TreEnhance: A Tree Search Method For Low-Light Image Enhancement

Marco Cotogni∗, Claudio Cusano
*Dep. of Electrical, Computer and Biomedical Engineering, University of Pavia, Via Ferrata 1, Pavia, 27100, Italy

Abstract

In this paper we present TreEnhance, an automatic method for low-light image enhancement capable of improving the quality of digital images. The method combines tree search theory, and in particular the Monte Carlo Tree Search (MCTS) algorithm, with deep reinforcement learning. Given as input a low-light image, TreEnhance produces as output its enhanced version together with the sequence of image editing operations used to obtain it. During the training phase, the method repeatedly alternates two main phases: a generation phase, where a modified version of MCTS explores the space of image editing operations and selects the most promising sequence, and an optimization phase, where the parameters of a neural network, implementing the enhancement policy, are updated.

Two different inference solutions are proposed for the enhancement of new images: one is based on MCTS and is more accurate but more time and memory consuming; the other directly applies the learned policy and is faster but slightly less precise. As a further contribution, we propose a guided search strategy that “reverses” the enhancement procedure that a photo editor applied to a given input image.

Unlike other methods from the state of the art, TreEnhance does not pose any constraint on the image resolution and can be used in a variety of scenarios with minimal tuning. We tested the method on two datasets: the Low-Light dataset and the Adobe Five-K dataset obtaining good results from both a qualitative and a quantitative point of view.

Keywords: Low-Light Image Enhancement, Deep Reinforcement Learning, Automatic Image Retouching, Image processing, Tree Search

1. Introduction

Image enhancement is a well-known image processing problem that consists in improving the appearance of a low quality image to obtain a high quality version without changing the content. Nowadays, Deep Convolutional Neural Networks are the methods of choice for many image processing tasks such as image enhancement, super resolution, image inpainting, etc. For this reason, neural networks are now part of many of the image processing tools used by professional photographers to post-process their pictures. However, the use of this kind of tools requires a certain level of experience and knowledge, making desirable to have both semi-automatic and fully automatic procedures.

Among neural models, image-to-image translation methods [1, 2, 3] have been shown to be very effective for image enhancement. However, they suffer of two major problems: the presence of artifacts their output images, and the limitation on the resolution of the input that is often imposed by the architecture of the neural network. Among these translation approaches, recent methods based on diffusion models seem able to produce high-quality images without artifacts [4, 5, 6, 7]. These methods repeat many generation steps, preserving every time the spatial resolution of the image. As a consequence, their computational cost can be prohibitive when generating medium- or high-resolution images.

Most recent methods work as black boxes, without providing any insight about the transformation found by the neural network. For a professional usage, it is of paramount importance that the method provides an “explainable” solution that makes it understandable and controllable.
possible for the user to verify it and possibly to adjust it to his or her needs. Possible solutions to provide an explanation to the neural networks’ predictions were explored in the recent years [8, 9]. In these works the authors proposed methods to analyze the predictions obtained by the model and obtain some explanation behind the networks’ reasoning.

On the basis of these considerations, we propose here TreEnhance, a low-light image enhancement method based on tree search and deep reinforcement learning. The method is able to obtain high quality output images with the application of a sequence of image editing operations. The sequence selected by the method is fully explainable, making it suitable not only for automatic enhancement, but also as a tool for a variety of interactive scenarios. For instance, it could be used for educational purposes to teach beginners how to edit their own images.

The method does not pose any constraint on the resolution of input images. In fact it can deal with both high- and low-resolution images without any modification. We tested our method on two different datasets tuning only the parameters of the image editing operations.

The main contributions presented in this paper are:

- To the best of our knowledge, this is the first work presenting the application of tree-search theory to image enhancement problems.
- Differently from several previous methods in the state of the art, TreEnhance acts as a fully explainable neural architecture for image enhancement. This method provides along with the resulting output image the sequence of enhancing operators selected by the algorithm to enhance the low quality image. The images provided as output are good from a qualitative and quantitative points of view without the introduction of any artifact and without constraints on the image resolution.
- TreEnhance includes two different inference strategies for enhance low quality image based on the data, time an memory constraints.
- A guided search procedure that can be used as a tool for “reverse engineering”. It decomposes into simple operations the enhancement process selected by an expert photographer. This tool could be included in an educational software to teach beginners how to enhance their own photographs.

The paper is organized as follows: we begin with the presentation of the state of the art in image enhancement in Section 2. In Section 3 we explain how TreEnhance works, what are the steps of the method and which neural architecture it uses. In Section 4 we describe the datasets used for the experiments and the results obtained. Finally we conclude the paper with Section 5 and Section 6 where we discuss the results obtained and possible future investigations on this topic. Code is available here: https://github.com/OcraM17/TreEnhance

2. Related Work

In recent years, many image enhancement methods have been proposed. Among these, one of the most effective approaches is that represented by image-to-image translation methods. In this family of methods, a neural network learns how to directly convert an image from one domain to another (i.e. low quality to high quality), while preserving its visual content. These methods learn by observing thousands of pairs of low-quality/high-quality images.

Ravirathanam et al. proposed a multi-context framework based on a modified version of the UNet architecture for low-light image enhancement [10]. In this work the authors combined a perceptual loss, a structural loss and a patch-wise euclidean loss to enhance a low-light input image.

Xu et al. presented a decomposition and enhancement method for low-light image enhancement working in the frequency domain. The architecture presented learns how to recover image objects low-frequency layers. Once these image objects have been recovered, the method is able to enhance high-frequency image details.

Ignatov et al. presented an image-to-image translation pipeline for enhancing smartphone pictures [3]. In this work the ground truth images were obtained with a professional camera. In order to deal with misalignment problems due to the different resolutions of the cameras, the authors trained the neural model with aligned patches obtained from the original images.

Pix2Pix [1], Cycle-GAN [2] and EnlightenGAN [11] are methods based on an adversarial learning
schema which involves a generator and a discriminator. In a image-to-image translation scenario, the generator learns how to produce images very similar to the training ones, the discriminator learns how to distinguish real training images from generated images. The optimal point is reached when the discriminator is not able to distinguish the images in the training from those produced by the generator. These works show very good performance in several computer vision tasks including image enhancement.

Lv et al. proposed a method based on local features extractors for low light images [12]. Lore et al. proposed a deep autoencoder approach for natural low-light image enhancement. [13]

Zhang proposed a pipeline based on decomposition network and illumination adjustment to tune the exposure in low-light images [14]. The same approach has been further explored by other research groups in the following years [15, 16].

Cai et al. presented a Pixel-level Noise-aware Generative Adversarial Network able to generate very realistic noisy images. By fine-tuning different denoising methods, they were able to reach state-of-the-art performance. This confirms the ability of adversarial networks in generating a very realistic noise in terms of distribution and intensity [17].

In the last years, methods based on diffusion models reached state-of-the-art performance in image generation, with the additional feature of limiting the artifacts in the output. Compared with GANs, diffusion models are more stable and the images produced obtained a higher FID [18]. Saharia et al. proposed a conditional diffusion model and applied it to four different image-to-image translation tasks: colorization, JPEG restoration, inpainting and uncropping [4]. Batzolis et al. proposed a conditional diffusion model able to reach state-of-the-art performance in inpainting, super-resolution and edge-to-image tasks [7]. Similarly to Cycle-GAN, Sasaki et al. proposed an unpaired approach with denoising diffusion probabilistic models for image-to-image translation [6].

Despite their good performance, the application of diffusion models to the enhancement of low-light images is still unexplored. Moreover, according to Dhariwal et al. [18], the computational resources required to train and to use this kind of methods are very high, due to the many refinement steps in the generation process and the necessity of keeping at each step the desired spatial resolution of the images.

Differently from image-to-image translation methods, parametric methods do not model directly the mapping between low-quality and high-quality images but they learn instead the parameters of the color transformation to be applied to the low-quality image.

Bianco et al. presented two works: in the first one, a neural network learns the parameters of a color transformation. This transformation is applied to a filtered image to recover its original color curve. In the second one, the parameters of a color transformation are combined with a basis function to enhance the visual content of low quality images [19, 20]. Another work from the same authors explores the use of a neural network to estimate the coefficients of splines, that are the used as color curves [21].

Zhang et al. proposed an Exposure Correction Network able to estimate the best S-shaped curve (a non linear curve used to adjust the exposure of shadow-tone areas in images) to restore low-light images. This curve is then applied to the low quality input image to enhance it [22].

Chai et al. presented an approach for parametric color enhancement. In this work, a convolutional neural network learns the parameters of a quadratic color transformation in a supervised learning scenario [23].

Kim et al. proposed a method based on representative color transformations. This method uses local and global enhancement modules to determine the most representative colors in input images and to estimate the transformation for these colors. Then, the model defines the enhanced colors by using the transformed colors. The criteria used to define the enhanced colors is based on the similarity between representative and input colors [24].

Guo et al. presented a zero-reference method for low-light image enhancement. In this work, a deep curve estimator network takes as input a low-light image and it estimates a series of light-enhancement curves. These curves are then iteratively applied to the RGB input image obtaining the enhanced image. In order to train the estimator in a zero-reference scenario, four differentiable non-reference losses were involved [25].

Another kind of related approaches, are those based on reinforcement learning. In this family of methods an agent selects the enhancement operators to be applied to a low quality image to obtain its enhanced version. During the training a numer-
Park et al. presented a deep q-learning based approach for image enhancement. The reward is modeled as the difference between the distance of the image from the ground truth before and after the application of the operator [26].

Yang et al. proposed a method based on Markov Decision Process for real-time exposure control. In this work, given the current frame to the agent, a fully convolutional neural network is trained by using the Gaussian policy gradient algorithm. The method is able to optimize the trade-off among convergence, minimal temporal oscillation and quality of the produced image [27].

Hu et al. proposed a white box-approach for image enhancement. This work is based on an actor-critic algorithm to enhance the content of a low quality image [28].

Yu et al. presented a deep q-learning based toolchain for image restoration. In this work, a neural network learns what are the most suitable restoration operators that should be applied to a corrupted image to restore its content [29].

Yu et al. proposed a mixed approach based on GANs and deep reinforcement learning [30].

The use of vision transformers for color transition problems has been proposed by Cai et al. [31] and by Lin et al. [32]. The former proposed a spectral-wise Multi-head Self-Attention block for hyperspectral image reconstruction within a masked transformer. The latter presents a transformer-based method that uses a coarse patch selection in combination with a fine pixel clustering for the reconstruction of hyperspectral images.

Finally, a new approach presented by Zhang et al. is based on a transformer architecture for enhancing low quality images. This algorithm, divides the images in patch, applies an embedding and passes the created vectors through several two-branches transformer modules obtaining the enhanced image [33].

3. Method

TreEnhance is a tree-based deep reinforcement learning method that finds sequences of image editing operations to improve the quality of digital photographs. It does so by emulating the behavior of a professional photo editor who selects the operations among those available in the software he or she uses. The selection follows a trial-and-error strategy in which multiple operations (such as gamma correction, contrast adjustment, etc.) are repeatedly tried, verified, canceled, and combined. This way many different sequences of operations are evaluated and the most promising one is taken.

More precisely, given an image \( x \) the method selects the editing operations \( a_1, \ldots, a_n \) from a finite set \( A \) of operations, and apply them to produce the enhanced image \( \hat{y} \):

\[
\hat{y} = a_n(a_{n-1}(\ldots a_1(x))).
\]  

The sequence is terminated by a special STOP operation.

To select the operations, the method builds a search tree that explores a large number of candidate sequences. A convolutional neural network guides the growth of the tree by promoting the exploration of the operations that are more likely to contribute to the improvement of the image. The network also evaluates the images obtained during the process so that the best one can be selected.

The network is trained in a supervised way with the goal of obtaining an enhanced image \( \hat{y} \) as close as possible to a target image \( y \) manually edited by a human expert. The following sections present the method in details.

3.1. Training Procedure

A set of input/target image pairs is used to train the neural network. The training procedure alternates two phases: generation and optimization. In the generation phase a revised version of the Monte Carlo Tree Search (MCTS) algorithm is used to grow trees associated to a random selection of the available images pairs. In the optimization phase the images obtained in the generation phase are used to train the parameters of the neural network.

3.1.1. Generation

starting from the root node, which represents the input image, the method selects an image editing operation (an action) which results in a new image (i.e. a new node). Nodes and actions form a tree which is built by a sequence of iterations, each one adding a new node. A numerical score, called return is associated to each node. The goal is to find actions that are likely to lead to nodes associated to high returns. Each iteration of MCTS includes four major steps [34]:

1. Selection: a tree policy is applied to select a leaf node.
2. Expansion: new child nodes (one for each possible action) are attached to the selected leaf. They will all become new leaves of the tree.

3. Evaluation: the return is assigned to the node.

4. Backup: the return is propagated from the selected node up to the root of the tree.

In the original MCTS the evaluation step consists in the maximization of the Upper Confidence Bound (UCB) which accounts for both the exploitation of the less frequently visited nodes and the exploration of the best operation that produced a high return.

The selection step starts from the root and descends the tree by choosing the nodes with the highest bound until a leaf is reached. As a further mechanism to promote exploration we add a small amount of Dirichlet noise to the policy. When a terminal node is found, it is directly evaluated instead of being processed by the neural network. This happens in two cases: when a maximum depth is reached, and when the special STOP operation is selected. In the first case the search fails and the return is just zero. In the second case the return depends on the distance between the resulting image $x$ and the target image $y$:

$$r(x) = \exp(-\alpha \|x - y\|^2), \quad (3)$$

where $\alpha$ is a tunable parameter.

When a set number of MCTS iteration is reached, a policy $\rho$ for the root node is obtained.

$$\rho(x, a) = \frac{N(a(x))}{N(x)}. \quad (4)$$

This policy represents the probability distribution of the root’s children visits. From this probability distribution a random sample is extracted and the corresponding action is applied to the root image. This ensures a good degree of exploration by allowing the generation of multiple sequences of operations for the same image. From this new image (that becomes the new root of the tree) the complete procedure restarts and it is repeated until a terminal node is reached. The upper part of Figure 1 depicts the generation phase of TreEnhance.

3.1.2. Optimization

The roots visited during the generation phase are used to build a training set. Each training element is a triplet combining (i) the image $x$, (ii) the return $r(x)$, and (iii) the probabilities of the stochastic policy $\rho(x, \cdot)$. The parameters of the neural architecture are updated by minimizing the loss function $L$:

$$L = \lambda (r(x) - v(x))^2 - \sum_{a \in A} \rho(x, a) \log \pi(x, a), \quad (5)$$

which combines the squared estimation error of the return with the cross entropy measuring the disagreement between the stochastic policy $\rho$ and the network policy $\pi$. The parameter $\lambda$ is a tunable hyperparameter used to balance the weight of the two terms in the loss.

The whole process is repeated multiple times, each time producing more training data in the generation phase and improving the neural network in the optimization phase.

3.2. Neural Architecture

The convolutional neural network used in this work is a modified version of the ResNet-18 where the last fully connected classification layer is replaced by two feed forward heads. Both include
Figure 1: TreEnhance generation (top) and optimization (bottom) phases. To make the figure more clear only part of the tree is actually shown. For the complete architecture of the ResNet-18, please refer to the original paper. The function $\pi(x, \cdot)$ is the policy over the editing operations and $v(x)$ is the estimated return.

two fully connected linear layers. The first head also includes the softmax function to produce the vector representing the policy $\pi(x, \cdot)$. The second one is followed by the application of the sigmoid activation function to compute the scalar value $v(x)$.

We decided to consider the ResNet-18 as a backbone architecture since with a deeper model there is a high chance to overfit the (relatively small) datasets we used in the experiments. However, in principle there are no special limitations on the architecture of the neural network. The network is trained from scratch after random initialization.

The neural network accepts images of any spatial resolution. The only limitation is the amount of memory available in the system. During training an image needs to be stored for each node in the tree. Images can be downsampled to increase the number of trees that are grown in parallel and to speed-up training in general (in the experiments we resampled images to the resolution of 256 $\times$ 256 pixels).

The Lower part of Figure 1 shows the neural network architecture. In the first layer of the two heads, in order to avoid overfitting, dropout is applied with probability $p$, where $p$ is a tunable hyperparameter. For our experiments the value of this parameter was tuned to 0.6.

3.3. Inference

After training, TreEnhance can be used in two different ways: by building a tree (tree search) or by using directly the trained neural network (policy-based search). The first strategy is more accurate but it also slower. The second is faster and slightly less precise.

3.3.1. Tree Search

at inference time the method builds the tree as in the generation phase, but without having access to the target image for the computation of the return of terminal nodes. When the STOP action is selected or the maximum depth is reached the return is estimated instead by $v(x)$. Once the tree has been built, a sequence of operations is obtained by descending the tree taking at each node the actions that maximize the policy $\rho(x, \cdot)$.

3.3.2. Policy-based Search

the second approach consists in the repeated selection of the operation that by maximizes the probability $\pi(x, \cdot)$ computed by the neural network. The procedure terminates when the STOP action is selected or the maximum number of operations is reached.

4. Experimental Results

To verify the quality of the solutions found by TreEnhance we evaluated it on two datasets widely used to assess the performance of image enhancement methods. More in detail, we used the LOw-light (LOL) dataset and the Adobe five-K dataset. For both datasets global editing operations for tone adjustment and contrast enhancement are required. In the case of LOL spatial operators (such as deblurring, edge enhancement, etc.) can also be useful.
4.1. LOw-Light Dataset

The LOL dataset [36] is composed of 500 pairs of low-light/normal-light images obtained from real scenes. The training set includes 485 images and the test set 15. One of the main problems of real-scene datasets is the alignment of the image pairs: for this reason the authors of the dataset removed all the pairs that have a mean squared error greater than a fixed threshold (0.1). The enhancement of the images in this dataset requires mostly the application of global transformations, white balancing algorithms and spatial filters. The set of operations \( \mathcal{A} \) we selected for this experiment includes:

- **Brightness adjustment**: a constant \( \delta \) is added to the values of all pixels:
  \[(a(x))_{ijc} = \text{clip}(x_{ijc} + \delta).\]  
  (6)

  The clip function limits the adjusted values to the [0, 1] range \((\text{clip}(x) = \min\{\max\{x, 0\}, 1\})\).

- **Contrast adjustment**: the pixels values are stretched by a factor \( \beta \) around their mean:
  \[(a(x))_{ijc} = \text{clip}(\mu_c + \beta \times (x_{ijc} - \mu_c)),\]  
  (7)

  where \( \mu_c \) is the average channel value. We considered the variants with \( \beta \in \{0.8, 2.0\} \) and applied to all the channels, or just one.

- **Gamma Correction**: we considered two values \( \gamma \in \{0.6, 1.1\} \), and applied them to all or to one of the channels:
  \[(a(x))_{ijc} = (x_{ijc})^\gamma.\]  
  (8)

- **Saturation adjustment**: after the conversion to the HSV color space, the \( S \) channel is scaled by a factor \( s \in \{0.5, 2.0\} \) and then the pixel is converted back in the RGB space.

- **Hue rotation**: the hue channel of the HSV color space is shifted by \( h \) degrees \( (h \in \{-18.0, +18.0\}) \).

- **Gray world**: the gray world white balancing algorithm is applied to the image. This algorithm makes the assumption that the images, on average, are gray and the value of the illuminant (in each channel) could be modeled as a constant factor multiplied to the channel. For this reason, in order to remove the variation due to the illuminant, the algorithm normalizes each channel of the image with the average value of the pixels.

  - **Max RGB**: the max RGB white balancing algorithm is applied. This algorithm relies on the fact that the brightest pixel in each channel should be considered as white. Each channel is divided by the highest value.

  - **Median Filter**: replaces the value of a pixel with the median of the values of its neighbors. The dimension \( k \) of the filter’s window was tune to 3 obtaining a \( 3 \times 3 \) filter.

  - **Gaussian Blur Filter**: blurs the input image with a Gaussian filter with standard deviation \( \sigma \). The parameter \( \sigma \) was set to 2.

  - **Sharpening Filter**: increases the sharpness of intensity transitions in an image. The sharpening filter that we considered is the unsharp masking, defined by the kernel \( F \):
    \[
    F = \begin{bmatrix}
    -0.125 & -0.125 & -0.125 \\
    -0.125 & 2 & -0.125 \\
    -0.125 & -0.125 & -0.125
    \end{bmatrix}.
    \]  
    (9)

  - **Edge Enhancement Filter**: highlights the edges in the image.
    \[
    F = \begin{bmatrix}
    -0.5 & -0.5 & -0.5 \\
    -0.5 & 5 & -0.5 \\
    -0.5 & -0.5 & -0.5
    \end{bmatrix}.
    \]  
    (10)

  - **Detail Filter**: the main purpose of this filter is to enhance the details of the subjects/objects present in the image.
    \[
    F = \begin{bmatrix}
    0 & -0.17 & 0 \\
    -0.17 & 1.67 & -0.17 \\
    0 & -0.17 & 0
    \end{bmatrix}.
    \]  
    (11)

  - **Smoothing Filter**: replaces the value of the pixels with the average of their neighbors in the filter window.
    \[
    F = \begin{bmatrix}
    0.077 & 0.077 & 0.077 \\
    0.077 & 0.385 & 0.077 \\
    0.077 & 0.077 & 0.077
    \end{bmatrix}.
    \]  
    (12)

  - **STOP**: terminates the enhancement procedure.
The total number of editing operations is 38 (eight brightness adjustments, eight contrast adjustments, eight gamma corrections, two saturation adjustments, two hue rotations, two white balancing algorithms, seven spatial filters and one stop). The parameters $\delta, \beta, \gamma, s, h, \sigma, k$ have been chosen to make it possible to enhance the images within a reasonable amount of steps. In the case in which, to save memory, TreEnhance is applied to resampled images, the parameters of the spatial operators need to be adjusted when finally applied to the original high resolution images.

The parameters of the training procedure described in Section 3 have been selected on the basis of preliminary experiments, also taking into account the cost in terms of computational resources. In each generation phase we randomly selected 100 training images and for each one we applied 10 000 MCTS iterations to grow the corresponding tree (we set to ten the maximum tree depth). The resolution of images was $600 \times 400$ pixels and standard data augmentation techniques (cropping, flipping and resizing) were used to improve the diversity of the training set. Ten $(x, r(x), p(x, \cdot))$ triplets were sampled from each tree and added to the training set. The parameter $\alpha$ in Equation (3) was set to 0.05 and the exploration parameter $c$ in Equation (2) was set to 10.

In each optimization phase the parameters of the neural network have been updated by minimizing the average loss (5) with 160 iterations of the ADAMW optimizer, with learning rate $10^{-3}$ and weight decay $10^{-2}$.

At the end of the training procedure the method has been applied to the test set. For each test image a tree of 1000 nodes has been grown and a final editing sequence was selected with the procedures described in Section 3.5. Table 1 reports the result of TreEnhance on the LOL test set with the Tree Search inference procedure.

We compared the results of our method, with several state of the art architectures in terms of PSNR, SSIM, $\Delta E$.

TreEnhance was able to outperform all the other methods considered in terms of PSNR and $\Delta E$. Figure 2 presents the results of the application of TreEnhance on the LOL test set. From a qualitative and a quantitative point of view the results obtained are very satisfactory.

4.2. Adobe Five-K Dataset

The second dataset we considered is the Adobe Five-K dataset [40]. It includes 5000 high-resolution images in the device RAW format and in the Adobe RGB space after the work of five different photo editing experts (Expert A, B, C, D, E). We followed the procedure presented by Hu et al. [28] to split the images in a training (4000 images) and a test set (1000 images). As target images we took those edited by Expert C, since it is the most consistent of the five.

For this second dataset, the set of enhancing operators $\mathcal{A}$ was reduced by removing the spatial filters and the white balancing algorithms, exploiting the knowledge that the experts used only those operations that can be obtained with the “color curves” tool of photo editing applications. For the Five-K dataset, the difference between input and ground truth is smaller than that of the LOL dataset. For this reason we changed the parameters of the operators obtaining more fine-grained actions.

- Brightness adjustment: $\delta \in \{-0.05, +0.05\}$;
- Contrast adjustment: $\beta \in \{0.894, 1.414\}$;
From the comparison we can see that TreEnhance provided by the authors, to our preprocessed data. In order to obtain a fair format images, before training the algorithm, a preprocessing was needed. In order to obtain a fair comparison with the other state of the art methods, we applied these algorithms, using the code comparison with the other state of the art methods on the Adobe five-k dataset.

| Method     | PSNR↑ | SSIM↑ | \(\Delta E\downarrow\) |
|------------|-------|-------|-------------------------|
| DSLR [37]  | 11.36 | 0.54  | 32.63                   |
| ExCNet [22]| 15.23 | 0.68  | 24.37                   |
| Zero-DCE   | 15.80 | 0.67  | 21.37                   |
| RetinexNet | 16.15 | 0.70  | 21.99                   |
| MBLEEN     | 17.16 | 0.72  | 18.31                   |
| KinD       | 17.38 | 0.75  | 20.48                   |
| LAE-Net    | 18.27 | 0.83  | 15.02                   |
| KinD++     | 18.30 | 0.64  | -                       |
| EnlightenGAN| 18.75 | 0.82  | 15.13                   |
| DPED [18]  | 19.71 | 0.78  | 15.06                   |
| HDRNet [30]| 20.14 | 0.81  | 14.81                   |
| RetinexDec | 20.23 | 0.83  | 14.26                   |
| TreEnhance | 21.96 | 0.84  | 13.10                   |

**Table 1:** Comparison of state-of-the-art enhancement methods on the LoL dataset.

PSNR↑ 11.36 15.23 15.80 16.15 17.16 17.38 18.27 18.30 18.75 18.75 19.71 20.14 20.23 21.96
SSIM↑ 0.54 0.68 0.67 0.70 0.72 0.75 0.83 0.64 0.82 0.78 0.81 0.83 0.84 0.81
\(\Delta E\downarrow\) 32.63 23.47 21.47 21.99 18.31 20.48 15.02 - 15.13 15.06 14.81 14.26 13.10 10.82

The total number of operations is 29: the STOP action, two saturation adjustments, two hue rotations, brightness, contrast and gamma correction applied either to a single channel or to three channels simultaneously.

The training procedure adopted for this second dataset was the same explained in Section 3. For each generation we considered 100 images from the training set, we downsampled them to 256 × 256 pixels and we applied data augmentation as for the LOL dataset. We executed 10 000 MCTS iterations (the maximum depth was set to ten). The value of the parameters \( c \) in Equation (2) was set to 4: a value smaller than that used for the LOL dataset. We executed 10 000 MCTS iterations (the maximum depth was set to ten). The value of the parameter \( \alpha \) in Equation (2) is 0.05, the same used for the LOL dataset. After each generation phase, we trained the network with mini batch gradient descent using the ADAMW optimizer for 70 iterations.

We compared our method (with the Tree Search inference procedure) with several other methods from the state of the art: results are reported in Table 2.

Since the images provided in the Five-K are RAW format images, before training the algorithm, a preprocessing was needed. In order to obtain a fair comparison with the other state of the art methods, we applied these algorithms, using the code provided by the authors, to our preprocessed data. From the comparison we can see that TreEnhance is competitive on this dataset performing very well from a quantitative point of view. Moreover, it reached the best results in terms of LPIPS a metric that measures the perceptual similarity between two images. It also obtained a very good SSIM value. In terms of the pixel-level metrics PSNR and \(\Delta E\), other methods obtained better results. This fact depends on the granularity of the set of editing operations considered. To obtain more accurate results at the pixel level, we would need to include more fine-grained operations, which would make significantly longer the sequences found by the algorithm. Moreover, when the sequences of operators becomes to long, the level of explainability of the model is reduced. In Figure 3 we compared the images obtained using TreEnhance and other state-of-the-art methods to enhance a sample from the Five-K test set.

TreEnhance is able to obtain very good performance on the two datasets considered. Its results are competitive for all the metrics considered, as reported in tables 1 and 2. However, it was not the best method on the Five-K dataset in terms of MSE and PSNR. A possible explanation for this is that some Five-K images have been retouched with space-
Figure 3: Examples of test images in the Five-K dataset processed by methods in the state of the art. The left most column shows the input image and the right most the ground truth.

Partially varying operators (e.g. masked operations), a condition that is difficult to reproduce exactly by using only global operators.

To better understand the differences among the images produced by the methods in the comparison, we organized a subjective evaluation test. We asked ten participants (all with some experience in the field) to select the best looking among seven versions of the same image. This selection process was repeated for 100 different images from the Five-K test set. The compared methods were TreEnhance, Pix2Pix, CycleGAN, Star-DCE, DaR, Exposure and HDRNet. The participants used a system that shows the seven alternatives in random order that, without any reference to the methods. The order used to present the 100 images was also different for each user.

From the results shown in Figure 4, TreEnhance was the most selected method with the 23.63% of the users preferences followed by HDRNet (22.53%) and Star-DCE(20.54%). The other methods showed much lower preferences: Pix2Pix 15.95%, Exposure 7.48%, DaR 5.58% and CycleGAN 4.29%.

These results confirm the ability of TreEnhance of producing high quality images from a perceptive point of view even if the pixel-level metrics are not the best one. This is confirmed by the best value of the perceptual LPIPS metric. An informal interview after the test pointed out that for most users there were three methods that were clearly superior than the others. These methods, were very stable in the choice of the tonality and their output was free from any artifacts, a very negative element for the others. The users considered the best three methods as substantially equivalent.

From the performed user study, TreEnhance is very competitive respect several methods for image enhancement. Moreover is able to produce images similar to modern state-of-the-art methods.

In addition to this experiments, we also conducted exploratory tests with probabilistic denoising diffusion models [4]. Preliminary results show that these models are very good at preserving the semantic content of images without the introduction of artifacts. However, diffusion models were not able to emulate the intent of the photor editor, and the result is a low accuracy in terms of performance metrics (especially PSNR and $\Delta E$). A possible explanation of these results, is the limited dimension of the datasets used in our experiments compared to those used in the reference paper [4].

5. Discussion

The results obtained by TreEnhance in the two different tasks are satisfactory from both a quantitative and a qualitative point of view. The high quality output images are obtained with the application of basic image editing operations, making the
process easy to explain. During the inference phase TreEnhance evaluates several candidate sequences of operators but at the end only the most promising is selected and applied to the input image.

In this section we discuss the memory and time requirements of TreEnhance, and we analyze more in detail its qualities.

5.1. Memory and Time Requirements

The generation phase of the training procedure is the most demanding step of TreEnhance in terms of memory. This depends on the number of MCTS steps, which also corresponds to the number of nodes in the trees. With respect to other state-of-the-art methods, TreEnhance does not impose any limitations on the resolution of the input images. However, to save memory (and therefore to allow the growth of larger trees) it is possible to work with resampled images. Once the sequence of editing operations has been found on a low-resolution version of the input image, it can be applied to the original high-resolution version. In this case the parameters of spatial operators must be suitably adjusted. Figure 5 shows some examples of the application of TreEnhance on high resolution images.

We observed that a good degree of exploration should be ensured in the initial phases of the training procedure. In fact, at the beginning it tends to look like a breadth-first search where many nodes at the same level are explored until a promising action is found. After that, the method explores more in depth the promising branches of the tree. This happens when the CNN becomes able to assign more accurate values allowing the algorithm to select promising actions more frequently.

As explained in Section 3, only a random subset of the training set is involved in each generation. This subset is split in batches (four in our experiments) in order to make use of all the available memory. To speed up the generation phase, the trees corresponding to each one of the images in the batch grow in parallel. In our experiments we used an Intel® Core™ i7-8700 CPU @ 3.20GHz with 48 GB of memory. For the evaluation step of the generation phase and for the optimization phase we used a NVIDIA RTX™ 3080Ti graphic card with 12 GB of memory.

5.2. Network Ablation

In Section 3 we presented the neural architecture used in this work. In Table 3 we compared different neural network architectures in terms of PSNR on the LOL dataset.

We decided to compare two relatively small networks as ResNet18 and DenseNet121, two mid-size networks as ResNet34 and ResNet50 and the large ResNet101. As we stated before, the method does not impose any constraint for the choice of the neural architecture used to estimate the probability distribution over the actions and the outcome of the enhancement process. The ResNet18 and DenseNet121 have a similar number of parameters and FLOPs (computed for images of resolution 256 × 256). These networks show similar performance in terms of PSNR. Due to the relatively high number of forward passes required by the Tree-based strategy and to an higher PSNR obtained, we decided to select for our method the ResNet18 even if its number of parameters is higher than that of the DenseNet. The performance obtained with larger networks, are worse than those obtained with the smaller ResNet. This is caused by the overfitting of the relatively small training set. We expect that even larger networks, such as ResNet152, would suffer the same problem.

5.3. Tree-based and Policy-based Enhancement

As explained in Section 3 TreEnhance can be used in the inference phase in two different ways: by building a tree for each input image or by directly using the neural network as a policy. In Table 4 we compared the two different inference strategies on the two test sets. Tree Search enhancement is the most accurate strategy since it considers a wide range of alternatives. Policy-based Search strategy acts greedy by taking each time the most promising operation. For this reason it is faster than the tree-based approach and uses less memory but is slightly less accurate.

The additional complexity of the Tree Search strategy can be modulated by limiting the number of MCTS steps during in the inference phase. Figure 6 (left) shows that the accuracy of the method

| Architecture    | Parameters (M) | FLOPs (G) | PSNR |
|-----------------|----------------|-----------|------|
| ResNet18        | 11.32          | 2.38      | 21.96|
| ResNet34        | 21.79          | 4.80      | 20.43|
| ResNet50        | 25.56          | 5.38      | 19.30|
| ResNet101       | 44.55          | 10.52     | 17.13|
| DenseNet121     | 7.98           | 3.76      | 20.16|
Table 4: Comparison of TreEnhance inference strategies on the two dataset considered.

| Search strategy | Five-K | LOL |
|-----------------|--------|-----|
|                 | PSNR  | ΔE  | SSIM | PSNR  | ΔE  | SSIM |
| Tree            | 21.24 | 11.25 | 0.89 | 21.96 | 10.82 | 0.81 |
| Policy-based    | 20.06 | 12.44 | 0.84 | 21.86 | 10.99 | 0.80 |
| Guided          | 25.54 | 7.18 | 0.92 | 24.34 | 8.44 | 0.84 |

increases with the number of steps. Another parameter that can be adjusted during the search is the coefficient $c$ in Equation (2). From Figure 6 (right) it seems that during inference a large value of $c$ is beneficial.

We compared the two inference strategies in terms of time, number of parameters and FLOPs. Both the solutions are based on the modified version of the ResNet18 architecture presented before. The number of parameter is 11.32M. As explained in Section 3, the main difference in the two inference phases is the fact that the first is growing the tree also during inference while the second directly applies the learned policy. The cost in terms of FLOPs for a forward pass on a $256 \times 256$ image is 2.38GFLOPs. For the tree strategy, the total computational cost depends on the number of times the network is evaluates an image associated to a node which, in turn, depends on the maximum number of steps and on the maximum depth of the tree. For the policy-based strategy, the image is evaluated by the neural network until the maximum number of editing operators have been applied or the STOP action has been selected. In Table 5 we compared the computational cost and time of the two search strategies on the Five-K test set with those of other methods from the literature. We computed the average FLOPs per image and the average time per image. Time was measured on a computer with an Intel® Core™ i7-8700 CPU @ 3.20GHz, 48 GB of memory and an NVIDIA RTX™ 3080Ti GPU with 12 GB of memory.

As expected, the Tree strategy is more accurate than the policy-based strategy, at the cost of a higher number FLOPs and, therefore, more time per image.

5.4. Guided Search

As a further experiment we verified if TreEnhance can be used as a tool for “reverse engineering” the work of a professional photo editor. To do
| Method       | Params(M) | FLOPs(G) | Time(ms) | PSNR  |
|--------------|-----------|----------|----------|-------|
| TreEnhance (Tree) | 11.32     | 40.54    | 397.14   | 21.24 |
| TreEnhance (Policy) | 11.32     | 12.43    | 66.91    | 20.06 |
| CycleGAN     | 11.37     | 1642.95  | 19.38    |       |
| Pix2Pix      | 54.41     | 18.15    | 267.54   |       |
| STAR-DCE     | 0.03      | 0.02     | 25.17    | 23.55 |
| DCE-Net      | 0.08      | 5.22     | 1693.44  | 22.31 |
| Unfiltering  | 148.02    | 2.59     | 816.56   | 21.67 |
| Parametric   | 0.03      | 0.05     | 756.93   | 23.20 |

Table 5: Comparison of the two inference strategies in terms of parameters, FLOPs and Time on the Five-K test set. Additionally, computational requirements of TreEnhance are also compared to those of other methods from the literature.

so, we modified TreEnhance by using the ground truth during the inference phase (Equation 3). This way the MCTS search will retrieve the sequence of operations that can be used to obtain a retouched version of the image from the input. We called this third strategy “Guided Search”. The guided search is useful for two different reasons; moreover, it provides an upper bound to TreEnhance that can help the selection of the pool of enhancement operators for a new dataset; moreover, it could be included in a photo editing application, to help beginners to imitate expert users and to help professionals to revise and optimize their enhancement pipelines.

We reported the results obtained in Table 4. The accuracy obtained in this scenario is very high confirming the effectiveness of MCTS in finding good sequences of image editing operations. This experiment suggests that this strategy could be included in an educational software, like a puzzle to be solved, to teach beginners how to enhance a low-quality image and what kind operators they should use to obtain a given result.

5.5. Analysis of the Sequences

The sequences produced by the three inference strategies are correlated. In fact, as it is shown in Figure 7 all the strategies favor global enhancing operators. In particular, the most selected operators are those that increase the brightness, the gamma, and the contrast. This depends on the fact that the original LOL test images are low light ones and need to be brightened. Operations that work on single channels are more prone to be selected when the current image is close to the target one.

Tree-based and policy-based strategies tend to over-select the gamma correction action all over the tree channels with respect to the guided search (action 6). Guided search, in fact, tends to have a more balanced distribution of the operations over the three channels with respect to the other strategies, resulting in higher quality images.

5.6. Artifacts

One of the main features of the generative models for image enhancement, like those based on GANs, is the ability to accurately model color distributions. However, as a drawback they often introduce visible artifacts in their outputs. Figure 8 shows a comparison among two generative methods (Pix2Pix and CycleGAN) and two reinforcement learning methods (TreEnhance and Exposure) on two Five-K test images. In both cases generative methods produced a very good color distribution, but with the presence of visible noise and artifacts. In fact, the value of perceptual metrics such as LPIPS tends to be relatively bad for this kind of methods. Reinforcement learning methods do not introduce any artifacts, but at the cost, sometimes, of less saturated colors.

6. Conclusions

In this paper we presented TreEnhance, a lightweight fully automatic and explainable low-light image enhancement method. TreEnhance shows that it is possible to obtain very good performance in two image enhancement tasks. It finds solutions that are fully explainable, since they can be decomposed as a sequence of simple image processing operations.

The method can be used for other enhancement tasks just by changing the set of image editing operations. Thanks to the support for multiple search strategies, TreEnhance is suitable for a variety of scenarios with different requirements in terms of...
constraints on memory and processing time, and with different levels of interaction with the user.

More in detail experiments demonstrated that the method can obtain good performance on two low-light datasets with the possibility to choose three different search strategies: Tree-based, building the tree for each input image, and choosing the most promising path toward the enhanced image without using the ground truth image, Policy based, where the policy is sampled directly from the neural network, and Guided search, where the tree is created with the usage of the ground truth image.

The results obtained are very good from a qualitative and a quantitative points of view. However, TreEnhance shows some limitations when the enhancement style it is required to reproduce is not consistent, or when it requires complex non-global editing operations. To solve this issue, a more complex and dataset-specific set of editing operators should be designed, possibly by also including spatially varying operations.

TreEnhance demonstrates how tree-search theory, and in particular MCTS, can be a very useful tool for image processing. As a future expansion of this work, we plan to investigate the application of tree search theory in different research areas. One of these, could be the field of image retargeting where the MCTS could be able to select areas of the image that could be deleted, with the help of a series of retargeting operators, in order to center correctly the subjects of the image. The application of TreEnhance to other relevant fields, like super-resolution and image retrieval, could give the possibility of not only achieving great results with the application of basic operators, providing an explaination to the processing steps applied to the image.

Another promising direction of research is the design of diffusion models for low-light image enhancement. In our preliminary tests, these generative models were able to produce very good images without artifacts, but they were not able to emulate the enhancement style of the photo editor. In the future, we plan to investigate how to overcome these limitations of diffusion models.

References

[1] P. Isola, J.-Y. Zhu, T. Zhou, A. A. Efros, Image-to-image translation with conditional adversarial net-
works, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1125–1134.

[2] J.-Y. Zhu, T. Park, P. Isola, A. A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2223–2232.

[3] A. Ignatov, N. Kobyshev, R. Timofte, K. Vanhoey, L. Van Gool, Dslr-quality photos on mobile devices with deep convolutional networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 3277–3285.

[4] C. Saharia, W. Chan, H. Chang, C. Lee, J. Ho, T. Salimans, D. Fleet, M. Norouzi, Palette: Image-to-image diffusion models, in: ACM SIGGRAPH 2022 Conference Proceedings, 2022, pp. 1–10.

[5] G. Kim, T. Kwon, J. C. Ye, Diffusionclip: Text-guided diffusion models for robust image manipulation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 2426–2435.

[6] H. Sasaki, C. G. Willcocks, T. P. Breckon, UniG-dpm: Unpaired image translation with denoising diffusion probabilistic models, arXiv preprint arXiv:2104.05358 (2021).

[7] G. Batsalis, J. Stanczuk, C.-B. Schönlieb, C. Etmann, Conditional image generation with score-based diffusion models, arXiv preprint arXiv:2111.13606 (2021).

[8] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Advances in neural information processing systems 30 (2017).

[9] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 618–626.

[10] P. Ravirathinam, D. Goel, J. J. Ranjani, C-lienet: A multi-context low-light image enhancement network, IEEE Access 9 (2021) 31053–31064.

[11] Y. Jiang, X. Gong, L. Zhang, L. Zhang, X. Liu, Y. Shen, S. Zhang, S. Zhao, Enlightengan: Deep light multi-context low-light image enhancement network, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 4459–4468.

[12] C. Guo, C. Li, J. Guo, C. C. Loy, J. Hou, S. Kwong, R. Cong, Zero-shot restoration of back-lit images using deep internal learning, in: Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 1625–1631.

[13] Y. Chai, R. Gires, L. Wolf, Supervised and unsupervised learning of parameterized color enhancement, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2020, pp. 992–1000.

[14] H. Kim, S.-M. Choi, C.-S. Kim, Y. J. Koh, Representative color transform for image enhancement, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 4459–4468.

[15] J. Park, J.-Y. Lee, D. Yoo, I. S. Kweon, Distort-and-recover: Color enhancement using deep reinforcement learning, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 5928–5936.

[16] H. Yang, B. Wang, N. Veshapunt, M. Guo, S. B. Kang, Personalized exposure control using adaptive metering and reinforcement learning, IEEE transactions on visualization and computer graphics 25 (10) (2018) 2953–2968.

[17] Y. Hu, H. He, C. Xu, B. Wang, S. Lin, Exposure: A white-box photo post-processing framework, ACM Transactions on Graphics (TOG) 37 (2) (2018) 1–17.

[18] K. Yu, C. Dong, L. Lin, C. C. Loy, Crafting a toolchain for image restoration by deep reinforcement learning, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 2443–2452.

[19] R. Yu, W. Liu, Y. Zhang, Z. Qu, D. Zhao, B. Zhang, Deepexposure: Learning to expose photos with asynchronously reinforced adversarial learning, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 2153–2163.

[20] Y. Cai, J. Lin, X. Hu, H. Wang, X. Yuan, Y. Zhang, R. Timofte, L. Van Gool, Mask-guided spectral-wise transformer for efficient hyperspectral image reconstruction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 17502–17511.
R. Timofte, L. Van Gool, Coarse-to-fine sparse transformer for hyperspectral image reconstruction, arXiv preprint arXiv:2203.04845 (2022).

[33] Z. Zhang, Y. Jiang, J. Jiang, X. Wang, P. Luo, J. Gu, Star: A structure-aware lightweight transformer for real-time image enhancement, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 4106–4115.

[34] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, S. Colton, A survey of monte carlo tree search methods, IEEE Transactions on Computational Intelligence and AI in games 4 (1) (2012) 1–43.

[35] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, D. Hassabis, A general reinforcement learning algorithm that masters Chess, Shogi, and Go through self-play, Science 362 (6419) (2018) 1140–1144.

[36] W. Chen, W. Wenjing, Y. Wenhan, L. Jiaying, Deep retinex decomposition for low-light enhancement, in: British Machine Vision Conference, 2018.

[37] S. Lim, W. Kim, Dslr: Deep stacked laplacian restorer for low-light image enhancement, IEEE Transactions on Multimedia (2020).

[38] X. Liu, W. Ma, X. Ma, J. Wang, Lae-net: A locally-adaptive embedding network for low-light image enhancement, Pattern Recognition (2022) 109039.

[39] M. Gharbi, J. Chen, J. T. Barron, S. W. Hasinoff, F. Durand, Deep bilateral learning for real-time image enhancement, ACM Transactions on Graphics (TOG) 36 (4) (2017) 1–12.

[40] V. Bychkovsky, S. Paris, E. Chan, F. Durand, Learning photographic global tonal adjustment with a database of input / output image pairs, in: The Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition, 2011.

[41] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in: CVPR, 2018.