0.2 [37].

The main features of the decoder layers are operations such as convolutions, batch normalization, deconvolutions, and concatenations. Because our network is composed of many layers, to avoid the vanishing gradient problem [38] studied by He et al. [39], we used an approach based on a residual learning network. This problem concerns the weight update in the backpropagation phase, which is proportional to the gradient of the error function compared with the weight that has to be updated. Progressing in the backpropagation phase and inverse crossing the network to update the weight may mean that the gradient is so small as to make inefficient updates on the weights belonging to the first deep neural network layers, and consequently, their training is stopped. A simple way to solve this problem is to use a block division of the VGG-16 and concatenate in depth each block's output with its counterpart in the decoder through a link called a shortcut connection. For this reason, we use concatenations layers.

In the image recognition task, the residual learning network is based on the assumption that the network has to learn a residual function. If we consider $H(x)$ to be the function that a layers group has to learn, and $F(x)$ to be the function that a layers group is able to approximate, whether at this function is concatenated to the group's input, it will obtain a function in the form $F(x) = H(x) - x$ and a residual function in the form $H(x) = F(x) + x$. In this way, the contiguous layers that belong to the group learn the residual function, and their input will be concatenated to their own output. A similar approach is often used in image-processing tasks [40], and it retrieves the information lost through the convolutions of the layer groups. This information can be retrieved to help the decoder to reconstruct the output image. The proposed encoder–decoder structured neural network works in the RGB (red, green, blue) domains only, and it is able to recreate similar input images of faces, but in a different light mode.

Starting from the VGG-16 output tensor, to reconstruct the output image, we use deconvolutional layers [41], which transpose the convolutional layers.

3.2. Training

Our CNN was trained to minimize the loss function using an algorithm called the Adam Optimizer [42]. This algorithm is a stochastic

gradient descent with momentum (SGDM) variant [16], which manages in a different way the problem of setting the learning rate. The choice of the learning rate can influence the CNN training convergence because a high value can lead to a possible divergence, while a very low value can lead to a slow convergence. SGDM addresses this problem by updating weights through a linear combination of the gradient and previous updates. Adam is an SGDM variant, which is based on two other well-known ones, called AdaGrad [43] and RMSProp [44]. These are classified in the category of adaptive algorithms because they adapt the learning rate for each of the parameters, leading to better convergence results. In particular, Adam combines the advantages of the methods mentioned above: It adapts the learning rate based on the first and second gradient moments. In our Adam configuration, the initial learning rate is set to 10^{-5}, while β_1 and β_2, called the forgetting factors, are left at the default values, 0.9 and 0.999, respectively. A parameter, ϵ, used to avoid divisions by zero, is set to 10^{-8}, and finally the mini-batch size was set as 4 due to the input image dimension and the GPU capability.

To increase the generalization level and to compensate for the amount of the training data available, we initialized the weights of the VGG-16 encoder using a pre-trained model, which is used for face recognition [45], exploiting the transfer learning concept [34]. The model was trained using a very large-scale dataset that consists of 2.6 million faces belonging to 2600 identities (about 1000 images for each identity), using four GPU Titan Blacks. The input images of the pre-trained model have a resolution of 224 × 224 pixels, from which was subtracted the mean of the training set's images.

The weights of our decoder were initialized using a continuous probability distribution called a truncated normal distribution [46], which ensures that the weights initialization has mean zero and unit standard deviation. This avoids increasing or dissolution of the gradient, which otherwise leads to critical errors during training. The weights of the last decoder layer were initialized using Xavier Initialization [47], which ensures that the weights are neither too large nor too small and that the signal passing through the neural network is propagated accurately. This prevents the signal from being amplified or reduced too much due to excessively large or small weight initialization.

3.3. Problem encoding

The network can be used in more than one way to achieve our goal. The table in Fig. 3 illustrates three alternative encodings of the problem, showing the neural network input, target, prediction, and algorithm output for each one. We indicate with x_i the flash image, o_i the uniformly lit image, and $\mathbf{y}_{\mathbf{i}_{[\mathbf{A}|\mathbf{B}|\mathbf{C}]}}$ the network prediction.

The most straightforward solution, shown in the first row of Fig. 3, consists of training the neural network by providing x_i as the input, o_i as the target, and, once trained, to take the network prediction $\mathbf{y}_{\mathbf{i}_\mathbf{A}}$ as the final output. With this setting, the network converged and gave good results in terms of colors and chromaticism. On the downside, the predictions were blurred, and small misalignments of facial expressions between x_i and o_i (e.g., eyes closed/open, the position of the lips, and other facial landmarks) created very visible artifacts in the predicted images.

To reduce the blur in the images, we can train the network, giving as input the flash image and as the target the difference between x_i and o_i. This way, the artifacts due to the alignment of facial expressions have been greatly reduced and the training has been simplified. Through this approach, shown in the second row of Fig. 3, the results visibly improved, but the blur was not entirely removed.

Inspired by these results, we chose an encoding that decouples high-frequency details such as hairs of facial features from low-frequency characteristics such as the global skin tone. We employed an accelerated version of the *Bilateral Filter* [48], which is a nonlinear filter that is ubiquitously used to smooth images preserving the edge features. Through this filter, we remove the input and the ground truth image's high frequencies, and we compute the distance between two images, normalizing it to $[0, 1]$. The network's input is the filtered flash image, and the target is the distance between the filtered input and the filtered ground truth:

$$t_i = \frac{[BL(x_i, \sigma_s, \sigma_r) - BL(o_i, \sigma_s, \sigma_r)] + 1}{2} \quad x_i, o_i, t_i \in [0, 1], \tag{1}$$

where BL is the bilateral filter operator, $\sigma_s = 16$ is the spatial sigma value, and $\sigma_r = 0.4$ is the range sigma value. These parameters were selected from guidelines parameters proposed by Durand and Dorsey [49] for separating low and high frequencies with a bilateral filter. Indeed, they suggested that σ_s needs to be equal to the 2% of the size of the maximum image dimension, and $\sigma_r = 0.4$. In our case, we slightly increased σ_s to 3% in order to increase quality when computing the filter with the used approximation [48] because samples are drawn proportionally to σ_s in this approximation.

The final reconstructed image is computed as:

$$pred_i = x_i - 2y_i + 1, \tag{2}$$

where y_i is the network output.

3.4. Loss function

As mentioned above, we trained our neural network with images filtered using the bilateral filter as the input, and the distance between input and ground truth filtered with the same filter as the target. The aim was to preserve the low frequencies and retrieve them in a subsequent step from the original non-filtered image. For this reason, we minimized the distance between the low frequencies of the input and ground truth. The objective function is therefore defined as follows:

$$L(y_d, t_d) = \frac{1}{3N} \sum_i \left((y_{di} - \mathbb{E}[y_{di}]) - (t_{di} - \mathbb{E}[t_{di}]) \right)^2, \tag{3}$$

where

$$\begin{aligned} y_{di} &= BL(x_i, \sigma_s, \sigma_r) - 2y_i + 1 \\ t_{di} &= BL(x_i, \sigma_s, \sigma_r) - 2t_i + 1 \end{aligned}. \tag{4}$$

In more specific terms, N is the number of pixels, $BL(x_i, \sigma_s, \sigma_r)$ is the CNN input, x_i is the flash image, y_i is the predicted difference of the CNN, and $t_i = BL(x_i, \sigma_s, \sigma_r) - BL(o_i, \sigma_s, \sigma_r)$, where o_i is the ground truth. The bilateral filter's arguments are the same in Eq. (1). Replacing Eq. (4) in Eq. (3), and by simplifying and exploiting the linear property of the mean, the objective function can be rewritten as:

$$L(y, t) = \frac{4}{3N} \sum_i \left((t_i - y_i) + \mathbb{E}[y_i - t_i] \right)^2. \tag{5}$$

To avoid negative values affecting the CNN convergence due to activation functions, we normalized the target difference image in the range $[0 \ldots 1]$ (2). In particular, since *ReLU* and *LeakyReLU* are non-saturating nonlinear activation functions [50,51], they tend to eliminate completely or partially the negative output values of each layer, leading to faster convergence than saturating nonlinear activation functions such as tanh, which lead to longer training times and a slower convergence. Furthermore, it has been shown that rectified units are much more efficient for tasks concerning images [52,53]. Therefore, the network will perform predictions in the $[0, 1]$ range; for this reason, the values are reported in the $[-1, 1]$ range and subsequently subtracted from the input values. If we consider the bilateral filter function and perform a further substitution, we can insert the original non-filtered images into Eq. (4), and the objective function can be explicitly rewritten as:

$$\begin{aligned} L(y, x, o) = \frac{4}{3N} \sum_i \Big(& (BL(x_i, \sigma_s, \sigma_r) - BL(o_i, \sigma_s, \sigma_r) - y_i) + \\ & + \mathbb{E}[y_i - BL(x_i, \sigma_s, \sigma_r) + BL(o_i, \sigma_s, \sigma_r)] \Big)^2 \end{aligned} \tag{6}$$

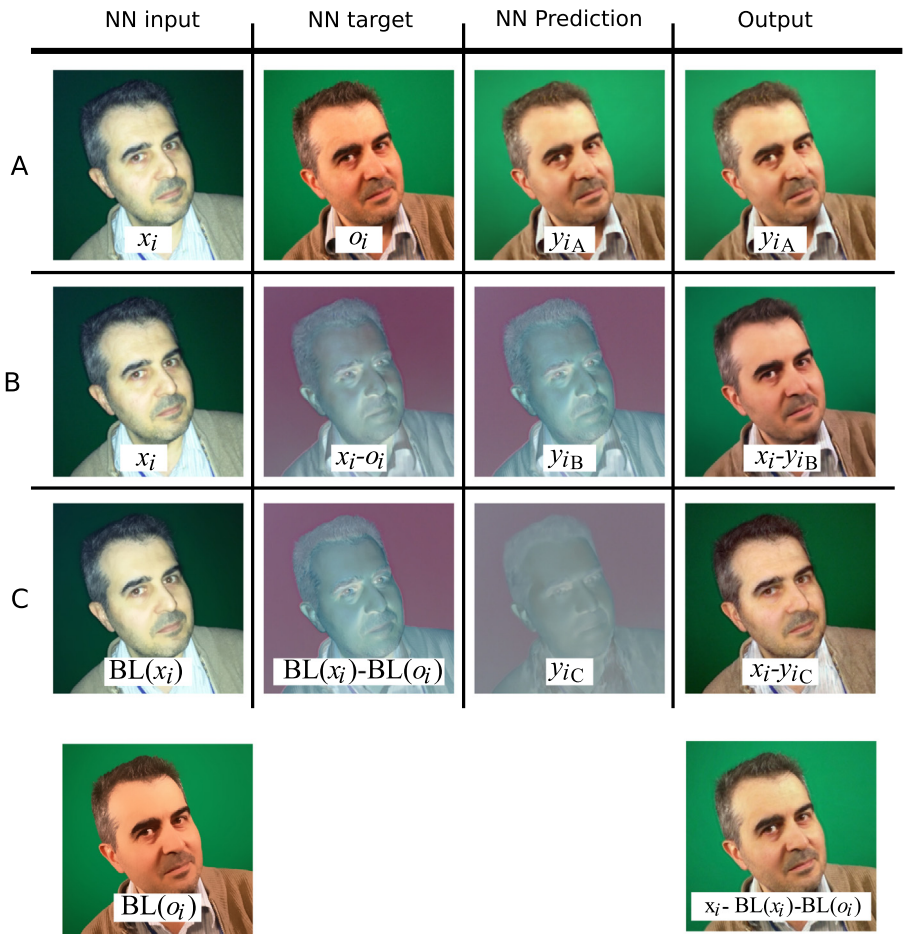

Fig. 3. Each row of the table shows a possible (and tested) encoding of the problem. For each row, the neural network (NN) input, target, prediction, and algorithm output are shown. The last row shows the bilateral filter applied to the uniform lit image (left) and the best achievable result with approach **C**.

Mean subtraction is performed for each channel of each image pixel by pixel only in the evaluation phase of the objective function, to centralize the data and to distribute the weights of each image across the training in a balanced manner, so that each image gives the same contribution to the training and does not have more or less important than the others. This normalization operation is performed differently for every single image, compared with the classic method of centralizing the data, which involves subtracting from each image the mean computed across the whole training dataset. Because our problem is confined to a specific domain, in which the data are stationary and the image lighting parameters are well defined and always the same both for the input and for the output, we subtracted the mean for each single image, which was computed on the same image to remove the average brightness or intensity from each pixel.

4. Experimental setup — dataset creation

We performed neural network training using pairs of photos taken with a 13 Megapixel Nexus 6 smartphone camera. The photographs were taken in a studio equipped with four Lupoled 560 lamps to provide uniform illumination. We choose not to use the synthetic data for pretraining because the variety and quality of the lighting artifacts that we aimed to capture and correct needed a high number of high quality photorealistic 3D models. The generation of such models was considered less practical and safe that direct acquisition of real examples. For each pose, a photograph was taken with the lamps on, which were then immediately switched off, and a second photograph was taken with the smartphone flash only. Because the switching off the lamp imposes a significant delay between the two shots (about 400 ms with our lamps), the pose of the face between the first and the second may change significantly.

To reduce the problem of face misalignment, we performed an *affine* alignment (i.e., translation, rotation, scale, and shear) using the MATLAB Image Processing Toolbox™. In particular, we considered the non-flash image as the misaligned image (M) and the flash image as our reference (R). Because the images have different lighting conditions, we employed a *multimodal* metric [54,55] and optimizer [56]. Once the geometric transformation was estimated, we applied it to M, obtaining M', which is a better alignment to R. Note that a limitation of this approach is that misalignments remain between open and closed eyes. After affine registration, we identified the face of a subject in M, M', and R using the Face Recognition API,[1] which returns a bounding box for the photograph. Each bounding box is used to crop the image, and then the image is downsampled to 512 × 512. As the aim of CNN is to automatically extract the input image features, preserving globally the spatial characteristics of the images such as the edges is very important especially in the first layers of the CNN [34]. In this way, the spatial structure of the images are preserved and equivariance to the translation of CNN is ensured.

We collected about 495 pair of photos of 101 of people (both females and males) in different poses. Then, we augmented the dataset in three ways. First, through 5 rotations from −20 to 20 degrees around the center of face bounding box, using a 10 degree step. Second, by cropping the image to the face bounding box and rescaling to original

[1] https://github.com/ageitgey/face_recognition.

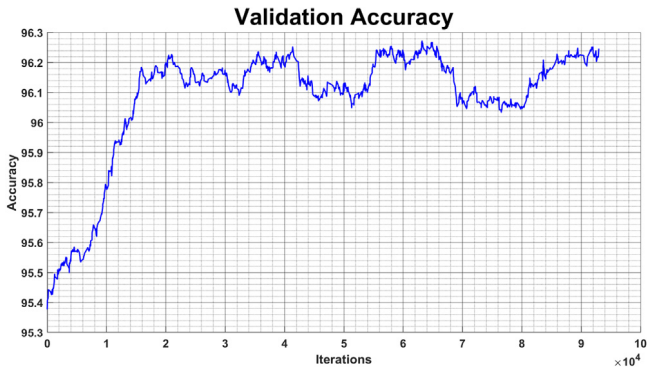

Fig. 4. Accuracy trend in the validation set at the end of training the CNN after 62 epochs and 458,000 iterations performed in 5 days.

image size. Finally, images are flipped horizontally. Altogether, we augmented the initial set of examples by a factor 20, so our training is performed with 9.900 images with a 3120 × 4160 resolution (13 Megapixel).

5. Results

We evaluated the results in validation and test sets (see Fig. 1). In particular, the CNN was trained using pairs of images with a resolution of 512 × 512 in about five days using a NVIDIA Titan Xp GPU and by performing 62 epochs and about 458,000 backpropagation iterations. We interrupted the training when the value of the loss function computed on 1500 images reached a low level of approximately 0.0042. To evaluate the similarity accuracy between the current prediction and ground truth, we computed the percentage difference between the two images as follows:

$$acc = 100 - \left(\frac{100}{3w(I)h(I)} \sum_i \sum_c |I_c - \tilde{I}_c| \right) \tag{7}$$

where $I = t_d, \tilde{I} = y_d, w(I) = width(I)$, and $h(I) = height(I)$. After the training step, we obtained an accuracy value of 96.2% (Fig. 4). In the test phase, we evaluated our approach using 740 test images, obtaining a loss-test value of 0.0045 and an accuracy value of 96.5% (Table 1).

5.1. Comparison with reconstructed ground truth

As explained in Section 3.3, the result provided by our pipeline is obtained by subtracting the CNN prediction from the original input image, allowing us to retrieve the high frequencies, which had been lost due to the bilateral filter. It follows that even for a loss function $L(y, x, o) = 0$ the exact ground truth can never be reconstructed. Furthermore, and more importantly, the misalignments due to pose changes between the flash and non flash photos would dominate when computing the input and output image differences. For these reasons, we introduce a preconditioning operator on the ground truth as:

$$\overline{o}_i = x_i - 2t_i + 1 \tag{8}$$

where t_i is the target difference, as explained in Section 3.4. Fig. 5 shows the final result for some examples of the validation data, while Figs. 6 and 7 show, similarly, the reconstruction of example images belonging to the training and test sets, respectively. In particular, it can be seen in Figs. 5 and 7 that the information lost because of the flash, such as hairs, beard and skin color, are retrieved through the CNN's prediction. The shadow cast by the subject is also corrected by the CNN to reduce the highlights due to the camera flash. By reconstructing the final image through the non-filtered input, we retrieved the high frequencies resolving the blur problem. We evaluated the data using the Structural Similarity Index (SSIM) [57] for each subset of the dataset. In particular, as can be seen from Figs. 5–7, which show comparisons between ground truth and the reconstructed predictions, the SSIM average value is around 80% for the validation set, around 92% for the test set, and around 94.5% for the training set. As a final step, we run a Red Eye Removal filter with GIMP to present the final results only on the test set images for aesthetic reasons; see Fig. 7. Please note that all metrics were run on images without removing the red-eye artifact (see Fig. 8).

Table 1
Loss validation and loss test after 62 epochs. This table also shows the maximum accuracy achieved in both phases.

	Loss	Accuracy
Validation	0.0042	96.2%
Test	0.0045	96.5%

5.2. Comparison with HDRNet

We compared our approach to HDRNet by Gharbi et al. [58]. This work is based on the use of a CNN inspired to the bilateral grid processing [59] and to local affine color transforms. HDRNet is designed to learn any image operator and hence is a suitable candidate to remove flash artifacts from photographs. In a first experiment, we trained HDRNet end-to-end with our dataset, by feeding the network

Fig. 5. Samples of validation data. Each group of images is composed of: the original image taken with the smartphone flash (top left); the flash image to which the bilateral filter was applied (bottom left); the image reconstructed by the difference prediction of the CNN (center); the ground truth reconstructed (top right); the ground truth (bottom right). The SSIM was computed by comparing the central image of each group and the image at top right.

Fig. 6. Training set example. For each row: the original input (left); the image reconstructed by CNN prediction (center); the ground truth reconstructed (right). The SSIM was computed by comparing the center images and the right images.

Fig. 7. Test set example. For each row: the original input (left); the image reconstructed by CNN prediction (center); the ground truth reconstructed (right). The SSIM was computed by comparing the center images and the right images. The SSIM was computed on the images without removing the red-eye artifact.

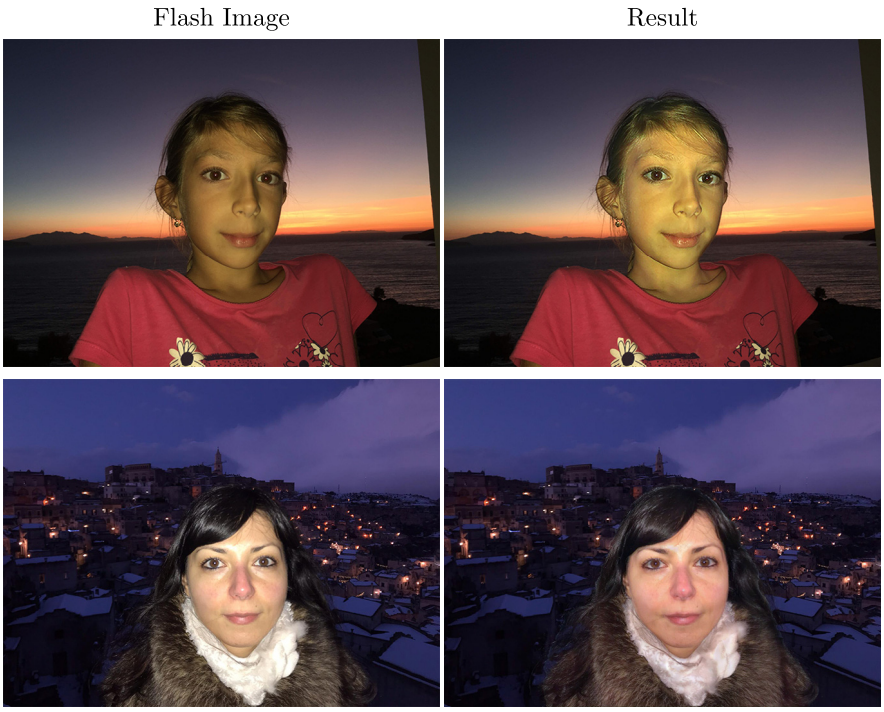

Fig. 8. Two example of a real images. The subjects were detected using Photoshop Subject Detection tool and were forwarded to our neural network with a green background. The results were blended with the original images.

with input and target images. We used the same parameters, times and training algorithm proposed by the authors and trained the network for 48 h. We obtained a stable loss value of 0.0031.

Fig. 9 shows the comparison with our approach. Although HDRNet is capable of approximating the colors of the ground truth image, the flash highlights remain substantially unchanged and the blurring is extremely high making the face unrecognizable. Note that this is also a consequence of the input and output image misalignments for which we introduced the preconditioning operator explained at the beginning of Section 5.1. On the contrary, our result more closely matches the studio portrait preserving the high frequencies of the images such as hair and face traits.

In a second experiment, we combined HDRNet with our encoding, that is, training HDRNet with the filtered images as input and the difference between the input and the filtered ground truth as target (see Section 3.3). Even in this experiment, we used the parameters proposed by the authors and trained the network for over 48 h, obtaining a stable loss value of 0.0007. A few result samples of this experiment are shown in Fig. 10. From this figure, we can notice that HDRNet performance has dramatically improved by using our encoding. However, our full approach (that is, our encoding on our network) does a better job at removing the flash artifacts (i.e., highlight and shadows).

We compared the two methods by using SSIM and the Peak Signal to Noise Ratio (PSNR) between the prediction of networks and the target images obtained a random sample of 30 images from each sub-set of the dataset (training set, validation set, and test set). The results are reported in Table 2. From this table, we can see that our approach results in higher SSIM and PSNR values for all subsets.

We use the SSIM [57] index to compare the images because such metric can measure the similarity between two images in a way that is consistent with human eye perception. In addition, we use also the PSNR, because it is an approach that estimates the absolute error between two images. Furthermore, such metric is often used to compare the quality of reconstructed images, as in this case; i.e., the CNN output.

Table 2
Comparisons on samples of the training, validation, and test sets. In particular, the SSIM and PSNR values are computed on samples of 30 images extracted randomly from each dataset.

	Our SSIM	HDRNet SSIM	Our PSNR	HDRNet PNSR
Training	**91.55%**	73.93%	**24.05 dB**	19.71 dB
Validation	**87.76%**	78.51%	**20.42 dB**	18.84 dB
Test	**89.80%**	82.92%	**21.28 dB**	19.02 dB

Summarizing, we can claim both that our approach outperforms HDRNet w.r.t. the specific image operator that removes the flash artifacts, and that HDRNet benefits from our encoding strategy by producing better results than its native end-to-end configuration.

5.3. Comparison with Pix2Pix

We also compared our approach against Pix2Pix by Isola et al. [60]. Such work is based on a particular type of Generative Adversarial Network (GAN) [61] in the conditional setting (cGAN) [62]. The use of such type of neural network was investigated as a general-purpose solution to image-to-image translation problems. Isola et al. tested their cGAN on several tasks such as photo generation and semantic segmentation.

In the training details, they explained that the images were randomly jittered through resizing the 256 × 256 (original size of images) to 286 × 286 and then randomly cropped to come back to 256 × 256. Therefore, we trained Pix2Pix cGAN by using the training information provided for the $Day \rightarrow Night$ task, which is performed by the authors. In particular, the network was trained using our dataset; i.e., 4 as batch size and 80 as the number of epochs. We had to increase the number of epochs because 17 epochs, as in the case of the $Day \rightarrow Night$ task proposed by Isola et al. produced low quality results.

The network was trained from scratch by using a Gaussian distribution to initialize the weights with 0 as mean and 0.02 as standard deviation, as suggested by the authors. We performed a mirroring and the random jitter starting from our image resolution 512 × 512 and doubling the resizing to 572 × 572.

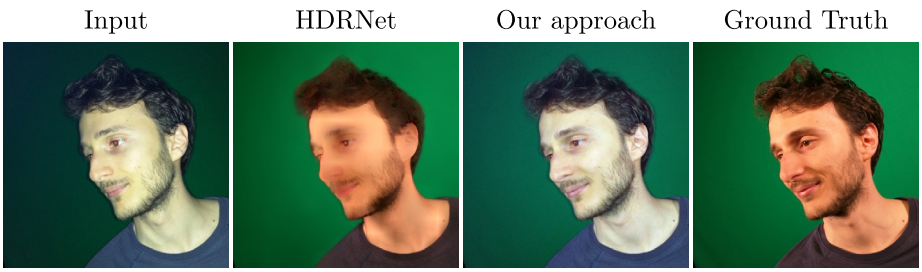

Fig. 9. A comparison between HDRNet end-to-end training and our approach.

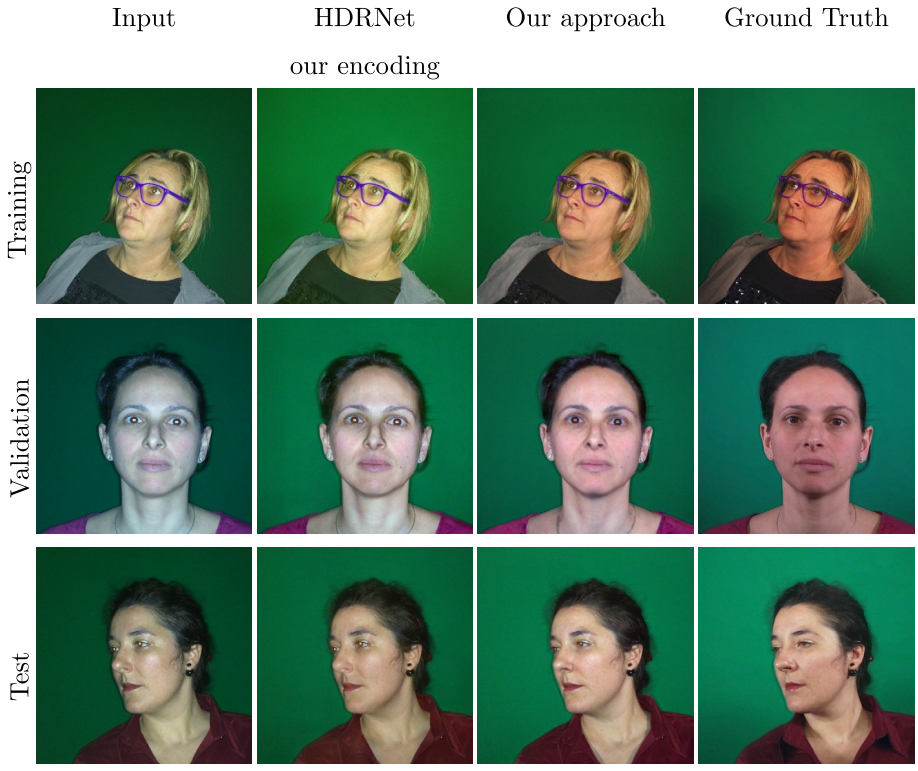

Fig. 10. An example of comparisons between HDRNet (combined with our problem encoding) and our approach. We show one example from the training set, the validation set and the test set, respectively.

As the previous comparison (see Section 5.2), we performed two experiments. In the first one, we trained Pix2Pix end-to-end using our dataset by feeding the network with input and target images. The network was trained for about 9 h. Fig. 11 shows the comparison with our approach. Although Pix2Pix better approximates the ground truth image colors, it introduces notable artifacts by changing the image content significantly. Many of these artifacts can be seen on eyes and facial features.

In the second experiment, we combined Pix2Pix with our encoding. We trained Pix2Pix with the filtered images as input and the difference between the input and ground truth filtered as a target; see Section 3.3. In this case, the network was trained for 9 h with the same parameters used in the first case. Fig. 12 shows some outcomes of this experiment. As the previous comparison, it is possible to notice a dramatic improvement of the results obtained by combining Pix2Pix and our encoding. However, artifacts are still present on the geometry of the image (i.e., facial features and around eyes), even though they are not as strong as in the end-to-end training.

The results of the comparisons are visible in Table 3, which reports the SSIM and PSNR values. As the previous comparison, these values are computed on a random sample of 30 images for each subset of the dataset (i.e., training set, validation set, and test set). From this table, it can be seen that our approach obtains higher values of SSIM and PSNR for all subsets.

Table 3
Comparisons on samples of the training, validation, and test sets. As before, the SSIM and PSNR values are computed on samples of 30 images each, which were extracted randomly from each dataset.

	Our SSIM	Pix2Pix SSIM	Our PSNR	Pix2Pix PNSR
Training	**90%**	71%	**24.8 dB**	17.82 dB
Validation	**83.16%**	73.16%	**19.72 dB**	17.64 dB
Test	**88.78%**	74.25%	**20.58 dB**	17.30 dB

Finally, this comparison elicits that our approach outperforms Pix2Pix w.r.t the image operator that remove the flash artifacts. Furthermore, Pix2Pix can benefit from our encoding strategy by generating high-quality results compared to end-to-end training.

5.4. Comparisons with style transfer

As our final comparison, we tested our method against the style transfer method by Shih et al. [3]. This method is specifically meant for portraits and based on a multi-scale local transfer approach. In our tests, we used the ground-truth image as target style to be transferred

Fig. 11. A comparison between Pix2pix end-to-end training and our approach.

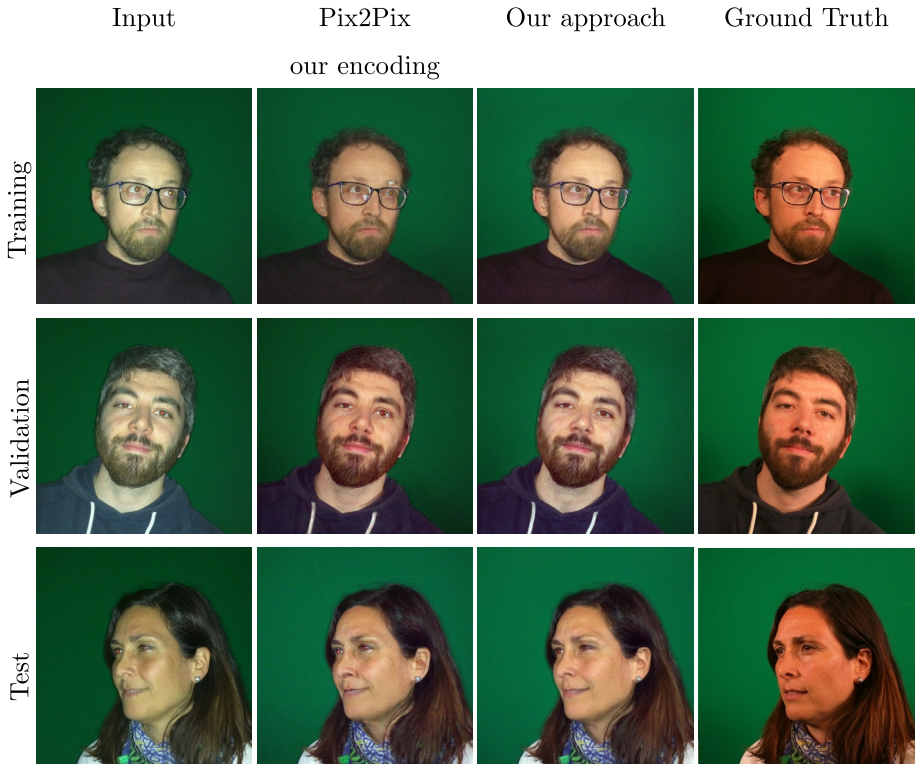

Fig. 12. An example of comparisons between Pix2Pix (with our problem encoding) and our approach. We show one example from the training set, the validation set and the test set, respectively.

to the input image that is the ideal condition. Unfortunately, we could not try Shih et al.'s method on a large dataset because the generation of masks and landmarks for the original code is extremely cumbersome (i.e., more than an hour per image). Therefore, we tested on a limited number of images that are displayed on Fig. 13. As can be seen from Fig. 13, the method by Shih et al. can transfer colors correctly, but it fails to remove flash artifacts. Moreover, these are enhanced creating unnatural effects especially for eyes and hard shadows and lighting.

6. Limitations

Thanks to the tightly bounded domain of the input, our system can provide convincing results even after being trained with a small dataset of 495 input images (prior to data augmentation). However, results would greatly improve if a more diverse training dataset was provided by relaxing some boundaries of the said domain. More specifically: all the subjects were white adults, the uniform-light setting was the same for all images, and all images were acquired with the same device.

7. Conclusions

We have proposed an unassisted pipeline to turn a smartphone flash selfie into a studio portrait by using a regression model based on supervised learning. We have defined a complete pipeline, starting from data collected by well-defined acquisition parameters, performing pre-processing by a bilateral filter, training the network, and finally, validating the results.

We have made several comparisons using different metrics to validate our approach. Among these, we used the SSIM that places more emphasis on the validity of the results because it measures the similarity between images in a way that is consistent with the perception of the human eye. Besides the obvious application of our method for correcting flash selfies, our results allow us to conjecture that a low-quality smartphone flash selfie contains enough information for reconstructing the actual appearance of a human face as one obtained with more uniform lighting.

The most likely future work will be to widen the acquisition domain, both in terms of hardware and illumination settings and in terms of the age and ethnicity of photographed subjects. Then, we will build on our method by incorporating state-of-the-art solutions for multiple-face detection, red-eye removal, and background subtraction. Our aim is to deploy a mobile app that can be used by any smartphone user.

On the method itself, we are planning to explore the solution proposed by Larsen et al. [63] and to use our network as the generator component of a GAN [61] where the discriminator is trained to

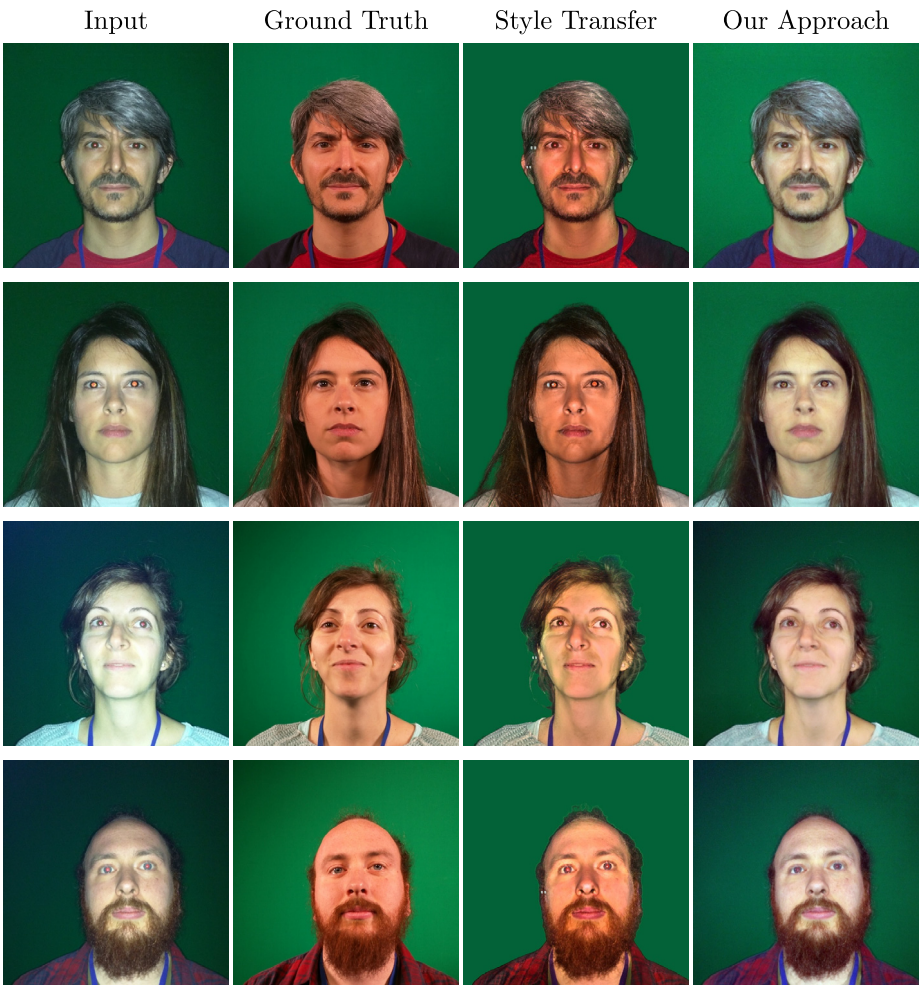

Fig. 13. An example of visual comparisons between the Portrait Style transfer [3] and our approach. Although there is a good match in colors several artifacts are presents such as flash hard shadows and around eyes.

recognize if a given image is a difference between the bilateral filtered version of a flash and no flash images. To improve our CNN encoder–decoder architecture as a future work, we could use ResNet or other models as encoder component [64] of our CNN as mentioned in recent works [65,66].

Acknowledgment

The authors thank NVIDIA's Academic Research Team for providing the Titan Xp card under the Hardware Donation Program. We wish also to thank Pietro Cignoni for building the hardware driving the flash lighting and the many colleagues of the Visual Computing Lab and of the whole ISTI-CNR who helped us in building up the photo dataset; in particular Mauro Boni and Giuliano Kraft for providing skilled technical assistance in setting up the light studio.

References

[1] V. Bychkovsky, S. Paris, E. Chan, F. Durand, Learning photographic global tonal adjustment with a database of input/output image pairs, in: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE, 2011, pp. 97–104.

[2] S. Wang, J. Zheng, H.-M. Hu, B. Li, Naturalness preserved enhancement algorithm for non-uniform illumination images, IEEE Trans. Image Process. 22 (9) (2013) 3538–3548.

[3] Y. Shih, S. Paris, C. Barnes, W.T. Freeman, F. Durand, Style transfer for headshot portraits, ACM Trans. Graph. 33 (4) (2014) 148.

[4] Y. Wang, L. Zhang, Z. Liu, G. Hua, Z. Wen, Z. Zhang, D. Samaras, Face relighting from a single image under arbitrary unknown lighting conditions, IEEE Trans. Pattern Anal. Mach. Intell. 31 (11) (2009) 1968–1984.

[5] Z. Wen, Z. Liu, T.S. Huang, Face relighting with radiance environment maps, in: Null, IEEE, 2003, p. 158.

[6] Z. Shu, S. Hadap, E. Shechtman, K. Sunkavalli, S. Paris, D. Samaras, Portrait lighting transfer using a mass transport approach, ACM Trans. Graph. 37 (1) (2018) 2.

[7] E. Gabriel, K. Joel, D. Gyorgy, M. Rafał, U. Jonas, HDR Image reconstruction from a single exposure using deep CNNs, ACM Trans. Graph. 36 (6) (2017).

[8] S. Iizuka, E. Simo-Serra, H. Ishikawa, Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification, ACM Trans. Graph. 35 (4) (2016) 110:1–110:11, http://dx.doi.org/10.1145/2897824.2925974.

[9] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, W. Shi, Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 105–114, http://dx.doi.org/10.1109/CVPR.201719.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, Ieee, 2009, pp. 248–255.

[11] C. Sun, A. Shrivastava, S. Singh, A. Gupta, Revisiting unreasonable effectiveness of data in deep learning era, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 843–852.

[12] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, K. Toyama, Digital photography with flash and no-flash image pairs, ACM Trans. Graph. 23 (3) (2004) 664–672, http://dx.doi.org/10.1145/1015706.1015777.

[13] E. Eisemann, F. Durand, Flash photography enhancement via intrinsic relighting, ACM Trans. Graph. 23 (3) (2004) 673–678, http://dx.doi.org/10.1145/1015706.1015778.

[14] A. Agrawal, R. Raskar, S.K. Nayar, Y. Li, Removing photography artifacts using gradient projection and flash-exposure sampling, ACM Trans. Graph. 24 (3) (2005) 828–835, http://dx.doi.org/10.1145/1073204.1073269.

[15] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[16] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533.

[17] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, in: NIPS'12, Curran Associates Inc., USA, 2012, pp. 1097–1105.

[18] A. Dosovitskiy, J.T. Springenberg, M. Riedmiller, T. Brox, Discriminative unsupervised feature learning with convolutional neural networks, in: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1, in: NIPS'14, MIT Press, Cambridge, MA, USA, 2014, pp. 766–774.

[19] Z. Shu, E. Yumer, S. Hadap, K. Sunkavalli, E. Shechtman, D. Samaras, Neural face editing with intrinsic image disentangling, in: Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, IEEE, 2017, pp. 5444–5453.

[20] S. Sengupta, A. Kanazawa, C.D. Castillo, D.W. Jacobs, Sfsnet: Learning shape, reflectance and illuminance of faces in the wild, in: Computer Vision and Pattern Regognition (CVPR), 2018.

[21] A. Ignatov, N. Kobyshev, R. Timofte, K. Vanhoey, L.V. Gool, DSLR-Quality Photos on Mobile Devices with Deep Convolutional Networks, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 3297–3305, http://dx.doi.org/10.1109/ICCV.2017.355.

[22] Y.-S. Chen, Y.-C. Wang, M.-H. Kao, Y.-Y. Chuang, Deep photo enhancer: Unpaired learning for image enhancement from photographs with gans, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2018, pp. 6306–6314.

[23] Y. Hu, H. He, C. Xu, B. Wang, S. Lin, Exposure: A white-box photo post-processing framework, ACM Trans. Graph. 37 (2) (2018) 26.

[24] J.-Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017, arXiv preprint.

[25] X. Shen, A. Hertzmann, J. Jia, S. Paris, B. Price, E. Shechtman, I. Sachs, Automatic portrait segmentation for image stylization, Comput. Graph. Forum 35 (2) (2016) 93–102, http://dx.doi.org/10.1111/cgf.12814.

[26] E. Shelhamer, J. Long, T. Darrell, Fully convolutional networks for semantic segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (4) (2017) 640–651, http://dx.doi.org/10.1109/TPAMI.2016.2572683.

[27] X. Zhang, R. Ng, Q. Chen, Single Image Reflection Separation with Perceptual Losses, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[28] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2015, pp. 234–241.

[29] C. Chen, Q. Chen, J. Xu, V. Koltun, Learning to See in the Dark, 2018.

[30] Y. Aksoy, C. Kim, P. Kellnhofer, S. Paris, M. Elgharib, M. Pollefeys, W. Matusik, A Dataset of Flash and Ambient Illumination Pairs from the Crowd, in: Proc. ECCV, 2018.

[31] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, CoRR arXiv:abs/1409.1556.

[32] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: G. Gordon, D. Dunson, M. Dudík (Eds.), Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, Vol. 15, PMLR, Fort Lauderdale, FL, USA, 2011, pp. 315–323.

[33] Y.T. Zhou, R. Chellappa, Computation of optical flow using a neural network, in: IEEE 1988 International Conference on Neural Networks, Vol. 2, 1988, pp. 71–78, http://dx.doi.org/10.1109/ICNN.1988.23914.

[34] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep learning, 1, MIT press Cambridge, 2016.

[35] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015, arXiv preprint arXiv:1502.03167.

[36] A.L. Maas, A.Y. Hannun, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: ICML Workshop on Deep Learning for Audio, Speech and Language Processing, 2013.

[37] B. Xu, N. Wang, T. Chen, M. Li, Empirical evaluation of rectified activations in convolutional network, 2015, CoRR arXiv:abs/1505.00853.

[38] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[39] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778, http://dx.doi.org/10.1109/CVPR.2016.90.

[40] N. Capece, U. Erra, R. Scolamiero, Converting Night-Time Images to Day-Time Images through a Deep Learning Approach in: 2017 21st International Conference Information Visualisation (IV), 2017, pp. 324–331, http://dx.doi.org/10.1109/iV.2017.16.

[41] M.D. Zeiler, D. Krishnan, G.W. Taylor, R. Fergus, Deconvolutional networks, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2010, pp. 2528–2535, http://dx.doi.org/10.1109/CVPR.2010.5539957.

[42] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[43] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (Jul) (2011) 2121–2159.

[44] T. Tieleman, G. Hinton, Lecture 6.5-rmsprop, COURSERA: Neural networks for machine learning, in: Technical Report, University of Toronto, 2012.

[45] O. M. Parkhi, A. Vedaldi, A. Zisserman, Deep Face Recognition, Vol. 1, 2015, pp. 41.1–41.12, http://dx.doi.org/10.5244/C.29.41.

[46] J. Burkardt, The truncated normal distribution, Department of Scientific Computing Website, Florida State University, 2014.

[47] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.

[48] F. Banterle, M. Corsini, P. Cignoni, R. Scopigno, A low-memory, straightforward and fast bilateral filter through subsampling in spatial domain, Comput. Graph. Forum 31 (1) (2012) 19–32.

[49] F. Durand, J. Dorsey, Fast bilateral filtering for the display of high-dynamic-range images, ACM Trans. Graph. 21 (3) (2002) 257–266, http://dx.doi.org/10.1145/566654.566574.

[50] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, in: NIPS'12, Curran Associates Inc., USA, 2012, pp. 1097–1105.

[51] Y. Sun, X. Wang, X. Tang, Deeply learned face representations are sparse, selective, and robust, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2892–2900.

[52] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the 27th International Conference on International Conference on Machine Learning, in: ICML'10, Omnipress, USA, 2010, pp. 807–814.

[53] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.

[54] S. Raghunathan, D. Stredney, P. Schmalbrock, B.D. Clymer, Image registration using rigid registration and maximization of mutual information, in: Poster Presented At: MMVR13. the 13th Annual Medicine Meets Virtual Reality Conference, 2005.

[55] D. Mattes, D.R. Haynor, H. Vesselle, T.K. Lewellyn, W. Eubank, Nonrigid multimodality image registration, in: Medical Imaging 2001: Image Processing, Vol. 4322, International Society for Optics and Photonics, 2001, pp. 1609–1621.

[56] M. Styner, C. Brechbuhler, G. Szckely, G. Gerig, Parametric estimate of intensity inhomogeneities applied to MRI, IEEE Trans. Med. Imaging 19 (3) (2000) 153–165, http://dx.doi.org/10.1109/42.845174.

[57] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612, http://dx.doi.org/10.1109/TIP.2003.819861.

[58] M. Gharbi, J. Chen, J.T. Barron, S.W. Hasinoff, F. Durand, Deep bilateral learning for real-time image enhancement, ACM Trans. Graph. 36 (4) (2017) 118.

[59] J. Chen, S. Paris, F. Durand, Real-time edge-aware image processing with the bilateral grid, ACM Trans. Graph. 26 (3) (2007) http://dx.doi.org/10.1145/1276377.1276506.

[60] P. Isola, J.-Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks, CVPR (2017).

[61] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.

[62] M. Mirza, S. Osindero, Conditional generative adversarial nets, 2014, arXiv preprint arXiv:1411.1784.

[63] A.B.L. Larsen, S.K. Sønderby, H. Larochelle, O. Winther, Autoencoding beyond pixels using a learned similarity metric, in: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, in: ICML'16, JMLR.org, 2016, pp. 1558–1566.

[64] A. Chaurasia, E. Culurciello, Linknet: Exploiting encoder representations for efficient semantic segmentation, in: 2017 IEEE Visual Communications and Image Processing (VCIP), IEEE, 2017, pp. 1–4.

[65] A.A. Shvets, V.I. Iglovikov, A. Rakhlin, A.A. Kalinin, Angiodysplasia detection and localization using deep convolutional neural networks, in: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2018, pp. 612–617.

[66] V. Iglovikov, A. Shvets, Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation, 2018, arXiv preprint arXiv:1801.05746.