UNIVERSITÀ
DI PAVIA

UNIVERSITY OF PAVIA

FACULTY OF ENGINEERING

DEPARTMENT. OF ELECTRICAL, COMPUTER AND BIOMEDICAL ENGINEERING

Ph.D. DEGREE in ELECTRONICS, COMPUTER SCIENCE and ELECTRICAL ENGINEERING

Ph.D. THESIS

Deep And Reinforcement Learning Approaches for Computational Photography

Candidate: Marco Cotogni

Supervisor: Prof. Claudio Cusano

Ph.D. Program Chair: Prof. Ilaria Cristiani

XXXVI Cycle - A.Y. 2022/2023

“Writing a PhD thesis is like trying to eat an elephant – one bite at a time. Just remember, even the grandest ideas started as scribbles on a napkin. So, buckle up, sip that coffee, and let’s turn these sleepless nights into a masterpiece! ”

— ChatGPT

Abstract EN

Computational photography represents a fusion of computer science and traditional photography, driven by the goal of enhancing the quality of images captured with conventional cameras. Leveraging digital processing techniques and advanced computational algorithms, this field elevates the perceptual quality of photographs while minimizing the associated resource and post-processing demands. Such algorithms have found their way into various post-processing software, providing valuable assistance to photographers, whether seasoned professionals or newcomers. Nevertheless, despite the effectiveness of these algorithms, several pressing challenges remain unaddressed.

In this thesis three main challenges in computational photography are analyzed: **(i)** the enhancement of low-quality images through artifacts-free fully interpretable algorithms, **(ii)** the lack of a photographic dataset for photo authorship attribution and photographic style transfer and **(iii)** the improvement of state-of-the-art convolutional architectures, particularly concerning color constancy in the presence of changing scene illuminants.

The first challenge focus on the enhancement of low-quality images through the development of artifact-free, fully interpretable algorithms. Two novel image enhancement methodologies are introduced, based on tree-search theory and deep reinforcement learning. These techniques generate interpretable sequences of enhancement operators enhancing the visual content of low-quality input images. Specifically, one approach employs global image enhancement operators, while the other utilizes local spatial operators to independently enhance different portions of input images. Additionally, an explainability method for image enhancement black-box algorithms is presented. This method, employing a path planning algorithm, not only emulates state-of-the-art enhancements but also rectifies artifacts in the resulting images.

The second challenge is tackled presenting a novel dataset, PhotoStyle60 and its subset PhotoStyle10. This dataset contains more than 5700 photographs from 60 different professional and amateurial photographers. This dataset is analyzed and tested on two foundamental scenarios, photo authorship attribution and photographic style transfer. Moreover a novel multi-image photographic style transfer method is proposed.

The third challenge centers on improving state-of-the-art convolutional neural networks, particularly concerning their performance in the presence of changing scene illuminants. To address this issue, a novel neural network, the "Offset Equivariant neural network," is introduced. This neural architecture shows high performance in the color constancy task, outperforming existing models in the considered scenarios. Moreover, the effectiveness of this neural design is validated across two additional tasks: image recognition and image inpainting.

This thesis makes substantive contributions to the field of computational photography, offering innovative solutions to critical challenges. These contributions aim to advance the field, ultimately augmenting the capabilities of traditional cameras and expanding the creative possibilities available to photographers.

Abstract IT

La fotografia computazionale rappresenta una fusione tra informatica e fotografia tradizionale, con l'obiettivo di migliorare la qualità delle immagini catturate con fotocamere convenzionali. Sfruttando tecniche di elaborazione digitale e algoritmi computazionali avanzati, questo campo eleva la qualità percettiva delle fotografie riducendo al minimo le risorse necessarie e le esigenze di post-produzione. Tali algoritmi hanno trovato applicazione in vari software di post-produzione, offrendo preziosa assistenza ai fotografi, sia professionisti esperti che principianti. Tuttavia, nonostante l'efficacia di tali algoritmi, rimangono aperte alcune sfide fondamentali.

In questa tesi sono analizzate tre sfide principali nella fotografia computazionale: **(i)** il miglioramento delle immagini di bassa qualità mediante algoritmi completamente interpretabili e privi di artefatti, **(ii)** la mancanza di un dataset fotografico per il riconoscimento dell'autore di fotografie e il trasferimento dello stile fotografico e **(iii)** il miglioramento delle architetture convoluzionali, in particolare per quanto riguarda il task di color constancy in presenza di cambiamenti nell'illuminazione della scena.

La prima sfida si concentra sul miglioramento delle immagini di bassa qualità mediante lo sviluppo di algoritmi completamente interpretabili e privi di artefatti. Vengono presentate due nuove metodologie di miglioramento delle immagini, basate sulla algoritmi di ricerca per alberi e reinforcement learning. Queste tecniche generano sequenze interpretabili di operatori di image processing che migliorano il contenuto visivo delle immagini di input di bassa qualità. In particolare, un approccio impiega operatori globali, mentre l'altro utilizza operatori spaziali locali per migliorare in modo indipendente diverse parti delle immagini di input. Inoltre, viene presentato un metodo di interpretabilità per algoritmi di enhancement delle immagini black-box. Questo metodo, utilizzando un algoritmo di path planning, è in grado non solo di emulare i miglioramenti dei metodi all'avanguardia, ma anche di correggere gli artefatti nelle immagini risultanti.

La seconda sfida affronta la mancanza di un dataset fotografico completo presentando PhotoStyle60 e il suo sottoinsieme PhotoStyle10. Questo dataset comprende oltre 5700 fotografie provenienti da 60 diversi fotografi professionisti e amatoriali. Questi dataset vengono analizzati e testati su due scenari fondamentali: il riconoscimento dell'autore delle foto e il trasferimento dello stile fotografico. Inoltre, viene proposto un nuovo metodo di trasferimento dello stile fotografico multi-immagine.

La terza sfida si concentra sul miglioramento delle architetture neurali convoluzionali, in particolare per quanto riguarda le prestazioni in presenza di cambiamenti nell'illuminazione della scena. Per affrontare questa sfida, viene introdotta una nuova rete neurale, la "Rete Neurale Equivariante", che si dimostra efficace nel task di color constancy, superando i modelli esistenti nei casi considerati. Inoltre, l'efficacia di questa architettura neurale viene testata su due task aggiuntivi: image recognition e image inpainting.

Questa tesi offre contributi significativi nel campo della fotografia computazionale, proponendo soluzioni innovative per sfide critiche. Questi contributi mirano a far progredire il settore, aumentando in definitiva le capacità delle fotocamere tradizionali e ampliando le possibilità creative a disposizione dei fotografi.

Contents

List of Figures

List of Tables

1 Introduction

In recent years, the field of photography has undergone a profound transformation due to the convergence of digital imaging technologies and advanced computational techniques. This fusion has given rise to a revolutionary approach known as computational photography, redefining the boundaries of traditional photography. Computational photography leverages the power of algorithms and computational methods to enhance, manipulate, and create images beyond the constraints of traditional optical systems. This paradigm shift opens up a realm of creative possibilities but also introduces a set of intricate challenges that require innovative solutions.

Computational photography transcends the limitations of conventional photography by integrating computational algorithms into the imaging process. Unlike traditional photography, which predominantly relies on optical principles to capture light, computational photography actively manipulates captured data through complex algorithms. These algorithms enable a range of functionalities such as image enhancement, noise reduction, depth estimation, high dynamic range (HDR) imaging, and even the synthesis of entirely new images. Nowadays, post-processing programs integrate these functionalities inside their suite of tools. Deep Learning algorithms exhibit an exceptional aptitude for extracting complex patterns and features from visual data. In the context of computational photography, Deep Learning algorithms can be trained to understand image semantics, distinguish between noise and signal, and even predict missing image information. Through iterative learning, these algorithms adapt to diverse photographic scenarios, enabling automated and intelligent enhancement processes.

Despite its effectiveness, Deep Learning algorithms for computational photography present many issues and challenges. One primary challenge is the preservation of visual realism and authenticity. As algorithms process and modify images, there is a risk of straying from the natural look and feel of the scene. Moreover, the application of algorithms may introduce artifacts that ruin the content of the images. Furthermore, computational photography often requires real-time processing, demanding substantial computational resources. This necessity raises concerns about energy efficiency and accessibility on various devices. Addressing these challenges is vital to ensuring that computational photography remains a tool for artistic expression while upholding the integrity of captured moments.

1.1 Focus of the thesis

The main objective of this thesis is to develop novel deep and reinforcement learning algorithms designed to tackle different computational photography challenges including image enhancement, photo authorship attribution and style transfer, computational color constancy and inpainting. In particular, three main tasks have been addressed:

- State-of-the-art image enhancement methods are usually black-box approaches without any insight into the enhancement process applied. Moreover, these methods often introduce artifacts and degradation. In Chapter 3 these challenges are addressed by presenting three different methods. The first is a deep reinforcement learning algorithm that makes use of tree-search theory to find sequences of global image enhancement operators able to improve the quality of input images. Similarly, the second method, through spatial and just one local histogram equalization operator is able to precisely enhance small portions of the input image. Finally, the third method has the goal of explaining and improving the results of black-box methods from the state of the art.

- Photography, like painting, allows artists to express themselves through their unique style. In digital photography, this is achieved not only with the choice of the subject and the composition but also by means of post-processing operations. The automatic identification of a photographer from the style of a photo is a challenging task, for many reasons, including the lack of suitable datasets including photos taken by a diverse panel of photographers with a clear photographic style. To address this, Chapter 4 presents a new dataset containing 5708 photographs from 60 different photographers. The dataset has been analyzed on two different tasks: photo authorship attribution and photographic style transfer. In the former, several state-of-the-art models for image classification have been trained to correctly identify the author of a photograph. In the latter, instead, several style transfer methods have been used to transfer the style of a given photographer to a new image. Finally, a new multi-image style transfer method has been proposed.

- Pretrained state-of-the-art neural networks usually show a shift in their prediction when photometric transformations are applied to the input images. This is due to the fact that these models are trained on data that has a clear and quantifiable illuminant component. For this reason, the performance degrades when the input is an image with a different illuminant is provided to the neural network. To address this challenge, Chapter 5 presents a novel architecture that is invariant to photometric transformations applied to the input images. This architecture, designed specifically for computational color constancy, has been also tested on two additional tasks like image classification and image inpainting, confirming its effectiveness with respect to state-of-the-art models under variable lighting conditions.

1.2 Thesis Structure

The thesis is organized in six chapters. Chapter 2 provides a review of deep and reinforcement learning methods for computational photography. The first half of this chapter presents the foundational building blocks of deep learning with neural networks and reinforcement learning. The second half presents a detailed survey of computational photography tasks and of their most effective learning solutions. The chapter concludes with an overview of other important computational photography tasks not covered in this thesis.

Chapter 3 presents tree and heuristic search-based methods for photographic image enhancement. The first two sections of this chapter show the outcomes and progress obtained through tree-based algorithms. The first algorithm mixes deep reinforcement learning with tree search theory to enhance low-light images. Similarly, the second method, based on the same pipeline, enables targeted enhancements that focus on independent areas of the images. The final section of the chapter introduces a heuristic search-based technique designed to explain and enhance the performances of black-box image enhancement methods.

In Chapter 4 a novel dataset "PhotoStyle60" is presented. It is composed of 5708 photographs from 60 distinct photographers. The chapter continues with an extensive analysis of the dataset through the application of methods from the state-of-the-art to address two tasks: photo authorship attribution and photographic style transfer. The chapter concludes by presenting a new multi-image style transfer method that ranked first in a user study we performed to test its effectiveness.

In Chapter 5, the concept of illuminant equivariant neural networks is introduced. The chapter first provides an overview of the layers composing these architectures, the equivariant layers, then it presents the effectiveness of these networks on three computer vision tasks such as computational color constancy, image recognition and image inpainting.

Chapter 6 summarizes the contributions of the thesis and provides an overview of the possible future research directions that can be analyzed.

1.3 Scientific Contributions

Main publications relevant for the thesis

Chapters 3, 4, and 5 of this thesis are built upon works that have been published in international journals or are currently under review. The deep and reinforcement learning algorithms for image enhancement and their results presented in Chapter 3 are based upon the works listed as 1, 4, and 2 in the list below. Chapter 4, addressing the tasks of photo authorship attribution and the transfer of photographic styles, executed through the new proposed dataset, is based on work 3. Finally, the concluding methodological chapter, Chapter 5, contains the results and the definition of illuminant equivariant networks based on work 5.

1. **Cotogni, Marco**, & Claudio, Cusano. TreEnhance: A tree search method for low-light image enhancement. Pattern Recognition (2023).

2. **Cotogni, Marco**, & Claudio, Cusano. Explaining Image Enhancement Black-Box Methods through a Path Planning Based Algorithm. Multimedia Tools and Applications (2023).

3. **Cotogni, Marco**, Arazzi, M., & Cusano, C PhotoStyle60: A Photographic Style Dataset for Photo Authorship Attribution and Photographic Style Transfer (2023). Under Review at IEEE Transactions on Multimedia.

4. **Cotogni, Marco**, & Claudio Cusano. Select & Enhance: Masked-Based Image Enhancement Through Tree-Search Theory and Deep Reinforcement Learning (2023). Under Review at Pattern Recognition Letters.

5. **Cotogni, Marco**, & Claudio, Cusano. Offset equivariant networks and their applications. Neurocomputing (2022).

Additional Publications

The next list contains the scientific contributions produced during the course of the Ph.D. program that are not included in the content of this thesis. These contributions focus on the use of deep learning to address various computer vision challenges. For instance, the recognition of handwritten mathematical expressions in offline scenarios was explored in work 6, while innovative architectural strategies were devised for enhancing visual transformers' efficacy in analyzing the exemplar-free continual learning task, is presented in work 13. Additionally, deep learning algorithms found application within the biomedical imaging field, as attested by works 11 and 12.

The other papers, reported in the list below, lay in the field of data science, and focus on the application of Machine and Deep Learning algorithms to structured data. In work 7 and 8 feature engineering and machine learning methodologies have been applied for prognosticating Parkinson's Disease progression. Moreover, works 9 and 10 extended their focus towards the application of Graph Neural Networks (GNNs) to predict social media engagement patterns.

6. **Cotogni, Marco**, Cusano, C., & Nocera, A. Recursive recognition of offline handwritten mathematical expressions. 2020 25th International Conference on Pattern Recognition (ICPR, 2021).
7. **Cotogni, Marco**, Sacchi, L., Sadikov, A., & Georgiev, D. Asymmetry at disease onset is not a predictor of Parkinson's disease progression. Journal of Parkinson's Disease (2021).
8. **Cotogni, Marco**, Sacchi, L., Georgiev, D., & Sadikov, A.. Detection of Parkinson's Disease Early Progressors Using Routine Clinical Predictors. 2021 19th International Conference on Artificial Intelligence in Medicine (AIME, 2021).
9. Arazzi, M., **Cotogni, Marco**, Nocera, A., & Virgili, L. Predicting Tweet Engagement with Graph Neural Networks. 2023 ACM International Conference on Multimedia Retrieval (ICMR, 2023).
10. Arazzi, M., **Cotogni, Marco**, Nocera, A., Ursino, D., & Virgili, L. A Multiclass Graph Neural Network-Based Framework for Predicting Post Engagement in Social Media (2023). Under Review at Applied Soft Computing.
11. Bosco, E., Stellino, C., **Cotogni, Marco**, Cusano, C., Ramalli, A., & Matrone, G. Image-to-image translation with deep neural networks for the enhancement of monostatic synthetic-aperture ultrasound images. IEEE International Ultrasonics Symposium (IUS, 2023).
12. Bosco, E., Casula, F., **Cotogni, Marco**, Cusano, C., & Matrone, G. Deep semantic segmentation of echocardiographic images using vision transformers. IEEE International Ultrasonics Symposium (IUS, 2023)
13. **Cotogni, Marco**, Yang, F., Cusano, C., Bagdanov, A. D., & Van De Weijer, J. Gated Class-Attention with Cascaded Feature Drift Compensation for Exemplar-free Continual Learning of Vision Transformers (2023). Under Review at International Journal of Computer Vision.

2 Literature Review of Deep and RL methods in Computational Photography

This chapter presents a literature review of deep and reinforcement learning methods in computational photography. The first part of the chapter outlines the fundamentals of deep learning with neural networks and reinforcement learning. The chapter then continues with a review of deep reinforcement learning, which is a learning paradigm that combines classical reinforcement learning algorithms with neural networks and deep learning. The second half of the chapter presents learning-based algorithms for computational photography, focusing on the tasks covered in the methodological part of this thesis. Finally, for the sake of completeness, the chapter concludes with a review of additional relevant tasks in computational photography that are not covered by the content of this thesis.

2.1 Deep Learning and Neural Networks[1]

Over the past 15 years, deep learning (DL) algorithms, a subset of algorithms belonging to the larger family of machine learning (ML) algorithms, have become the standard techniques for learning and understanding intrinsic patterns from unstructured data such as images, text, and audio. This is due to the ability to use mathematical models with a significantly larger number of parameters compared to classical machine learning algorithms which have shown great potential when applied to structured or tabular data, but have limitations in terms of scalability and performance when applied to unstructured data.

2.1.1 Deep Learning in a nutshell

While there are differences between machine learning (ML) and deep learning (DL), these learning paradigms share the same objectives and use similar optimization techniques to achieve them. The first requirement for an ML/DL algorithm, denoted as f_θ, is a set of structured or unstructured data, denoted as X. Depending on the task the learning algorithm needs to solve, the dataset X may be accompanied by a set of labels Y. In supervised learning, where each data sample $x \in X$ is associated with a corresponding label $y \in Y$, the task is labeled as "supervised". However, in

[1]This section is based on [37, 43, 60, 63, 190]

cases where the label $y \in Y$ is only available for a subset of the data in X, the model f_θ faces a "semi-supervised" problem. In both supervised and semi-supervised tasks, the objective is to optimize the parameters θ of the mathematical model f_θ in order to learn a representation capable of automatically mapping each sample $x_i \in X$ to its corresponding label $y_i \in Y$.

Differently, in unsupervised or self-supervised learning, the label set Y is not available. In unsupervised learning, the parameters θ are optimized to discover intrinsic patterns and relationships within the data, without relying on labels. On the other hand, self-supervised learning aims to find a shared representation of the data X in a high-dimensional space that can be beneficial for solving various supervised and unsupervised downstream tasks.

Independently of the task, the data X is typically divided into three subsets: the training set, the validation set, and the test set. The training set usually contains the majority of the data from X, ideally ranging from 75% to 85%. This set is typically shuffled to eliminate any positional or temporal dependencies and is provided as input to the model f_θ during the training phase. The validation set instead, is used during the training phase to validate the performances of the model while it is refining its parameters and for tuning its hyperparameters. This set is also important to analyze if the model is overfitting, i.e. the model is adapting too much its parameters on the basis of the training data losing its ability of generalization, or underfitting, i.e. the model is not able to learn the patterns hidden in the training data, the training data. Once the model concludes the training phase, it is tested with the test data. This last set, composed of unseen data, is used to assess the ability of generalization of the model.

Considering any type of neural network f_θ (a more detailed description is provided in the next section), the training phase aims at optimizing the parameters θ to ideally approximate a target function. In supervised learning scenarios, such as mapping each sample $x_i \in X$ to its corresponding label $y_i \in Y$, the target function to approximate is denoted as f^*. To achieve this, the training data can be divided into batches $b_0, \dots, b_n$, each containing a subset of randomly sampled data. The batch size can vary between 1 and the dimension of the training set, determining the number of batches. One batch at a time is forwarded to the neural network, initialized with random or non-random parameters θ_0, producing a prediction $\hat{y} = f(b_0)$ as output. Due to the complexity of the problem, it is unlikely that the neural network can accurately approximate f^* after a single forward pass. Thus, the prediction $\hat{y}$ often deviates from the target.

To quantify the error computed by the neural network, a loss function $\mathcal{L}(f_\theta(x))$ is computed using the prediction. This loss function inherently contains information about the current state of the parameters θ, which are responsible for the inaccurate predictions and need to be refined to match the correct parameters θ^* of the target function f^*. For this, the loss function $\mathcal{L}$ is backpropagated up to the input layer of the neural networks computing the gradients for each learnable parameter $\theta_i \in \theta$

composing the neural network. The idea behind backpropagation, is to find the contribution of each of the parameters of the model to the computed error. Following the concept of optimizing a convex function, the parameters θ can be iteratively updated using the Gradient Descent algorithm, which aims to find the global minima. Specifically, considering f_θ^* as the target function, the parameters θ of our function approximator (neural network f_θ) are updated using the following equation:

$$\theta' = \theta - \eta \nabla_\theta \mathcal{L}(\theta) \tag{2.1}$$

where, θ' represents the updated parameters of the neural network, η is the learning rate (i.e., the step size for the update), and $\nabla_\theta \mathcal{L}(\theta)$ is the gradient of the loss function $\mathcal{L}$ with respect to the weights θ. Once the weights have been updated, the next batch of data is provided as input to the neural network, and the gradient step is repeated.

The choice of batch size is crucial for accurately approximating the target function. When the batch size is 1, resulting in a number of batches equal to the dimension of the training set, it is referred to as Stochastic Gradient Descent (SGD). In contrast, if the batch size is equal to the dimension of the training set, resulting in only 1 batch, it is called Batch Gradient Descent. Mini-batch Gradient Descent falls between these two techniques, constructing batches of k examples and obtaining N/k batches (where N is the total number of training examples), resulting in N/k gradient updates. These three well-known techniques differ in how they approach reaching the global minima. Other optimization algorithms that are commonly (and widely) used in deep learning are Adam [113], RMSprop [67] and L-BFGS [103].

After observing all the batches, it is possible that the updated weights are still far from the optimal values. Therefore, a common technique is to iteratively provide the batches to the network until the weights converge to a minimum. Each iteration over all the available batches is called an epoch. Typically, at the end of each epoch, the neural network is evaluated using the validation set to assess its generalization performance and to analyze whether the model is overfitting or underfitting. Possible regularization techniques, such as weight decay, or early stopping, can be employed to prevent overfitting. Then, a new epoch begins. As shown in Figure 2.1, Batch Gradient Descent reaches the minima of the function without fluctuation at the end of training. However, due to the requirement of observing the entire dataset before computing a gradient update, it can be time-consuming when dealing with large datasets. Mini-batch and Stochastic Gradient Descent, on the other hand, are faster since they compute updates for every batch or every sample. However, they are more sensitive to fluctuations in the error, resulting in multiple passes near the minima before convergence. Once the model achieves satisfactory performance on the validation data, it is ready to be tested on the test set.

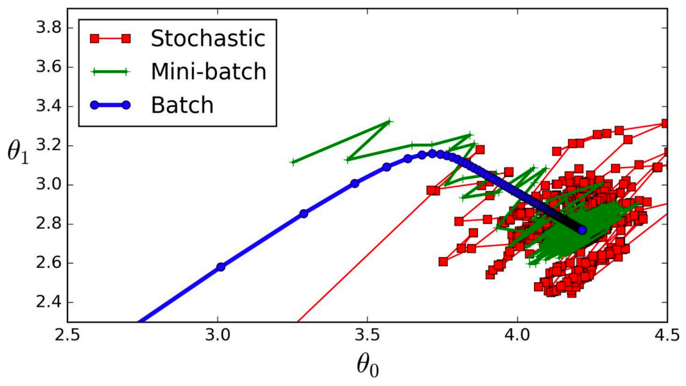


Figure 2.1: Gradient Descent Algorithms Examples. Illustration taken from [60].

2.1.2 Neural Networks

In this section, the most influential neural network architectures are presented, with a particular focus on state-of-the-art architectures for computer vision tasks.

MLPs

One of the first neural network developed is the feedforward neural network, or multilayer perceptrons (MLPs). This neural network typically consists of a stack of basic multi-perceptron layers. Each perceptron in layer l is connected to every perceptron in layer $l+1$ with a weight $\theta^{l}_{n,q}$ but the perceptrons within a layer l are not interconnected As it is possible to observe from Figure 2.2, the neural network in this example has three layers: an input layer (Layer 0), a hidden layer (Layer 1), and an output layer (Layer 2). The input layer receives the input data sample x and its dimension is equal to the dimension of the input data. In the case of unstructured data, i.e. tensors with a number of dimensions > 1, each data sample has to be flattened (for example $C, H, W \rightarrow C \times H \times W$). The remaining layers, are composed by a variable number of perceptrons. Each of perceptron in the neural network is a linear function

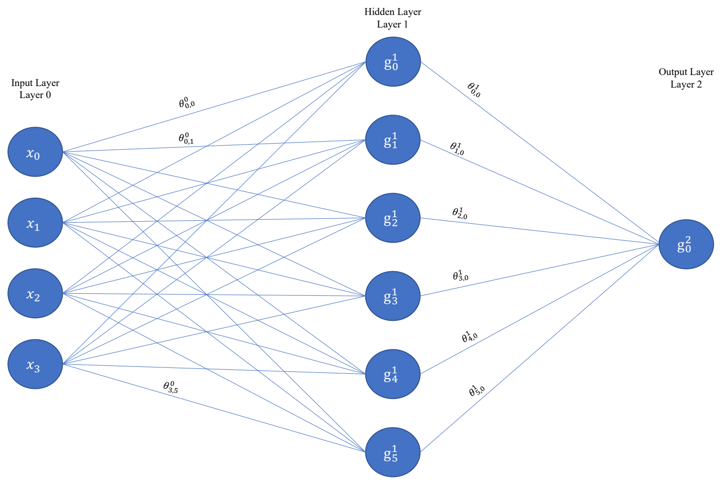


Figure 2.2: Example of a simple MLPs network. For clearness only few weights are reported and the biases b are omitted.

combining all the inputs (the indexes are referred to the MLP in figure) plus a bias b:

$$\begin{aligned} &x = [x_0, x_1, x_2, x_3], \qquad x \in X \\ &\theta^0_0 = [\theta^0_{0,0}, \theta^0_{1,0}, \theta^0_{2,0}, \theta^0_{3,0}], \\ &g^1_0(x) = x^T\theta^0_0 + b^0_0. \end{aligned} \tag{2.2}$$

In order to model each possible non-linear function, a non-linear activation function is applied to the linear combination of θ^0_0, x and b^0_0. The choice of the most suitable activation functions heavily depends on several factors such as the objective of the problem to be solved, the type of architecture used, etc. Here are reported most of the commonly used activation functions :

$$\begin{aligned} z &= g^1_0(x); \\ a(z) &= \frac{1}{1+e^{-z}}; && \text{sigmoid} \\ a(z) &= \frac{1}{tan(z)}; && \text{arctangent} \\ a(z) &= \frac{e^z - e^{-z}}{e^z - e^{-z}}; && \text{hyperbolic tangent} \\ a(z) &= \max(z, 0); && \text{ReLU} \end{aligned} \tag{2.3}$$

Among them ReLU, Rectified Linear Units, and its variants like Leaky ReLU or GeLU, is the standard activation functions adopted in neural networks. In fact, thanks to his piecewise linearity, it preserves the linear property of the model, making the optimization of the parameters θ easier, while at the same time it allows the model to ideally emulate complex non-linear functios. The last layer of the MLP is the output layer. This layer, composed as the other by a variable number of perceptrons, provide as output the logits, i.e. the unnormalized output of the model. The number of perceptrons in the output layer and the activation function applied to the logits depend on the task being solved (e.g., binary classification, multi-class classification, regression, segmentation, etc.). For binary classification, commonly used is a single perceptron in the output layer with a sigmoid activation function:

$$\sigma(z) = \frac{1}{1+e^{-z}}. \tag{2.4}$$

This function takes as input the logit from the only perceptron and transforms it in the probability of a sample x belonging to the target class. In the case of multi-class classification, the number of perceptrons in the output layer corresponds to the number of classes a sample can be classified into. The activation function applied to the logits is the softmax function:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}. \tag{2.5}$$

This function transforms the N logits in a probability distribution. The sample x will be classified into the class corresponding to the one with the highest probability.

Despite their simplicity, MLPs have limitations in terms of computational requirements when applied to high-dimensional data such as images. Due to their design, MLP networks require a large number of parameters, making them inefficient for learning and finding patterns in this type of data. Moreover, the input layer of MLPs requires a 1D vector, which means that unstructured data with spatial information, such as images, needs to be flattened before feeding it to the network, resulting in the loss of positional information. Another issue with MLPs is their inability to learn invariant representations of the input. For instance, a rotated image can yield a completely different prediction from the neural network. These limitations have led to MLPs being used as simple building blocks for more complex neural networks.

Convolutional Neural Networks

Due to their limitations with unstructured and high-dimensional data, MLPs have been replaced by Convolutional Neural Networks (CNNs). CNNs are specifically designed to handle high-dimensional data, such as images, without the need for data flattening. This advancement is made possible by the introduction of two fundamental building blocks: convolutional layers and pooling layers. CNNs consist of blocks that typically contain multiple convolutional layers followed by pooling layers. Convolutional layers perform mathematical operations called convolutions between the layer's filters and the input data. This allows them to capture hidden patterns and features while preserving the spatial information of the input sample. On the other hand, pooling layers summarize the local features within a window, reducing the spatial dimensions of the data while retaining important information. One of the main strengths of CNNs is their depth, which enables them to capture both low-level features (e.g., edges or colors) in the initial layers and complex high-level features in the subsequent layers. Depending on the specific task at hand, MLPs are often employed as the last layer of a CNN, commonly referred to as the "classifier". The alternation of convolutional and pooling layers progressively summarizes the information contained in the input sample into feature maps of reduced dimensions. Once the dimensions of the feature maps are sufficiently small, they are typically flattened and fed into the MLP classifier to obtain predictions.

The main component of CNNs is the discrete convolution. This mathematical linear operator convolve an input signal $q = [q_0, \dots, q_{n-1}]$ and filter or kernel signal $h = [h_0, \dots, h_{k-1}]$ obtaining an output signal $z = [z_0, \dots, z_{n+k-1}]$ as:

$$\begin{aligned} z &= q \star h \\ z_u &= \sum_{s=0}^{k-1} h_s \cdot q_{u+s}. \end{aligned} \tag{2.6}$$

When dealing with high-dimensional signals such as images, the convolution between 1D signals can be extended to 2D signals:

$$z_{u,v} = \sum_{t=0}^{k-1} \sum_{s=0}^{k-1} h_{s,t} \cdot q_{u+s,v+t} \tag{2.7}$$

where q is an input image and h is a $k \times k$ kernel. As it is designed, the convolution operator provides as output a signal z that is smaller than the input. This problem is amplified when multiple convolutional layers are involved in a CNN: in fact, stacking several convolutional layers without addressing this, will result in making the later feature maps disappear. For addressing this problem, it is possible to pad the input image. This procedure allows to add a series of not informative values (usually zeros) in the border of the input image. Depending on the number of zeros that it is possible to add to the border of the image, the dimension of the output z can be smaller than the input (no padding), equal to the input (adding $\frac{k-1}{2}$ zeros on each side) or greater than he input (adding $k-1$ zeros to each side). The convolution and the padding strategies can be easily extended to multi-channel images such as RGB images (where the number of channels is $C = 3$). In this case, given an image $C \times H \times W$ and C filters $k \times k$, the output of the convolution is a single-channel image where the values are obtained summing the C convolutions between each channel of the image and the corresponding filter. It is also possible to have a multi-channel filter $C \times k \times k \times p$, in this case, each element of the output is obtained as:

$$z_{u,v,d} = \sum_{m=0}^{C-1} \sum_{t=0}^{k-1} \sum_{s=0}^{k-1} h_{s,t,d,m} \cdot q_{u+s,v+t,m} \tag{2.8}$$

A possible alternative for reducing the time required to perform the convolutions between the input image and the kernels is the strided convolution. In this kind of convolution, instead of moving the kernels over all the input image (so moving the kernel by one pixel at a time), some of the pixels, s, are skipped. Of course, the dimension of the output of the convolutions is reduced by a s^2 factor. The formula for strided convolution is:

$$z_{u,v,d} = \sum_{m=0}^{C-1} \sum_{t=0}^{k-1} \sum_{s=0}^{k-1} h_{r,t,d,m} \cdot q_{u\cdot s+r,v\cdot s+t,m}. \tag{2.9}$$

Convolutional neural networks, like MLPs, are an example of feed-forward neural networks. These networks, composed of several stacked layers, make the input data pass through all the layers and make a prediction. As for the linear behavior of the perceptrons, also the convolutions are linear operators, for this reason, one of the previously presented activation functions $a(z)$ is required for modeling complex non-linear functions. Each convolutional layer l takes as input the output of the convolutional layer $l-1$ and applies the convolution adding at the end a bias for each channel of the kernel:

$$z^{(l)} = z^{(l-1)} \star w^{(l)} + b^{(l)}. \tag{2.10}$$

As previously explained, CNNs are composed not only of convolutional layers and the MLP final layer but also of pooling layers. Pooling is a summarizing operation (peaking the maximum, the minimum, the average, etc.) on the elements of feature maps that fall inside a moving window. For example, if the pooling operator is the average pooling with a window of $s \times s$ pixels, the first $s \times s$ elements of the feature maps are substituted by their average, then the next $s \times s$ window, and so on. This operator compensates for the increase in the number of channels in the feature maps while passing to deeper layers of the CNN. It also speeds up training. However, with this kind of summarization, spatial information is lost.

Another component that can be inserted in a CNN (as well as in MLPs), is the dropout layer. For each neuron, with probability p dropout deactivates it by setting its activation to zero. With probability $1-p$ the activation is multiplied by $1/(1-p)$. The goal of this layer is to avoid overfitting and to improve the generalization ability of the network. Dropout layers are used during the training phase but not during the testing phase.

The last typical layer composing a CNN is normalization. The goal of this layer is to make the training of a neural network faster, more stable, and more accurate. In classical machine learning algorithms, normalization is applied before feeding the data to the model. However, in deep neural networks, the running statistics of the data after each layer change during training. One of the first solutions proposed is batch normalization [79]. In this kind of normalization technique, a batch of data is normalized computing its mean and variance before forwarding it to the next layer. In contrast to pre-input normalization, the statistics on the batch are computed during training. Other normalization techniques are group normalization [173], layer normalization [2], and instance normalization [162].

One of the first working CNNs has been the LeNet [93] composed of only convolutional, pooling, and linear layers. Despite its simplicity, the key building blocks of this network have been extensively used and improved. For example in [90], the authors expanded the LeNet architecture by increasing the number of layers to deal with images of higher resolution. Moreover, the authors inserted dropout layers to improve generalization. This network has been subsequently analyzed, optimized, and improved in [153]. Other notable models are [157] and [71]. In particular the latter, called ResNet, is still heavily used nowadays. In fact thanks to his residual mechanism, it has been shown that a very deep convolutional model can be easily trained avoiding overfitting. These ResNets and their residual mechanism are still nowadays the reference models in the field of convolutional neural networks.

Recurrent Neural Networks

Despite the ability of CNNs in working with grid-shaped data such as images, their effectiveness is limited when dealing with sequences of data of variable length. In domains such as time series or natural language processing where the sequential processing of the data is crucial, CNNs are limited by the fixed size of the convolutional kernels. Recurrent Neural Networks (RNN), instead, are able to process whole sequences, independently in their lengths.

When dealing with sequences, we can define a data sample as a sequence of vectors $\mathbf{x}_0, \ldots, \mathbf{x}_{T-1}$. These vectors are forwarded to the RNN one at a time. For each of them,

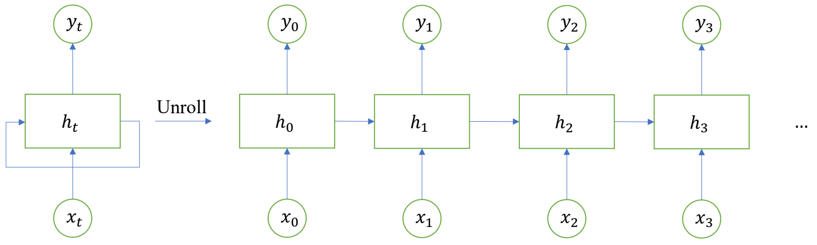


Figure 2.3: Example of a basic RNN module. A single RNN cell can be unrolled to observe the behavior: the hidden state at time t, takes as input the actual input x_t and the previous hidden state h_{t-1}.

$\mathbf{x}_t$, the RNN provides an output $\hat{\mathbf{y}}_t$ as:

$$\hat{\mathbf{y}}_t = a_y(W_y\mathbf{h}_t + \mathbf{b}_y), \tag{2.11}$$

where $\mathbf{h}_t$ is the hidden state. This vector is computed by combining information from the actual input x_t and the previous hidden state $\mathbf{h}_{t-1}$:

$$\mathbf{h}_t = a_h(W_h x_t + U_h h_{t-1} + b_h). \tag{2.12}$$

This hidden state mechanism allows the network to encode in some way, the previous computations and form some sort of memory. In Figure2.3 is it possible to observe the composition and explanation of a basic RNN and how it works. Despite their theoretical ability to remember and connect relevant parts of the sequence observed earlier, RNNs struggle in learning long-term dependencies. In fact, when relevant information to predict the next output are temporally far, RNNs have shown low performances. This is due to the problem of “exploding/vanishing gradient”. In fact, when the gradients are computed in order to backpropagate the error, even for short sequences, the gradient of the matrix U_h can either become 0 or ∞.

The problem has been solved thanks to Long Short Term Memory networks (LSTM). Thanks to a new cell state and a gating mechanism, the important information previously observed is kept while useless information is ignored. Then the cell state is used to update the hidden state with only useful information and the output $\hat{\mathbf{y}}_t$ is produced. In the upper part of Figure 2.4 the inner composition of a LSTM is reported. However, compared to RNNs, LSTM are more complex and requires more computational resources for training.

For this reason, a simplified has been developed starting from LSTMs. In the Gated Recurrent Unit (GRU), the cell state has been replaced by a reset gate and

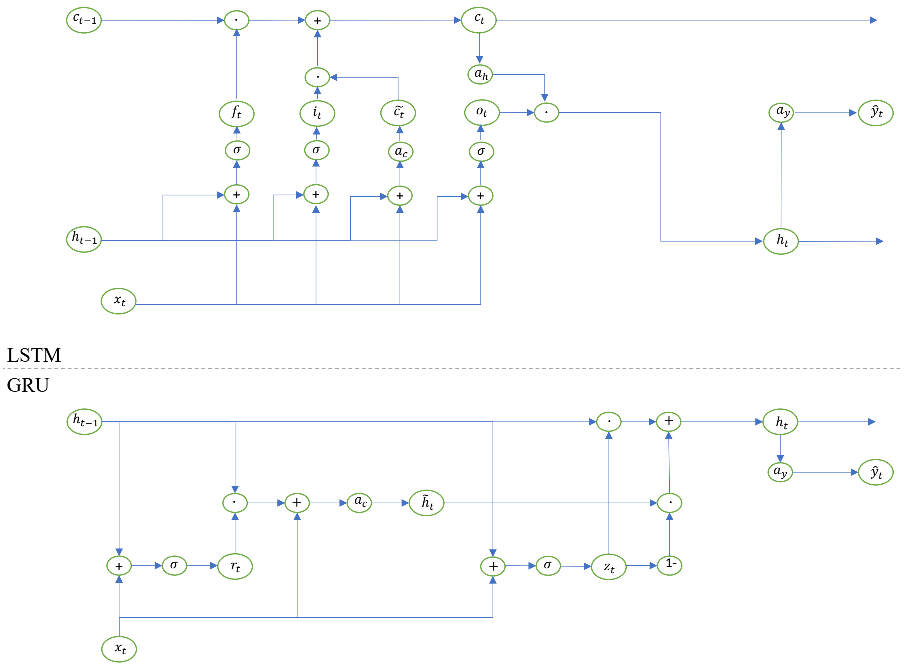


Figure 2.4: Comparison between LSTM (upper) and GRU (lower) architectures.

update gate. These two gates are used to mix the previous and the candidate states and to modulate the previous state to obtain the newer ones. In the lower part of Figure 2.4 its inner structure is reported.

One of the most popular applications of RNNs is sequence-to-sequence prediction. In this kind of task, both the input and the output are sequences. For example, language translation or question answering are sequence-to-sequence tasks. When dealing with such kinds of problems, the most used architecture is an encoder-decoder. The encoder is typically an RNN that takes the input sequence and produces a representation of its final hidden state. Then, the decoder, still an RNN, process this vector and generate the output sequence. The input and output sequences can be of different lengths. However one of the drawbacks of this kind of architecture is that the hidden state can be too small to completely summarize the input vector. In [3], a novel attention mechanism able to map part of the last hidden state to the output sequence has been proposed. In fact, thank to this, it is possible to analyze which was part of the hidden state (and intrinsically of the input sequences) that allows the decoder to make a particular prediction in the output sequence. This attention mechanism basically makes a weighted average of the hidden states computed using the input sequence.

RNNs are considered complex models that require long training times when the sequences analyzed are very long. In fact, due to the necessity of sequentially analyzing the input sequence, the parallelization of the training of the RNNs is hard. Moreover, during training RNNs require fixed-length input sequences, with the necessity of padding or truncating them when they are shorter or longer than the set dimension. Thanks to Transformers these limitations have been overcome.

Transformers

The first work presenting the Transformers is [164]. As it is possible to observe from Figure 2.5, it is composed of several blocks. Each element of the input sequence is projected in a high dimensional space using the input embedding. The space into which the sequence has been projected strongly depends on the application. For example for text applications, Glove and word2vec are very famous embedding functions that try to group words that are semantically similar. Once the input sequence is projected, a positional encoding is applied. This positional encoding assigns to each element of the sequence a unique identifier in order to retrieve later what was the position of the element in the sequence. This was inserted due to the absence of any recurrent or convolutional block that can embed in some way the information about the position of each element of the sequence. In the original paper the positional encoding was sinusoidal:

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{model}}), \end{aligned} \tag{2.13}$$

where pos is the position and i the dimension of the embedding. From now on, elements of the input sequence will be called tokens. The tokens are then forwarded to N encoder blocks. Each of these blocks is composed of a Multi-Head attention module, normalization layers, and a feed-forward layer. The Multi-head attention module is composed of H self-attention modules where each of them is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.14}$$

where Q, K, V are the query, key, and values. The goal of Attention is to map a Query and a Key-Value pair to an output. This output is the weighted sum of the values, where each weight identify how the key is related to the query. The idea behind this is that, given a query and a series of Key-Value pairs, the output is a probability

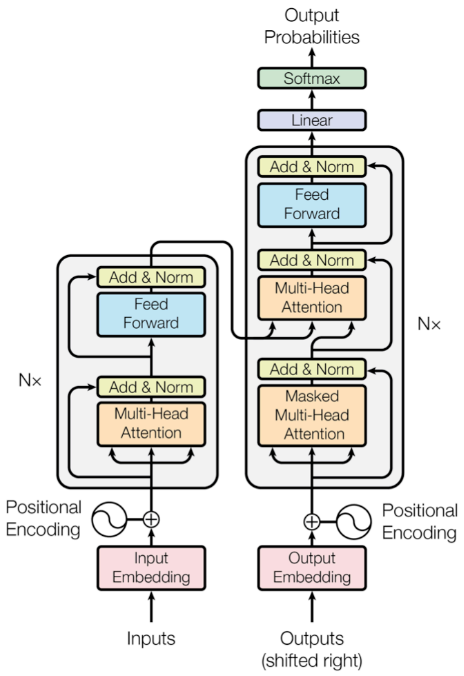


Figure 2.5: Original transformer model. Illustration taken from [164].

distribution of the matching between the query and the key. d_k is the dimension of the queries and keys while the values have dimension d_v. In the case of multi-head self-attention:

$$\begin{aligned}\text{MultiHead}(Q,K,V) &= \text{Concat}(\text{head}_1,\ldots,\text{head}_h)W_0\\ \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V),\end{aligned} \tag{2.15}$$

where each of the matrices have dimension: $W_i^Q \in \mathbb{R}^{d_{mod}\times d_k}$, $W_i^K \in \mathbb{R}^{d_{mod}\times d_k}$, $W_i^V \in \mathbb{R}^{d_{mod}\times d_v}$ and $W^0 \in \mathbb{R}^{hd_v\times d_mod}$ where d_mod is the dimension of the input tokens and h is the number of heads of attention. The output of this mechanism is normalized and added to the input using a residual connection. The multi-head attention mechanism emulates the attention mechanism in sequence to sequence RNNs. The feed-forward network is a two-layer MLP with a ReLU activation function. Its output is then summed with its input and normalized. The output of the encoder is then forwarded to the decoder which is composed of N layers. The structure of these layers is equal to the

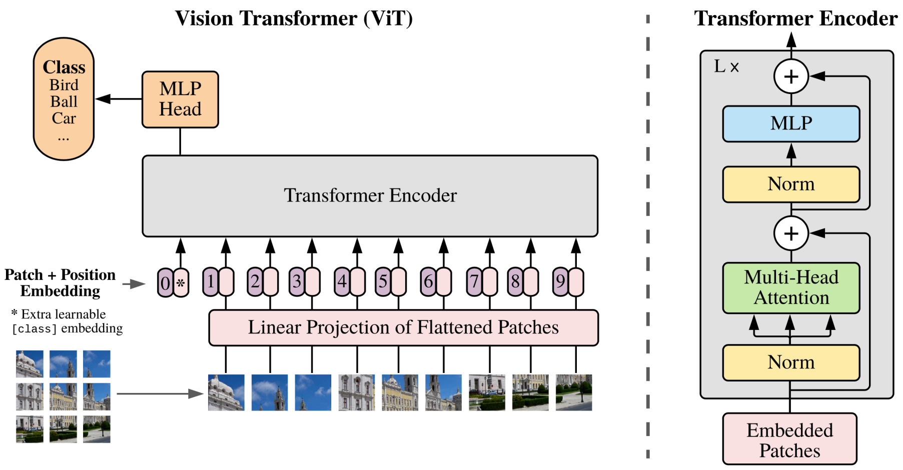


Figure 2.6: Vision Transformer Architecture. Illustration taken from [52].

encoder ones but with an additional layer that computes the multi-head self-attention on the output of the encoder. With respect to RNNs, these models can be trained easily in parallel. In fact, transformers directly process the entire sequence without any recurrent mechanism. Moreover, they showed higher performances in working with longer sequences with respect to RNNs, thanks to their ability to capture long-term dependencies.

Recently, Transformers, have been applied also to computer vision tasks. In [52], a new model, Vision Transformer (ViT) has been presented. In this model, an image x of dimension $C \times H \times W$, is split in patches of dimension $p \times p$. These patches are then flattened obtaining $\mathbf{x}_p \in \mathcal{R}^{N \times (p^2 \cdot C)}$, where $N = HW/p^2$ is the number of patches. In this way, the original image is converted into a sequence of N vectors $p^2 \cdot C$. These vectors, or tokens, are then linearly projected in a D-dimensional space. Additionally, a learnable class token is prefixed to the sequence of tokens in the D-dimensional space and a positional embedding is applied to the entire new sequence. The ViT is an encoder-only architecture. In fact, differently from the architecture previously presented, here only the encoder is present. The patches are then forwarded to a stack of transformer encoder layers. Once obtained the output of the last layer, which is still a sequence of tokens, the class token is extracted and mapped, using an MLP, into one of the classes that have to be recognized. In Figure 2.6, the complete ViT architecture is reported.

Despite transformers having a less inductive bias with respect to CNNS due to their loss of 2D information of the input images, they outperform CNNs in a variety of scenarios and applications such as image segmentation, classification, detection, etc. However, with respect to CNNs, transformers usually require more data to train and avoid overfitting.

Generative Models

Until now, we mostly focused on the inner composition of state-of-the-art neural networks independently on the applications. In fact, all the neural networks described above can be applied, with some adjustment, to perform classification, segmentation, translation, and other tasks. However, these models can be composed and specifically trained in order to perform a generative task. In this setting, the neural network is used to generate new synthetic data on the basis of the data used to train it. The main generative architectures are four: Generative Adversarial Networks(GANs), Variational AutoEncoders(VAEs), Flow-based Generative Models, and Diffusion Models. In this section, we will focus only on GANs, since they are the only generative models covered in this thesis.

Generative Adversarial Networks, have been introduced in [65]. GANs, are based on two neural networks, a generator G and a discriminator D. The generator takes as input a random noise z and provides as output synthetic data $G(z)$. Ideally, the job of the generator is to produce data $G(z)$ that have the same distribution of real data x. But how does the generator can generate data similar to some real data x if it does not observe these real samples? The data generated with G from the random noise is then forwarded to the discriminator. The Discriminator in fact is trained on both the real data x and the "fake" data $G(z)$ to distinguish them. The generator continuously refines its parameters in order to generate data such that $G(z) \sim x$ and the discriminator refine its ones in order to recognize the new fake data. This game between the discriminator and generator stops when the generator is able to generate data that are completely similar to real data and the discriminator is not able to distinguish real data from fake data. In Figure 2.7 the architecture of the GANs is reported. Usually, the discriminator is a binary classifier outputting the probability that a the data has been received as input is real or fake. The discriminator is trained using a binary cross entropy:

$$\min_D\{-y\log D(x) - (1-y)\log(1-D(x))\}. \tag{2.16}$$

The generator instead, samples the noise $z \sim \mathcal{N}(0,1)$ and wants that the probability of the discriminator $D(G(z))$ is as close as possible to one. So the parameters of the generator are updated with:

$$\max_G\{-\log(1-D(G(z)))\}. \tag{2.17}$$

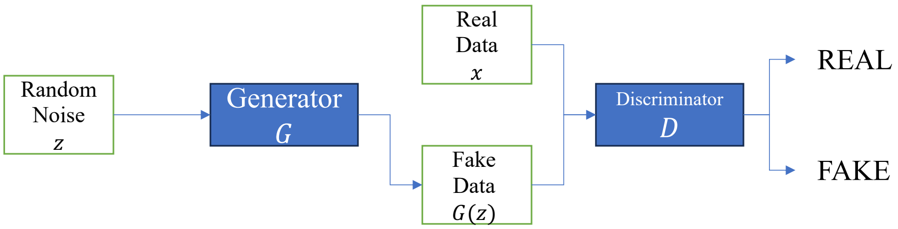


Figure 2.7: Generative Adversarial Networks Architecture

The final loss function that the generator and the discriminator have to optimize is:

$$\min_D \max_G \{-E_x \log D(x) - E_z \log(1 - D(G(z)))\}, \tag{2.18}$$

that summarizes the game that should be solved by these two actors.

2.2 Reinforcement Learning[2]

Along with supervised learning, another important learning paradigm is Reinforcement Learning (RL). Differently from supervised ML, RL is an interactive kind of process where a learner discovers what are the best actions to take in each state where it can be. In this learning paradigm, the learner does not learn to find patterns from data or hidden structures, as it happens in supervised learning or unsupervised learning, but it explores the space of possible states and actions exploiting the situations that can provide it a high reward.

2.2.1 Reinforcement Learning in a nutshell

The main character of RL is an agent, i.e. the learner, that interacts with the environment that surrounds him. At every step, the agent can be in a state S_t, and it can take an action A_t. This action modifies the environment moving the agent to a new state S_{t+1}. At the same time, the environment provides a reward R_{t+1} quantifying the goodness of the action. This process continues until a final state S_T is found (finite Markov Decision Process, MDP). Figure 2.8 a summarizes the Reinforcement Learning schema for finite sequential decision making processes. The sequence of actions, state, rewards, actions, state . . . is called trajectory.

When dealing with finite MDP, the number of states, rewards, and actions is finite.

[2]This section is based on [43, 124, 125, 156, 192]

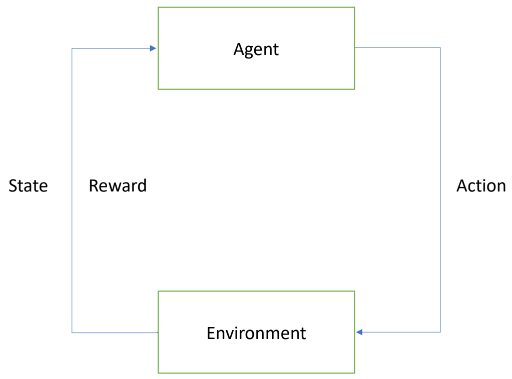


Figure 2.8: RL Schema.

This means that each of the possible state and action has a discrete probability that depends only on the previous action and state:

$$p(s', r|s, a) = P(S_t = s', R_t = r|s_{t-1} = s, A_{t-1} = a), \tag{2.19}$$

where $s, s' \in \mathcal{S}$, $r \in \mathcal{R}$ and $a \in A(s)$ ($\mathcal{S}$ is the states space, $\mathcal{R}$ is the rewards space and $\mathcal{A}$ is the actions space). Being p, a probability function, and considering the finite MDP setting, the sum of the ps computed for all the possible values of the states and the rewards is one. Starting from p, it is possible to compute the state-transition probability function as

$$p(s'|s, a) = p(S_t = s'|S_{t_1} = s, A_{t-1} = a) = \sum_{r \in \mathcal{R}} p(s', r|s, a). \tag{2.20}$$

Other two relevant functions that can be computed are the expected reward for state-action and the expected reward for state-action-next-state:

$$\begin{aligned} r(s, a) &= \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} \sum_{s' \in \mathcal{S}} p(s', r|s, a) \\ r(s, s', a) &= \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a, S_t = s] = \sum_{r \in \mathcal{R}} \frac{p(s'|s, a)}{p(s', r|s, a)}. \end{aligned} \tag{2.21}$$

Typically in RL, the main objective of the agent is not to maximize the next reward but to maximize the sum of the rewards that it can obtain until the end of the MDP.

This quantity, called return, is denoted by G_t and is defined as:

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T. \tag{2.22}$$

In this case, the return is computed at the end of the MDP or, as it is called, at the end of an episode (episodic task setting). When dealing with a setting without a final state, it is called continuing task and the time horizon is $T = \infty$. Moreover, the return can be discounted by a factor $\gamma \in [0, 1]$,

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{2.23}$$

This discounted version can be useful to weigh the value of the future rewards with respect to the actual. If $\gamma = 0$, just the next reward is important for the agent otherwise, the agent has to take into account the next rewards. When $\gamma = 1$, all future rewards weigh as the next one.

Two fundamental concepts in Reinforcement Learning (RL) are value functions and policies. Value functions, both for states and state-action pairs, quantify the desirability of a particular state or the advantage of taking a specific action in a given state. These functions provide insights into the quality of the state where the agent currently resides and the value associated with taking a particular action in that state. On the other hand, policies represent a probability distribution over the available actions that the agent can take, denoted as $\pi(a|s)$. Policies define the likelihood of the agent selecting a specific action when presented with a particular state. Together, these concepts form the foundation of RL algorithms, enabling agents to make informed decisions based on the value of states and the probability distribution of actions. The value function of a state s following a policy π, is:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s], \tag{2.24}$$

that is the expected return computed starting from s and following the policy π. This can be defined as the state-value function for a determined policy π. In the same way, the action-value function for the policy π can be defined as:

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a]. \tag{2.25}$$

The state-value function can be combined with the action-value function:

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s, a). \tag{2.26}$$

Given that the return can be computed as the discounted sum of the future rewards, the action-value function can be redefined as:

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi[R_t + \gamma R_{t+1} + \ldots)|S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_t + \gamma(R_{t+1} + \ldots)|S_t = s, A_t = a] \\ &= \tilde{R}(s,a) + \gamma \mathbb{E}_{S_{t+1}}[v_\pi(s_{t+1})|S_t = s, A_t = a], \end{aligned} \tag{2.27}$$

where the next state, is conditioned only by the actual state and the action taken from this state following π. The value $\tilde{R}(s,a)$ identifies the expected value of $R(s,a)$

In order to maximize the return in an MDP, the agent has to follow the best possible policy. But how can this be defined? A policy π' is better than another policy π if, for every state, the expected return for π' is greater or equal to the one obtained with π. This can be translated in $\pi' \geq \pi$ iff $v_{\pi'}(s) \geq v_\pi(s)$. The best possible policy is defined as the optimal policy π^*. Similarly, the optimal state-value function and the optimal action-value function can be defined as:

$$\begin{aligned} v_*(s) &= \max_\pi v_\pi(s) \\ q_*(s,a) &= \max_\pi q_\pi(s,a) \end{aligned} \tag{2.28}$$

with

$$v_*(s) = \max_{a \in A} q_*(s,a). \tag{2.29}$$

By composing these equations we can obtain the Bellman optimality equation:

$$q_*(s,a) = \tilde{R}(s,a) + \gamma \mathbb{E}_{S_{t+1}}[\max_{a' \in A} q_*(s_{t+1}, a')|S_t = s, A_t = a]. \tag{2.30}$$

In the case of finite MDP scenarios, this equations can be solved by obtaining the optimal value function q_* and from that π_*. The optimal policy is a policy that is acting greedy with respect to q_*. In the same way, also v_* can be obtained. The Bellman equation is an easy and consistent solution for finite MDPs with a complete knowledge of the dynamics of the environment. When the dynamics of the environment are not known or when the number of states becomes higher, the computational requirements for solving this system of equations grow as well. When this happens, the solutions that can be used are approximations of the real solution. In the next section RL algorithms with the goal of approximating exact solutions, are presented.

2.2.2 RL Algorithms

From what we observed so far, the knowledge of the environment is vital for solving an MDP. However, the states composing the environment could be not always observable, i.e. the dynamic of the environment is not always known, and this could be a strong limitation. Moreover, when dealing with a complex system, the modeling of the environment could not perfectly emulate its dynamic (due to uncertainty errors, etc). For these reason the RL algorithms can be divided into model-free and model-based algorithms.

Model-Free RL algorithms

In model-free algorithms, the agent interacts with the environment without any knowledge of the dynamics. In fact, the agent takes an action and obtains a reward. On the basis of this reward, it will update its functions (state and action) but without any new information on how the environment is modeled etc. It is a pure trial-error scenario. In this family of algorithms, two main categories can be identified: value-based methods and policy-based methods.

In value-based methods, the goal is to estimate the value-function (state-value or action-value depending on the objective) for allowing the agent to correctly evaluate the state in which it is at the moment and select the next promising action. Two of the most effective algorithms are SARSA and Q-Learning. These algorithms use a function Q for estimating the action-value function q_π.

The first, SARSA, is an on-policy algorithm (it iteratively redefines its policy and uses it for selecting the actions). This algorithm updates Q as

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{2.31}$$

after every transition from state-action pair to state-action pair. The algorithm is called in this way since it makes use of $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$. The policy π is adjusted in the function Q.

Q-Learning is an off-policy algorithm. This means that the value function is updated on the basis of a policy (behavioral policy) that is not the one that has to be learned (target policy). Differently from SARSA, Q-Learning learns the function Q approximating directly the optimal action-value function q_*. The update rule for the function Q is:

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]. \tag{2.32}$$

The other branch of model-free RL algorithms is the Policy-based algorithms family. In this type of method, the goal is to learn a parameterized policy $\pi(a|s, \theta)$ that can lead the agent to maximize the cumulative rewards, without using a value function. The parameters θ of the policy are optimized maximizing an objective function $J(\theta)$ as follows:

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t). \tag{2.33}$$

Thanks to the policy-gradient theorem, it is possible to approximate the gradient of the objective function $J(\theta)$ can be approximated by $\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\theta)$ where μ is the state distribution. Two of the most famous policy-based algorithms, REINFORCE and Actor-Critic, are based on this parameter update.

In the REINFORCE algorithm, following the target policy π, the gradient of $J(\theta)$ can be exactly computed as:

$$\nabla J(\theta) = \mathbb{E}_\pi[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\theta)]. \tag{2.34}$$

This allows us to compute the update of the parameters θ using gradient ascent:

$$\theta_{t+1} = \theta_t + \alpha \sum_a \hat{q}(S_t,a,w) \nabla \pi(a|S_t,\theta), \tag{2.35}$$

where $\hat{q}$ is an approximation of q_π. Multiplying and dividing by $\pi(a|S_t,\theta)$ and following π, it is possible to obtain:

$$\nabla J(\theta) = \mathbb{E}_\pi[G_t \frac{\nabla \pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta)}], \tag{2.36}$$

where G_t is the return. This term can be inserted in the update of the weights obtaining:

$$\theta_{t+1} = \theta_t + \alpha[G_t \frac{\nabla \pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}]. \tag{2.37}$$

The weights update is proportional to the return and to the gradient of the probability of taking an action A_t normalized by its probability. One of the issue of the REINFORCE algorithm, is that it is a Monte Carlo algorithm, which means that it needs to wait until the end of the episode for computing the return and then update the policy's weights.

As an alternative to the REINFORCE algorithm, the second algorithm instead, the Actor-Critic, does not wait for the end of the episode. It makes instead an estimation of q_π using a function estimator q_ω. The two main components of the algorithm are the policy π_θ and the critic q_ω that evaluates the choices taken by the policy. The weights update for both the actor and the critic are the following:

$$\begin{aligned}\theta_{t+1} &= \theta_t + \alpha(R_{t+1} + \gamma q_\omega(S_{t+1}) - q_\omega(S_t))\frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)} \\ \omega_{t+1} &= \omega_t + \eta(R_{t+1} + \gamma q_\omega(S_{t+1}) - q_\omega(S_t))\nabla q_\omega(S_t).\end{aligned} \tag{2.38}$$

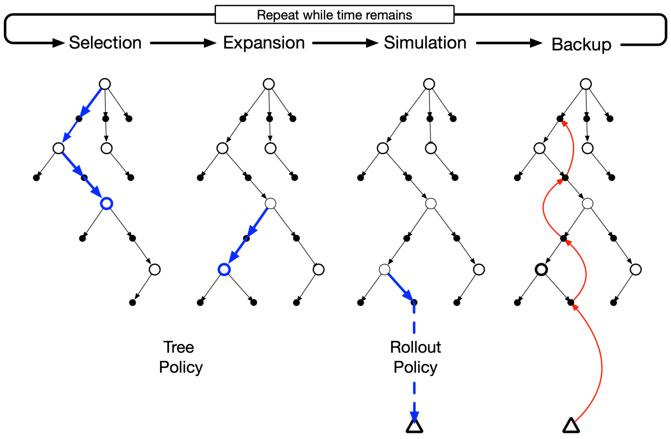


Figure 2.9: Monte Carlo Tree Search phases. Illustration taken from [156].

Model-Based RL algorithms

The second major family of algorithms in RL, are the model-based algorithms. In this family of algorithms, the main component is the model of the environment, i.e. the dynamic of the environment is known or could be learned. This model can be used for planning, i.e. iteratively refine a policy that can be used by the agent to follow the path that provides it the highest cumulative reward. This definition is in contrast with the planning over the state space that is common in heuristic search algorithms such as $A*$, Best-First Search, etc. In fact, in heuristic search, the approximated value function is not updated during the search, but it is defined a-priori. However, the heuristic search can be treated as an extension of the ϵ-greedy algorithm with a longer horizon. Another example of RL algorithms that can be used to obtain optimal policy, when the model of the environment is perfectly known, is the dynamic programming algorithms. Two of the most successful dynamic programming algorithms are the policy iteration and the value iteration. Despite their effectiveness, dynamic programming algorithms are not suitable for scenarios where the number of states is extremely high. However, these algorithms are more efficient with respect to linear programming and direct search.

One of the most famous model-based RL algorithms is the Monte Carlo Tree Search (MCTS). This rollout algorithm, based on the Monte Carlo Control Theory, performs simulations in order to update the estimation of the value of a state and guide future simulations toward the most promising states. Given a new state (or the initial), a policy, called tree policy, is used to select a leaf node (selection). This leaf node is expanded adding some or all the child nodes (expansion). Starting from the leaf node selected or from one of its child nodes, a simulation is performed until a terminal node is reached using the rollout policy (simulation). When the terminal node is reached, the total return obtained via the simulation is backed up to the root node updating the action values of the traversed added nodes (the values of the nodes explored during the simulation are not updated; backup). These four steps are repeated until a total number of iterations is reached. When this happens, the most promising action is selected starting from the root node. The resulting new node is then the new root of the algorithm and then the complete process is repeated again. The complete algorithm terminates when a terminal node is selected. Figure 2.9 shows the four MCTS steps composing one iteration.

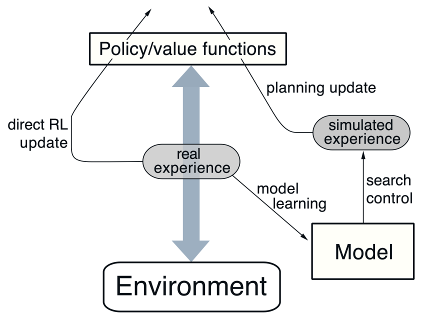


Figure 2.10: Dyna agent schema. Illustration taken from [156].

In other cases, when the dynamics of the model, are not completely known, other model-based algorithms can interact with the model and iteratively refine its dynamics, while at the same time they can learn a policy. A basic example is the Dyna family of algorithms. Figure 2.10 provides an example of the Dyna agent. In this algorithm, the planning is performed online while the agent is discovering the environment interacting with it. This algorithm implements Q-Learning for planning while at the same time, on the basis of the interactions, the new model of the environment can be used to update and correct the Q-Learning predictions of the value functions. In fact, from the Figure, it is possible to observe that there are two kinds of experiences, a real one, obtained by interacting with the environment, and a simulated one obtained by applying search control theory on the learned model. So, these two experiences are used to update directly the value functions or the policy.

2.2.3 Deep Reinforcement Learning

In the last years, with the expansion of deep learning algorithms and the higher availability of computational resources, a new branch of RL, called Deep Reinforcement Learning (DRL), that makes use of neural networks as function approximators, has gained popularity. The usage of neural models allowed classical reinforcement learning algorithms to deal with more complex and wider state-action spaces and at the same time gave the possibility to the development of more efficient algorithms.

One of the most famous examples is Deep Q-Network (DQN)[124]: This algorithm combines the power of neural networks with the Q-Learning algorithm. In this algorithm, the optimal action-value function Q^* is approximated using a neural network $Q_\theta(s,a)$ with the support of an experience replay buffer that contains, for each time step t, $e_t = (s_t, a_t, t_t, s_{t+1})$ composing a dataset $D_t = \{e_1, \dots, e_t\}$. The neural network is updated minimizing the following

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(D)}[(r + \gamma max_{a'} Q_{\theta_i^-}(s', a') - Q_{\theta_i}(s,a))^2], \tag{2.39}$$

where θ_i^- are the weights of the target network which is aligned with θ_i every k steps. This algorithm has been tested on several Atari games providing as input to the neural network a frame of a game. The neural network output was a probability distribution over the actions that can be taken. The resulting agent reached the ability of a professional human tester.

Another example of the application of deep neural networks to reinforcement learning is the family of AlphaGo[150], AlphaGozero[152], AlphaZero[151] algorithms. These algorithms, based on the MCTS, have shown performance higher with respect to human professional players of several games such as go, chess, and shogi. The first algorithm presented was AlphaGo. This method is composed of two stages: in the first, a rollout policy ρ_π and a policy networkρ_α are trained in a supervised way using a dataset of human expert positions and actions to predict the next moves given as input the current position on the board. Then, in the second stage, called self-play, the policy network ρ_β is used and improved to generate go matches and a value network v_θ is used to evaluate the position and predict the outcome of a game. During the second stage, a modified version of the MCTS algorithm is used to generate games. In this variant, a leaf node is selected following the exploration-exploitation criteria:

from the root of the tree, an action is selected with $a_t = \text{argmax}_a(Q(S_t, a) + \frac{P(s_t,a)}{1+N(s_t,a)})$ where Q is the action value, N is the visit count of the node and P is the probability computed using the policy network ρ_α until a leaf node is selected. This node is then evaluated using the policy network $P(s, a) = \rho_\beta(a|s)$, the value network $v_\theta(s)$ and performing a rollout using the rollout policy $z = \rho_\pi(s)$. Then when a terminal state has been reached, the return is summed with the evaluation provided by the value network ($V(s) = (1 - \lambda)v_\theta(s) + \lambda z$) and it is backed up to the root updating Ns and Qs of each node traversed. The policy network ρ_β is trained through policy-gradient reinforcement learning in order to play games against previous versions of the policy network. The value function instead is trained with supervised learning to predict the outcome of self-played games. One of the main constraints of this algorithm is that it has been pretrained on a dataset of go matches. In AlphaGo Zero instead, the rollout policy and the pretraining, have been excluded. In fact, in this newer version, a joint policy and value network is trained just with self-play reinforcement learning $(\mathbf{p}, v) = f_\theta(s)$. $\mathbf{p}$ is the probability distribution over the possible actions that can be taken from s, v is instead a scalar value representing the probability of the current player winning from the position s. The algorithm is still divided in two phases (that run in parallel): in the first phase, several self-play games are executed where the neural network plays again itself, in the second one, the neural network weights are updated with the generated matches. Specifically, during the self-play, from each position, s_t, an MCTS is executed in order to select the best promising action $a_t\tilde{\pi}_t$. The self-play continues until a match terminates (resignation, time limits, one of the player wins, etc). When this happens, the outcome of the game, r_T is stored. For instant t, a triplet $(s_t, \pi_t, z_t\ )$ is stored (where $z_t = \pm r_T$ depending on the player). These triplets are used to update the neural network during the second phase:

$$l = (z - v)^2 + \pi^T log(p) + c||\theta||^2. \tag{2.40}$$

This updated neural network is then evaluated versus the best player (the best neural network until now) and if this new model wins more than 55% of the games, the best model is then updated and it will be used during the new self-play iterations. Recently, a more general version called AlphaZero, has been proposed. This algorithm is a generalization of the AlphaGo Zero algorithm and it has been applied beyond the game of Go. In this newer version, some limitations were removed like the binary outcome due to the rules of Go or the assumptions abouth the symmetries of the board game. In AlphaZero, the neural network is updated after every training phase without playing any games with any best model. Practically speaking, this algorithm is a more general RL algorithm without any assumption on the scenario that will be faced.

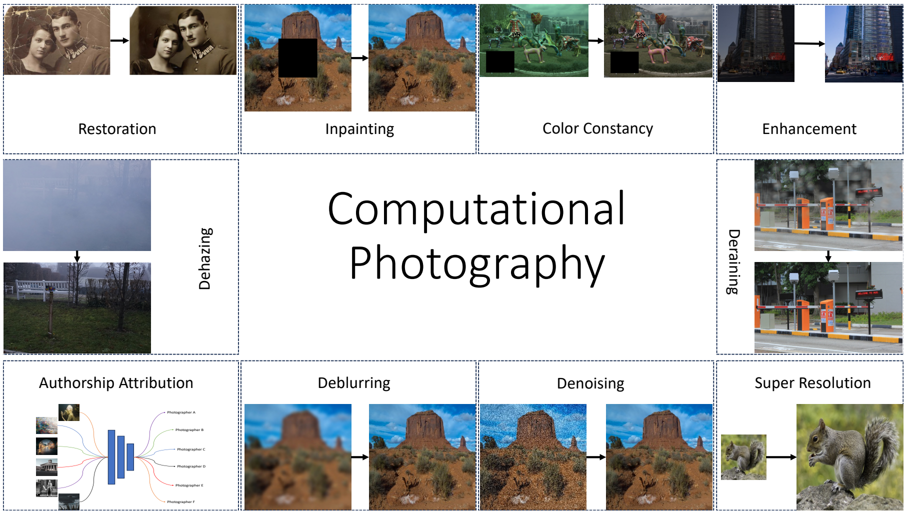


Figure 2.11: Computational Photography tasks overview.

2.3 Computational Photography and Learning Algorithms

Recently, deep learning algorithms in computational photography led to high-quality results in many tasks. Specifically, the intrinsic pattern discovery capabilities of neural networks, coupled with their ability to apply nonlinear transformations to input images, enabled the resolution of complex tasks that were previously addressed using statistical machine learning algorithms with handcrafted features. Furthermore, the increased availability of computational resources facilitated the direct processing of images from datasets using neural models, thereby preserving the spatial information within the images. Deep learning algorithms have also demonstrated their effectiveness in addressing various challenges computational photography poses. These algorithms can effectively deal with illumination variations, noise reduction, and other complex issues. By leveraging large amounts of image data, these methods became able to tackle these complex challenges with high performance and robustness.

Figure 2.11 reports an overview of common tasks in computational photography. In the following sections, tasks that have been explored in this thesis like image enhancement, color constancy, inpainting, photo authorship attribution, and neural style transfer are defined and analyzed. Finally, other relevant tasks like image

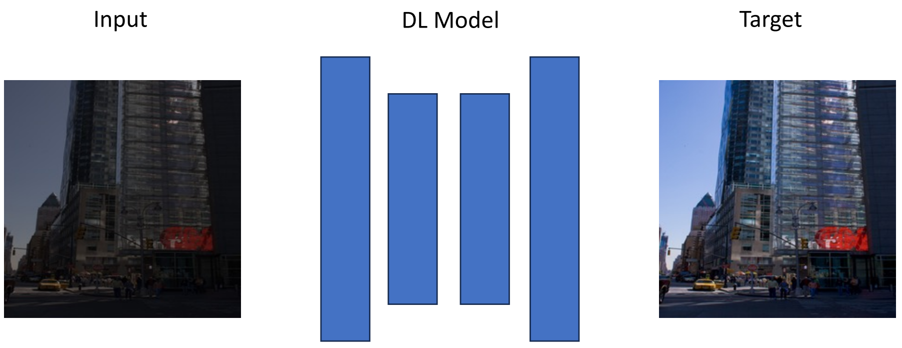


Figure 2.12: Example of the image enhancement task.

denoising, restoration, etc. are briefly described.

2.3.1 Image Enhancement

Image enhancement is a classic problem in image processing in which a low-quality image is transformed in its high-quality version while preserving its visual content. In Figure 2.12 an example of this task is reported. 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). These methods learn by observing thousands of pairs of low-quality/high-quality images. Ravirathinam et al. proposed a multi-context framework based on a modified version of the UNet architecture for low-light image enhancement [135]. 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 from 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 [77]. 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 [80], Cycle-GAN[204], and EnlightenGAN [81] are methods based on an adversarial learning schema that involves a generator and a discriminator.

In an 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 feature extractors for low light images [118]. Lore et al. proposed a deep autoencoder approach for natural low-light image enhancement [112]. A similar approach for underexposed images has been proposed instead by Wang et al.[168]. Zhang proposed a pipeline based on a decomposition network and illumination adjustment to tune the exposure in low-light images [197]. The same approach has been further explored by other research groups in the following years [119, 202]. Liu et al. presented a Retinex rule-based model for low-light image enhancement. The method first defines the neural architecture following the optimization process of Retinex-based models. Then, a two-level search strategy is presented to define suitable architectures for noise removal and illuminant estimation [106]. 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 [25]. 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 [49]. 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 [141]. 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 [9]. Similarly to Cycle-GAN, Sasaki et al. proposed an unpaired approach with denoising diffusion probabilistic models for image-to-image translation [143]. 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. [49], the computational resources required to train and to use this kind of method 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 [12, 13]. Another work from the same authors explores the use of a neural network to estimate the coefficients of splines, that are used as color curves [14]. 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 [194]. 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 [29]. 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 [87]. 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 [69].

Other 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 numerical reward is used to improve the agent. 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 [130]. 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 [181]. 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 [75]. Yu et al. presented a deep q-learning based tool-chain 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. [184]. Furuta et al. presented an actor-critic approach for image processing based on a pixel-wise reward. They applied the method to several image processing tasks such as color enhancement, image restoration, etc. [57]. Yu et al. proposed a mixed approach based on GANs and deep reinforcement learning [185].

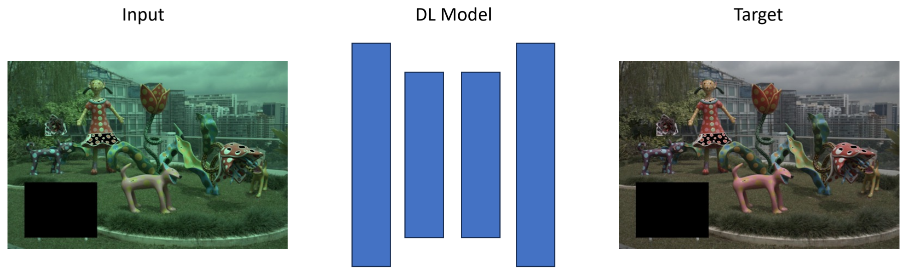


Figure 2.13: Example of the Color Constancy Task.

The use of vision transformers for color transition problems has been proposed by Cai et al. [26] and by Lin et al. [102]. 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 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 patches, applies an embedding, and passes the created vectors through several two-branches transformer modules obtaining the enhanced image [199].

2.3.2 Color Constancy

Color Constancy is a computational photography task, where the goal is to estimate the real colors removing the effect of light sources and illuminant variations. In Figure 2.13 an illustrative example of color constancy, is reported. The approaches that have been presented in the state-of-the-art can be grouped in learning-based and static-based algorithms.

Statistical algorithms are methods that make assumptions and estimates on the illuminant that characterize a scene in a picture on the basis of its statistics like color mean, variance, etc. One of the first algorithms, called Gray-World [21], propose a mathematical model that estimates the illuminant in a scene using a linear combination of fixed bases and standard in the color space. The White Point algorithm [28] presents an illuminant estimation based on the hypothesis that the maximum reflectance in a picture, is achromatic. Practically, the highest pixel value of each channel is considered the brightest. This algorithm is also known as MaxRGB. Similarly, the Shades of Gray algorithm, considers that the $p-$Minkoski norm of a scene is achromatic [54]. In the same way, General Grey World [7], Gray Edge[163] and Second Order Gray Edge[163] make the same assumption about the achromaticity of the scene but apply before a transformation or considering a first/second order derivative of the norm. Other relevant static-based color constancy algorithms are Least Mean Squares Committee[27], No-N-Max (NNM) [17], Gamut Mapping[56], Color by Correlation[53].

The second category of algorithms, the learning based algorithms, are methods that learn how to estimate and correct the illuminant present in a scene from the data. Rosenberg et al. [138] proposed a method based on the Kulleback-Leiber divergence for estimating the global illumination parameters of real-world scenes. Cheng et al. [35] proposed a tree-based method using four simple color features for illuminant estimation. The authors designed the color features and provided them to an ensemble of regression trees. Bianco et al. [15, 16] proposed the first work involving CNNs for the estimation of the illuminant. The authors sampled patches from the images and performed contrast normalization for each patch using histogram stretching. The patches are then forwarded to the CNN and then combined to produce an estimation of the illuminant of the image. Similarly, in [114], the authors proposed a CNN directly working with the whole image instead of splitting it in several patches. Shi et al. [148], proposed a Specialized Network for the estimation of robust local illuminants. This adaptive network is composed of two branches, a hypotheses network, and a selection network. The first makes multiple estimations for the illuminants while the second adaptively selects the most plausible one. Banic et al. [5, 6] presented an unsupervised learning-based method able to learn the parameter of the model after approximating an unknown ground-truth illumination, without calibration. Similarly, in [10], the authors proposed a quasi-unsupervised illuminant estimation method. In this method, a neural network is trained for detecting achromatic pixels in images after their conversion to grayscale. This method, does not make any hypothesis on the illuminant but is based on the assumption that any training image has been already balanced. Finally, Buzzelli et al. [22], presented an unsupervised approach for illuminant estimation based on object recognition. The proposed solution does not make use of a ground truth image, but instead, it aims at improving the performance of object recognition.

2.3.3 Image Inpainting

Another computational photography task is image inpainting. In this kind of task, an image with an occluded region has to be restored filling the area with a reasonable approximation of what is missing. In Figure 2.14, an example of the task is reported. In the last years, deep learning based methods have shown higher performances with respect to traditional approaches. In particular, it is possible to categorize deep learning methods published in the state-of-the-art in end-to-end methods and generative methods.

End-to-end methods, aim to directly learn the mapping between the occluded image

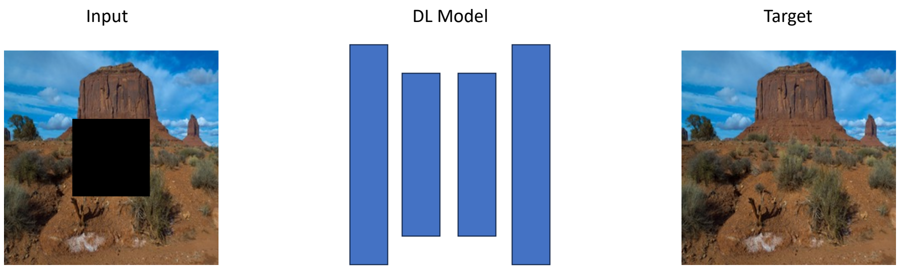


Figure 2.14: Example of the Image Inpainting Task.

and the target. Yang et al. proposed [180] a multi-scale approach for patch synthesis based on a double optimization of the textures and the content of the image. This optimization, obtained using a content and a texture network, ensures the preservation of the content of the image, while, at the same time, is able to highly detailed images. Liu et al. [104] presented a CNN composed of partial convolutional layers, i.e. masked and renormalized convolutional layers. Moreover, the authors proposed a mechanism to update the mask for the next partial convolutional layer directly during the forward pass. In [96], the authors proposed a Recurrent Feature Reasoning (RFR) network and a Knowledge Consistent Attention (KCA) module. The RFR network infers pixels in missing areas starting from the boundaries and then refining iteratively the more central ones. The KCA module, integrated with the RFR, helps to model the long-range dependencies between distant pixels. More recently a masked autoencoder (MAE), has been proposed. This model, composed by an encoder and a decoder, has been pretrained with random masked images to reconstruct it filling the missing pixels. After the pretraining, the decoder is discarded and the encoder is used for downstream tasks. This method was able to reconstruct, with high fidelity, images masked up to 95%.

Generative methods, instead, leverage the ability of generative models of creating high-quality images to perform the inpainting task. Iizuka et al. [78] proposed a fully-convolutional network for consistent image inpainting trained with two discriminator networks (one acting on the global features and the other on the local ones). Similarly, in [183], the authors presented a generative network with a contextual attention layer for image inpainting. The proposed network is composed of a cascade of two stages: the first, called coarse network, makes use of dilated convolutions for a rough reconstruction of the input; the second instead refines the output of the coarse network using the contextual attention layers. Finally, the local and global adversarial losses are computed for training the whole architecture. Zheng et al. [201] proposed a pluralistic approach for image inpainting. This approach generates multiple plausible solutions for filling the missing areas in an image. The method is based on a system of two parallel GANs, one reconstructive and the other generative. Moreover, the authors inserted a memory system for modeling the distant dependencies between the encoder and the decoder to improve the consistency of the final image. Similarly, Cai et al. proposed PiiGAN, [24] for pluralistic image inpainting. Additionally to a GAN architecture, the authors proposed a consistency loss for refining masked images. In [200], the authors presented UCTGAN, an unsupervised cross-space translation GAN. This network is composed of two branches: The first branch uses a manifold and the generative network to learn a mapping between the masked image and the completion image in a manifold space. The second branch is composed of a conditional encoder that acts as a conditional constraint. In [105], a Probabilistic Diverse GAN (PDGAN) for image inpainting has been proposed. Finally, denoising diffusion probabilistic models that deal with the inpainting task have been presented in [116, 141].

2.3.4 Photo Authorship Attribution

Despite the large interest in the automatic recognition of painters, authorship attribution for photography has yet to be thoroughly explored. In [85, 142] classical machine learning models use visual and color features for the recognition of painters. More recently, in [18, 19], the authors tackled the problem of recognizing the artist, artistic school, and genre category in paintings using a deep multi-branch neural network. In [1, 4, 38, 76, 158] the authors presented a variety of deep learning models for predicting the painting, photo, and illustration style, and the artist.

The availability of photographic style datasets allowed the development of deep learning algorithms for photographic style recognition. One of the first publicly available photographic style datasets is Flickr-Style[83]. This dataset is composed of 80k photographs annotated with 20 curated style labels. The style labels depend, for example, on the color of the photos, the composition styles, or the genre. In addition to Flickr-style, the authors provided Wikipainting, a painting style dataset composed of 85k images of paintings grouped in 25 styles. Similarly, AVA has been presented in [127]. The AVA dataset contains more than 250k images. The images have three different types of annotations: aesthetic, semantic, and photographic style annotations. The authors grouped the images in 14 different styles according to the shooting technique achieved by manipulating camera configurations. Wilber et al. proposed the BAM dataset [172], a large dataset composed of 65M of images. The images are labeled based on their type (e.g. photography, comic, pencil, etc.), the image content, and the emotion that the image can transfer to the viewer. Gairola et al. proposed Wall art[58], a photographic style dataset composed of images scraped from the web. The images have been gathered in 13 different style labels. One of the most recent photographic style datasets is Photozilla [154]. This dataset is composed of

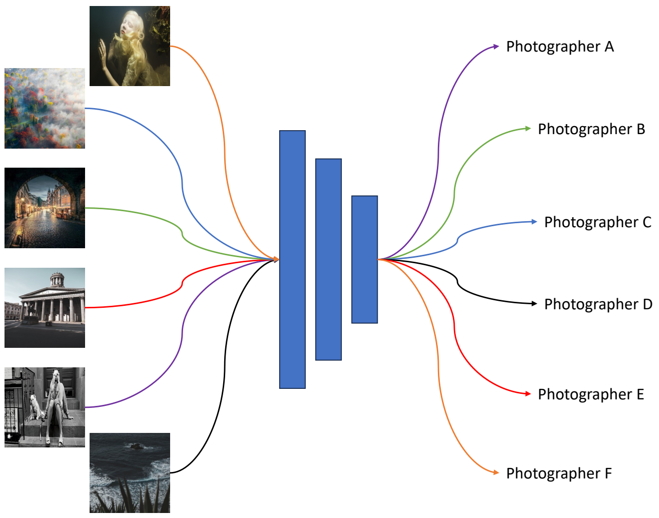


Figure 2.15: Example of the Photo Authorship Attribution Task.

over 990k images divided in 20 photographic styles. Finally, Chen et al. [32] proposed a time-lapse architectural dataset. The over 21k architectural images contained in this dataset are labeled as day, golden, blue, and night on the basis of the daytime when the shot has been taken.

Thomas et al. presented a different kind of photographic dataset, in which the "style" is what distinguishes photos taken by different artists [159]. They proposed a large dataset including photos by 41 photographers. A limitation of this dataset is that most photos predate digital photography or even color photography. Therefore, the contribution of digital post-processing, which is a key factor in today's photography, is not represented. The authors have also been the first contributors trying to recognize photographers defining this task as Photo Authorship Attribution. In Figure 2.15 an example of the photo authorship attribution task is reported.

2.3.5 Neural Style Transfer

Neural style transfer (NST) consists in transferring the style of one image to another. Figure 2.16 reports an example of the NST task. One of the first works in this field is [59]. In this work a style image and a content image are provided as input to a pre-trained VGG19 network and feature maps at several layers are extracted from the two images. An image is then iteratively forwarded to the VGG network and

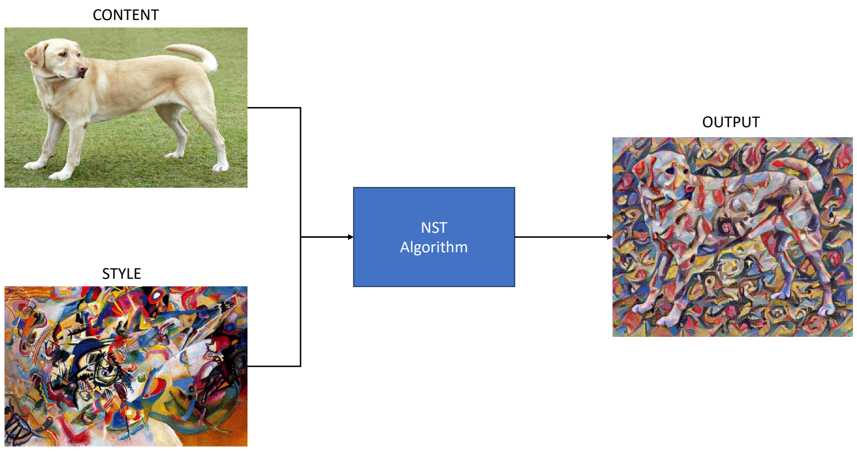


Figure 2.16: Example of the neural style transfer task. Given a content image and a style image, the goal is to apply the style to the content image while preserving its structure.

updated to make its feature maps as close as possible to those from the style and content images. This method inspired other works using neural feature matching to transfer the style to a content image[73, 97, 99, 167]. A different approach has been proposed by Zhu et al. [204]. They defined a generative architecture based on a cycle loss for image-to-image translation of unpaired datasets. It has been tested for transferring the painting style of various artists to content images. However, a potential drawback of this method is that a separate model needs to be trained from scratch for each style. Other style-specific models have been presented in [8, 33, 98, 115]. In recent years, multiple-style or multiple-image style transfer methods have been proposed[170, 191, 198]. For example, He et al. [72], presented a new method for color transfer with images that share perceptually similar semantic structures. Among the applications, the authors presented the results of the application of a multi-image style to a content image. In [169], the authors proposed an interactive method for multi-style transfer. The proposed framework aims to select different parts of the image and apply to each of them the style from a style image using the NST algorithm. Liu et al. proposed a variational autoencoder (VAE) for multi-style transfer [109].

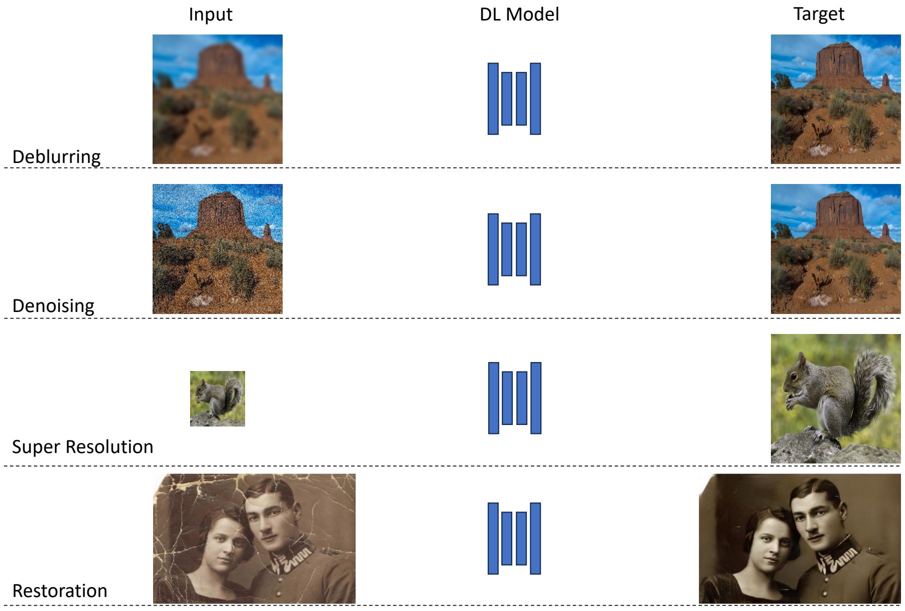


Figure 2.17: Example of other four computational photography tasks.

2.3.6 Other Tasks

Additional relevant tasks in computational photography are, for example, Denoising, Deblurring, super-resolution, restoration, etc. The denoising task aims to remove the noise, i.e. randomly distributed wrong pixels, in the image and retrieve its original content [30, 121, 188, 193]. Similarly, the Deblurring task consists in removing the blur effect that can be present in an image due to different reasons (for instance the motion of the camera, etc.)[30, 92, 187, 189]. In Super Resolution, the main goal is to increase the resolution of an image without inserting using any interpolation or inserting low-quality artifacts in the image[51, 82, 95, 147]. The image restoration task instead, aims to retrieve damaged images, like old scans, improving their quality[30, 166, 174, 187]. Examples of these tasks are reported in Figure 2.17.

3 Artifacts-Free White-Box methods for Image Enhancement

Image enhancement is a widely recognized image processing challenge that revolves around improving the visual quality of low-quality images to achieve higher fidelity without altering their content. In contemporary scenarios, Deep Convolutional Neural Networks have emerged as the go-to solutions for a multitude of image processing tasks, including image enhancement, super resolution, and image inpainting, among others. Consequently, these neural networks have become integral components of the toolkit employed by professional photographers for post-processing their images.

However, effectively using such tools demands a certain level of expertise, prompting the need for both semi-automatic and fully automatic procedures. Notably, within the domain of neural models, image-to-image translation methods [77, 80, 204] have exhibited significant efficacy in image enhancement. Yet, these methods grapple with two critical challenges: the emergence of artifacts in their output images and the limitations on input resolution often imposed by neural network architectures.

Recent advancements in image translation methods, particularly those based on diffusion models [9, 86, 141, 143], show promise in producing high-quality images devoid of artifacts. These methods involve multiple iterative steps, each time preserving the spatial resolution of the image. Nonetheless, due to the computational complexity of these steps, generating medium- or high-resolution images can be computationally intensive.

3.1 Introduction

Many recent methods function as enigmatic "black boxes," withholding any insight into the transformations performed by neural networks. In professional contexts, it's pivotal for a method to be "explainable," enabling users to verify and potentially tailor it to their requirements. Approaches to offer explanations for neural network predictions have been explored in recent years [117, 146]. These efforts propose methods for dissecting model predictions and uncovering the rationale behind network decisions.

This chapter presents three distinct methods to address these challenges. Section

3.3 introduces a fully explainable image enhancement approach. This method employs tree-search theory and deep reinforcement learning to deliver sequences of global enhancement operations. It effectively enhances the content of low-quality input images without introducing artifacts. Similarly, in Section 3.4, a white-box method for targeted enhancement is presented, allowing the independent enhancement of specific image areas. Lastly, Section 3.6 introduces an Explainable AI method. This approach utilizes a path planning algorithm to explain the enhancement process of black-box image-to-image translation methods using fundamental image processing operators.

3.2 Datasets and Metrics

In this section, the datasets and metrics, that are shared by the methods presented in the chapter, are reported.

3.2.1 Datasets

In this chapter, two widely recognized image enhancement datasets have been utilized. Although they were made available to the community to drive progress in the field of image enhancement, their intrinsic characteristics diverge significantly.

Adobe Five-K dataset

The first dataset considered is the Adobe Five-K dataset [23] for photographic image enhancement. This dataset is composed of 5000 images, available in the RAW format and in five different versions converted to the Adobe RGB space and retouched by expert photo editors (called A, B, C, D, and E). The entire dataset has been split into training (4000) and test (1000) sets by following the division proposed by Hu et al. [75]. As target images, the images processed by Expert C, considered the most consistent among the five experts, have been employed.

LOw-Light dataset

The LOL dataset [31] 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).

3.2.2 Metrics

Four different metrics have been employed: the Peak Signal-to-Noise Ratio (PSNR) [84], Learned Perceptual Image Patch Similarity (LPIPS) [195], Delta E (ΔE) [66] and Structural Similarity Index SSIM[74].

PSNR

The Peak Signal-to-Noise Ratio (PSNR) is a metric used to assess the quality of enhanced images. It measures the ratio of the peak signal power to the noise power in an image, providing a quantitative assessment of image fidelity. It is defined as:

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{\mathrm{MAX}^2}{\mathrm{MSE}}\right), \tag{3.1}$$

where MAX is the maximum possible pixel value and MSE is the Mean Squared Error.

SSIM

The Structural Similarity Index (SSIM) is a metric used to analyze the similarity between two images, considering both structural information, luminance and contrast. It is defined as:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{3.2}$$

where x and y are images, μ_x and μ_y are average pixel values, σ_x and σ_y are standard deviations, and σ_{xy} is the covariance.

Delta E (ΔE)

The Delta E (ΔE) metric is commonly used to quantify the perceptual difference between two colors or images. This metric computes the difference between the value of the pixels projected in the CIELAB color space. It is defined as:

$$\Delta E = \sqrt{(L_2 - L_1)^2 + (a_2 - a_1)^2 + (b_2 - b_1)^2} \tag{3.3}$$

LPIPS

The Learned Perceptual Image Patch Similarity (LPIPS) metric is used to assess the perceptual similarity between two images. It measures perceptual differences by comparing local image patches extracted from a convolutional neural network. It is defined as:

$$d(x, x_o) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} ||w^l \odot (\hat{y}^l_{hw} - \hat{y}^l_{0hw})||^2_2 \tag{3.4}$$

where x and x_0 are two patches respectively from the reference and input images, $\hat{y}^l$ and $\hat{y}^l_0$ are the normalized feature maps extracted from the layer l of convolutional neural network and w^l is a scaling vector.

3.3 A Tree Search Method For Global Image Enhancement

One of the main challenges of image-to-image translation algorithms for image enhancement is their black-box intrinsic nature. In fact these algorithms usually transform low-quality image in their high quality version without any insight on the transformation applied. Moreover due to the different nature of these methods, can happens that some artifacts are introduced in the final image degrading its quality.

On the basis of these considerations, in this section, TreEnhance, a global image enhancement method based on tree search and deep reinforcement learning, is proposed. The method is able to obtain high quality output images with the application of a sequence of global image editing operations. The chosen sequence derived from the method is entirely transparent, rendering it apt not only for automated enhancement but also as a versatile tool across various interactive scenarios. For instance, it could be employed for educational purposes, guiding beginners in learning how to edit their own images effectively.

The method operates without imposing any resolution constraints on input images. It seamlessly handles both high- and low-resolution images without requiring any adjustments. The effectiveness of the method has been evaluated on two distinct datasets, with parameter tuning focused solely on the image editing operations

This section's main contributions are:

- Pioneering the application of tree-search theory to address image enhancement problems, introducing a unique approach.
- Differently from methods form the state-of-the-art, TreEnhance serves as an entirely explainable neural architecture for image enhancement. It not only generates high-quality output images but also provides insight into the sequence of enhancing operators chosen to enhance low-quality images. These outputs excel both qualitatively and quantitatively, devoid of artifacts and unaffected by image resolution constraints.

- TreEnhance offers two distinct inference strategies for enhancing low-quality images, adaptable to data, time, and memory limitations.

- Featuring a guided search mechanism, it doubles as a “reverse engineering" tool, breaking down an expert photographer’s enhancement process into simple steps. This tool holds promise for educational software, helping beginners enhance their own photographs effectively.

Section 3.3.1 presents the operational process of TreEnhance, outlining its steps and neural architecture. Subsequently, Section 3.3.2 presents the results obtained in the experimental evaluation. Finally, Section 3.3.3 concludes the section with additional analyses exploring the potential of this novel method.

3.3.1 Method

TreEnhance is a tree-based deep reinforcement learning approach designed to enhance the quality of digital photographs. It achieves this by simulating the actions of a professional photo editor, who, from the range of available software operations, selects those to apply. This selection process mirrors a trial-and-error strategy, where multiple operations (like gamma correction, contrast adjustment, etc.) are repeatedly tested, assessed, revoked, and combined. Consequently, various sequences of operations are evaluated, leading to the identification of the most promising one.

More precisely, when provided an image **x**, the method identifies editing operations $a_1, \ldots, a_n$ from a finite set $\mathcal{A}$ of operations and applies them to yield the enhanced image $\hat{\mathbf{y}}$:

$$\hat{\mathbf{y}} = a_n(a_{n-1}(\ldots a_1(\mathbf{x}))). \tag{3.5}$$

The sequence is terminated by a special `STOP` operation.

To select the operations, the method constructs an expansive search tree that traverses through numerous potential sequences. A convolutional neural network guides the tree’s expansion, favoring the exploration of operations with higher potential to enhance the image. This network additionally assesses the images generated during the procedure, enabling the identification of the optimal outcome.

The network is trained through supervised learning, aiming to yield an enhanced image $\hat{\mathbf{y}}$ that closely approximates a target image **y** manually refined by a human expert. The following sections present the method in details.

Training Procedure

A set of input/target image pairs is employed to train the neural network. The training process involves alternating between two phases: generation and optimization. During the generation phase, an enhanced variant of the Monte Carlo Tree Search (MCTS) algorithm expands trees linked to randomly chosen image pairs from the dataset. Subsequently, in the optimization phase, the images generated during the generation phase are utilized to train the neural network's parameters

Generation Starting from the root node representing the input image, the method proceeds by choosing an image editing operation (an action) that produces a fresh image (a new node). These nodes and actions collectively shape a tree, developed through a sequence of iterations, each augmenting the tree with a new node. Each node is assigned a numerical score known as the *return*. The objective is to identify actions that are probable to yield nodes linked to elevated returns. Each iteration of the Monte Carlo Tree Search (MCTS) comprises four key stages [20]:

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 framework, the evaluation step entails repeatedly applying a random rollout policy until a terminal node is reached, obtaining the corresponding return.

In this context, similarly to the approach introduced by Silver et al. [151], nodes selected during the selection step undergo processing by a convolutional neural network. This network takes an image $\mathbf{x}$ as input and generates a policy $\pi(\mathbf{x}, \cdot)$ along with a value $v(\mathbf{x})$. The policy assesses the likelihood distribution of the optimal operation among all available editing operations ($\pi(\mathbf{x}, a)$ reflects the estimated probability that operation a is optimal for image $\mathbf{x}$). On the other hand, the value estimates the expected return $r(\cdot)$ achieved by the descendants of $\mathbf{x}$.

The tree policy employed in the selection phase is founded on maximizing the Upper Confidence Bound (UCB), which accounts for both the exploration of less-visited nodes and the exploitation of those nodes that have obtained higher average returns:

$$\mathrm{UCB}(\mathbf{x}) = \bar{r}(\mathbf{x}) + c \cdot \pi(\mathbf{x}_p, a) \frac{\sqrt{N(\mathbf{x}_p)}}{N(\mathbf{x}) + 1}, \tag{3.6}$$

where, $\bar{r}(\mathbf{x})$ represents the average return that has been propagated through $\mathbf{x}$ in previous iterations, $a \in \mathcal{A}$ signifies the operation that transformed $\mathbf{x}$ from its parent $\mathbf{x}_p$ (i.e., $\mathbf{x} = a(\mathbf{x}_p)$), and $N(\cdot)$ denotes the count of prior visits to a specific node in previous selection steps (thus, $N(\mathbf{x}_p)$ and $N(\mathbf{x})$ represent the visit count for the parent and child nodes, respectively). The coefficient c strikes a balance between the exploration and exploitation components.

The selection step starts from the root and progresses down the tree, favoring nodes with the highest bound (3.6) until a leaf node is reached. To enhance exploration, a slight measure of Dirichlet noise has been introduced into the policy. If a terminal node is encountered, it is directly evaluated rather than being processed through the neural network. This occurs in two situations: upon reaching a maximum depth and when the special `STOP` operation is chosen.

In the former scenario, the search terminates with a return of zero, indicating failure. In the latter case, the return hinges on the dissimilarity between the resultant image $\mathbf{x}$ and the target image $\mathbf{y}$:

$$r(\mathbf{x}) = \exp(-\alpha \|\mathbf{x} - \mathbf{y}\|^2), \tag{3.7}$$

where α is a tunable parameter.

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

$$\rho(\mathbf{x}, a) = \frac{N(a(\mathbf{x}))}{N(\mathbf{x})}. \tag{3.8}$$

This policy defines the probability distribution for visits to the root's children. A random sample is drawn from this distribution, determining the action applied to the root image. This approach ensures exploration by enabling the generation of multiple operation sequences for the same image. The process restarts with this new image, which becomes the new root of the tree, and repeats until a terminal node is reached. The upper part of Figure 3.1 illustrates the generation phase of TreEnhance.

Optimization The visited roots in the generation phase form the basis of the training dataset. Each training instance consists of a triplet comprising (i) the image $\mathbf{x}$, (ii) the return $r(\mathbf{x})$, and (iii) the probabilities of the stochastic policy $\rho(\mathbf{x}, \cdot)$. The neural architecture's parameters are refined through the minimization of the loss function L:

$$L = \lambda(r(\mathbf{x}) - v(\mathbf{x}))^2 - \sum_{a \in \mathcal{A}} \rho(\mathbf{x}, a) \log \pi(\mathbf{x}, a), \tag{3.9}$$

which combines the squared estimation error of the return with the cross-entropy

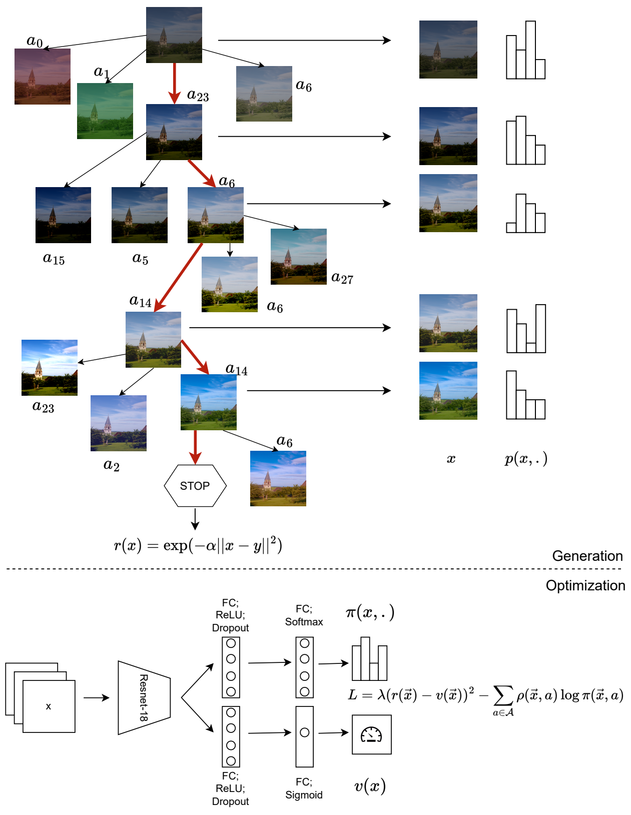


Figure 3.1: TreEnhance generation (top) and optimization (bottom) phases. To make the figure more clear only part of the tree is actually shown. For the complete architecture of the ResNet-18, please refer to the original paper. The function $\pi(\mathbf{x},\cdot)$ is the policy over the editing operations and $v(\mathbf{x})$ is the estimated return.

computing the discrepancy between the stochastic policy ρ and the network policy π. The parameter λ is a tunable hyperparameter that regulates the impact of the two terms in the loss.

The iterative procedure is carried out multiple times, with each iteration generating additional training data in the generation phase and refining the neural network further in the optimization phase.

Neural Architecture

The convolutional neural network utilized in this study is an adapted ResNet-18 model, where the final fully connected classification layer is substituted with two feed-forward heads. Both heads consist of two fully connected linear layers. The first head includes the softmax function to generate the vector representing the policy $\pi(\mathbf{x}, \cdot)$. The second head employs the sigmoid activation function, producing the scalar value $v(\mathbf{x})$.

The choice of ResNet-18 as the backbone architecture was driven by the concern of overfitting with deeper models due to the relatively small datasets used in the experiments. Nevertheless, there are no inherent restrictions on the neural network's architecture. The network is trained from scratch after undergoing random initialization.

The neural network can accommodate images of varying spatial resolutions, with memory availability in the system serving as the main constraint. Each node in the tree necessitates storing an image during training. Downsampling images can be employed to amplify the number of concurrently grown trees and enhance overall training speed (in the experiments, images were resampled to a resolution of 256×256 pixels).

The lower part of Figure 3.1 illustrates the architecture of the neural network. In the first layer of the two heads, dropout is implemented with a probability of p to mitigate overfitting. Here, p is a tunable hyperparameter, with its value set to 0.6 for these experiments.

Inference

After training, TreEnhance can be used through two distinct inference approaches: growing a tree (tree search) or directly using the trained neural network (policy-based search). The former strategy is characterized by greater accuracy, albeit at the expense of speed. In contrast, the latter approach is faster albeit marginally less precise.

Tree Search during inference, the method constructs a tree similar to the generation phase, but without access to the target image for computing terminal node returns. In cases where the `STOP` action is chosen or the maximum depth is reached, the return is approximated using $v(\mathbf{x})$. The algorithm proceeds as previously explained, propagating the value upwards to the root of the tree. Once the tree is constructed, a sequence of operations is derived by traversing the tree, selecting at each node the actions that maximize the policy $\rho(\mathbf{x}, \cdot)$.

Policy-based Search The second approach involves iteratively selecting the operation that maximizes the probability $\pi(\mathbf{x}, \cdot)$ calculated by the neural network. The process concludes either when the `STOP` action is chosen or when the maximum number of operations is reached.

3.3.2 Results

To assess the effectiveness of the solutions generated by TreEnhance, it was evaluated using the two datasets introduced in Section 3.2.1. Both datasets necessitate global editing operations for tone adjustment and contrast enhancement. In the case of the LOL dataset, spatial operators like deblurring and edge enhancement could also be beneficial.

LOw-Light Image Enhancement

The enhancement of the images in the LOL dataset requires mostly the application of global transformations, white balancing algorithms and spatial filters. The set of operations $\mathcal{A}$ selected for this experiment are:

- *Brightness adjustment*: a constant δ is added to the values of all the pixels:

  $$(a(\mathbf{x}))_{ijc} = \text{clip}(\mathbf{x}_{ijc} + \delta). \tag{3.10}$$

  The clip function limits the values of the adjusted pixels to the $[0,1]$ range ($\text{clip}(x) = \min\{\max\{x,0\},1\}$). Eight different variants of this editing operator have been considered: with $\delta \in \{-0.1, +0.1\}$, and applied to all the color channels or to a single color channel at a time $c \in \{R,G,B\}$.

- *Contrast adjustment*: the pixels values are stretched by a factor β around their mean:

  $$(a(\mathbf{x}))_{ijc} = \text{clip}(\mu_c + \beta \times (\mathbf{x}_{ijc} - \mu_c)), \tag{3.11}$$

  where μ_c is the mean channel value. The variants with $\beta \in \{0.8, 2.0\}$ have been considered and applied to all the channels, or just one.

- *Gamma Correction*: two different values have been selected, $\gamma \in \{0.6, 1.1\}$, and applied them to all or to one of the channels:

  $$(a(\mathbf{x}))_{ijc} = (\mathbf{x}_{ijc})^{\gamma}. \tag{3.12}$$

- *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 pixels are 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 employed on the image. This algorithm operates under the assumption that, on average, images appear gray, and the illuminant's value (per channel) can be approximated as a constant factor multiplied to the channel. Hence, to eliminate the illuminant's influence, the algorithm normalizes each channel of the image using the average pixel value.

- *Max RGB*: the max RGB white balancing algorithm is applied. This method is based on the principle that the brightest pixel in each channel should be treated as white. Consequently, each channel is divided by its highest value.

- *Median Filter*: replaces a pixel's value with the median of the values of its neighbours. The dimension k of the filter's window was tune to 3 obtaining a 3×3 filter.

- *Gaussian Blur Filter*: this operator blurs the input image with a Gaussian filter with standard deviation σ. The parameter σ was set to 2.

- *Sharpening Filter*: increases the sharpness of intensity transitions in an image. The sharpening filter considered is the unsharp masking, defined by the kernel F:

$$F = \begin{bmatrix} -0.125 & -0.125 & -0.125 \\ -0.125 & 2 & -0.125 \\ -0.125 & -0.125 & -0.125 \end{bmatrix}. \tag{3.13}$$

- *Edge Enhancement Filter*: highlights the edges in the image.

$$F = \begin{bmatrix} -0.5 & -0.5 & -0.5 \\ -0.5 & 5 & -0.5 \\ -0.5 & -0.5 & -0.5 \end{bmatrix}. \tag{3.14}$$

- *Detail Filter*: the primary purpose of this filter is to enhance the details of the subjects or objects present in the image.

$$F = \begin{bmatrix} 0 & -0.17 & 0 \\ -0.17 & 1.67 & -0.17 \\ 0 & -0.17 & 0 \end{bmatrix}. \tag{3.15}$$

- *Smoothing Filter*: thid filter replaces the value of the pixels with the average of their neighbors in the filter window.

$$F = \begin{bmatrix} 0.077 & 0.077 & 0.077 \\ 0.077 & 0.385 & 0.077 \\ 0.077 & 0.077 & 0.077 \end{bmatrix}. \tag{3.16}$$

- `STOP`: terminates the enhancement procedure.

A total of 38 editing operations are available, including eight for brightness adjustments, eight for contrast adjustments, eight for gamma corrections, two for saturation adjustments, two for hue rotations, two for white balancing algorithms, seven for spatial filters, and one stop operation. The parameters $\delta, \beta, \gamma, s, h, \sigma, k$ have been selected to enable image enhancement within a reasonable number of steps. In scenarios where TreEnhance is applied to resampled images to conserve memory, adjustments to the parameters of spatial operators might be necessary when finally applied to the original high-resolution images.

The parameters of the training procedure outlined in Section 3.3.1 were chosen based on preliminary experiments, while also considering the computational resource costs. During each generation phase, a total of 100 training images were selected, and for each of them, 10 000 MCTS iterations were executed to construct the corresponding tree (with a maximum tree depth of ten). The image resolution was set to 600×400 pixels, and standard data augmentation techniques such as cropping, flipping, and resizing were employed to enhance the diversity of the training set. From each tree, ten $(\mathbf{x}, r(\mathbf{x}), \rho(\mathbf{x}, \cdot))$ triplets were sampled and included in the training set. The parameter α in Equation (3.7) was set to 0.05, and the exploration parameter c in Equation (3.6) was assigned a value of 10.

During each optimization phase, the neural network's parameters were updated by minimizing the average loss (3.9) using 160 iterations of the ADAMW optimizer, with a learning rate of 10^{-3} and weight decay of 10^{-2}.

After the training procedure, the method was applied to the test set. For each test image, a tree with 1000 nodes was grown, and a final editing sequence was selected using the procedures outlined in Section 3.3.1. Table 3.1 presents the results of TreEnhance on the LOL test set using the Tree Search inference procedure.

The performance of TreEnhance was compared with several state-of-the-art architectures in terms of PSNR, SSIM, and ΔE. The definition of these metrics have been presented in Section 3.2.1.

TreEnhance demonstrated superior performance compared to all other considered methods in terms of PSNR and ΔE. Figure 3.2 shows the results of the application of TreEnhance to the LOL test set. Both qualitatively and quantitatively, the obtained results are highly satisfactory.

Table 3.1: Comparison of state-of-the-art enhancement methods on the LoL dataset.

Method ↓	PSNR↑	SSIM↑	ΔE ↓
RRDNet[202]	11.36	0.54	32.63
DSLR[101]	15.23	0.68	23.47
ExCNet[194]	15.80	0.67	21.47
Zero-DCE[69]	16.15	0.70	21.99
RetinexNet[31]	17.16	0.72	18.31
MBLLEN[118]	17.38	0.75	20.48
KinD[197]	18.27	0.83	15.02
LAE-Net [107]	18.30	0.64	-
KinD++[197]	18.75	0.82	15.13
EnlightenGAN[81]	18.82	0.78	15.06
DPED[77]	19.71	0.81	14.81
HDRNet[61]	20.14	0.83	14.26
RetinexDec[119]	20.23	**0.84**	13.10
TreEnhance	**21.96**	0.81	**10.82**

Photographic Image Enhancement

For the second dataset, the Adobe Five-K dataset, the set of enhancement operators $\mathcal{A}$ was reduced by excluding the spatial filters and white balancing algorithms. This choice was based on the understanding that experts solely utilized operations achievable through the "color curves" tool in photo editing applications. For the Five-K dataset, the difference between the input and ground truth images is smaller compared to that observed in the LOL dataset. For this reason the parameters of the operators has been changed obtaining more fine-grained actions.

- *Brightness adjustment*: $\delta \in \{-0.05, +0.05\}$;
- *Contrast adjustment*: $\beta \in \{0.894, 1.414\}$;
- *Gamma Correction*: $\gamma \in \{0.775, 1.05\}$;
- *Saturation adjustment*: $s \in \{0.707, 1.414\}$;
- *Hue rotation*; $h \in \{-9.0, +9.0\}$.

The total number of operations is 29, which includes the `STOP` action, two saturation adjustments, two hue rotations, and operations involving brightness, contrast,

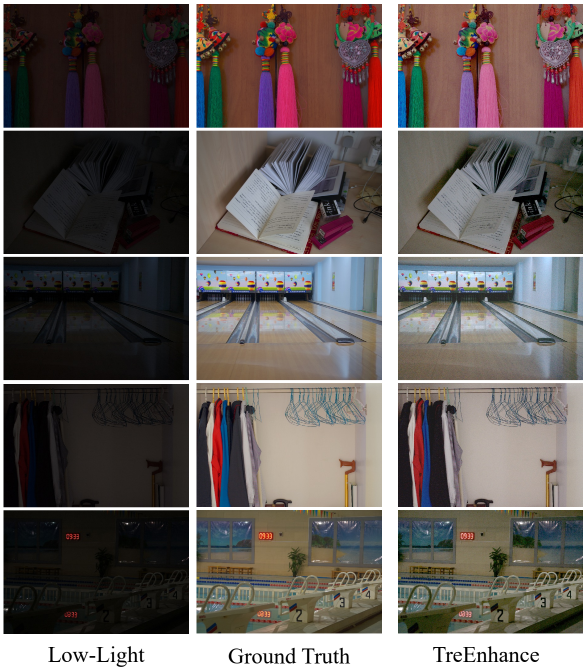


Figure 3.2: Results of the application of TreEnhance on the LOL test set. From left to right: low-light images, TreEnhance output and ground truth images.

and gamma correction applied either to a single channel or to all three channels simultaneously.

The training procedure employed for this second dataset followed the approach explained in Section 3.3.1. During each generation phase, 100 images were sampled from the training set, downsampled to 256×256 pixels, and subsequently subjected to data augmentation similar to the LOL dataset. The number of MCTS iterations performed was 10 000, maintaining a maximum depth of ten. The parameter c in Equation (3.6) was set to 4, a smaller value compared to that used for the LOL dataset due to the reduced number of operations, necessitating less exploration. The parameter α in Equation (3.6) has been set to the value of 0.05, the same as that used for the LOL dataset. After each generation phase, the network underwent training with mini-batch gradient descent using the ADAMW optimizer for 70 iterations.

To evaluate the performance of the proposed method on the Five-K dataset (with the Tree Search inference procedure), a comparison was conducted with several other

Table 3.2: Comparison of state-of-the-art enhancement methods on the Adobe five-k dataset.

Method	LPIPS↓	PSNR↑	ΔE ↓	SSIM↑
Exposure[75]	0.16	18.74	13.57	0.81
CycleGan[204]	0.16	19.38	12.95	0.78
DCE-Net[199]	0.13	20.47	11.77	0.85
DCE-Net-Pooling[199]	0.15	20.33	12.13	0.85
DaR[130]	0.10	20.91	13.05	0.86
Unfiltering[12]	0.09	21.67	10.22	0.88
Pix2Pix[80]	0.09	23.05	8.84	0.86
HDRNet[61]	0.08	22.31	9.53	0.89
Star-DCE[199]	0.08	**23.55**	**8.45**	0.89
Parametric[13]	0.07	23.20	8.52	**0.90**
TreEnhance	**0.06**	21.24	11.25	0.89

methods from the state of the art. The results are summarized in Table 3.2.

Given that the images provided in the Five-K dataset are in RAW format, a preprocessing step was necessary before training the algorithm. To ensure a fair comparison with other state-of-the-art methods, these algorithms were applied to the preprocessed data using the code provided by the respective authors.

From the comparison, it is evident that TreEnhance competes effectively on this dataset, demonstrating strong quantitative performance. Notably, it achieved the best results in terms of LPIPS [196], a metric that measures the perceptual similarity between images. Additionally, it obtained a high SSIM value. However, concerning pixel-level metrics such as PSNR and ΔE, other methods obtained higher results. This difference can be attributed to the granularity of the set of editing operations considered. To enhance pixel-level accuracy, a more fine-grained array of operations would be necessary, potentially resulting in longer sequences found by the algorithm. It's important to note that overly lengthy sequences can diminish the model's explainability.

Figure 3.3 presents a visual comparison of images obtained using TreEnhance and other state-of-the-art methods to enhance a sample from the Five-K test set.

TreEnhance demonstrates impressive performance on the two datasets under consideration. Its results remain competitive across all the metrics analyzed, as presented in Tables 3.1 and 3.2. However, it was not the best method on the Five-K dataset concerning MSE and PSNR. A plausible explanation for this discrepancy could be

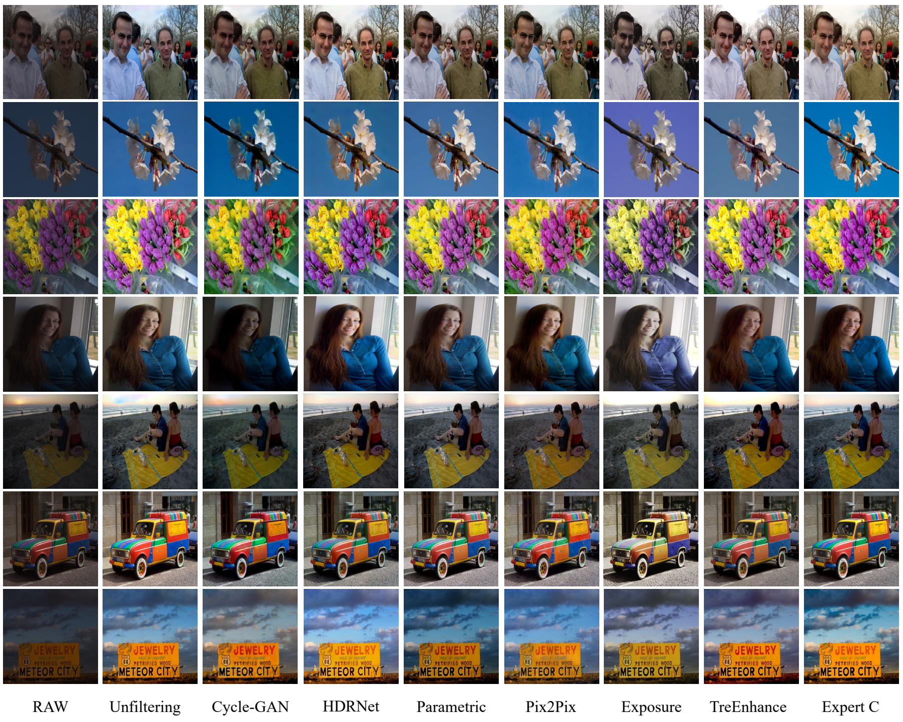


Figure 3.3: Examples of test images in the Five-K dataset processed by methods in the state of the art. The left most column shows the input image and the right most the ground truth.

attributed to the fact that some images in the Five-K dataset have undergone retouching using spatially varying operators, such as masked operations. Replicating this specific condition with solely global operators remains challenging. In Section 3.4 a different solution involving spatial operators for enhancing low-quality images is presented.

To better understand the differences among the images generated by the compared methods, a subjective evaluation test was conducted. Ten participants, all possessing some experience in the field, were selected to choose the most aesthetically pleasing version among seven versions of the same image. This assessment procedure was repeated for 100 different images from the Five-K test set. The methods under comparison included TreEnhance, Pix2Pix, CycleGAN, Star-DCE, DaR, Exposure, and HDRNet. Participants utilized a system that randomly displayed the seven alternatives

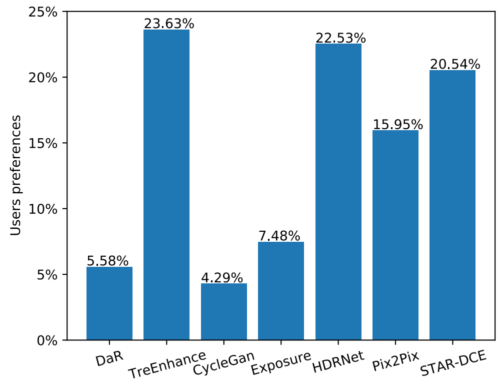


Figure 3.4: Results of the user study on the 100 images from the Five-K test set.

without any indication of the associated methods. The sequence in which the 100 images were presented varied for each participant.

The results shown in Figure 3.4 reveal that TreEnhance was the preferred choice for **23.63%** of the participants, closely followed by HDRNet (22.53%) and Star-DCE (20.54%). The remaining methods obtained considerably lower preferences: Pix2Pix 15.95%, Exposure 7.48%, DaR 5.58%, and CycleGAN 4.29%.

These results confirm TreEnhance's ability to generate high-quality images from a perceptual point of view, even if its pixel-level metrics are not the most favorable. This is confirmed by its higher performance in terms of the perceptual LPIPS metric. An informal post-test interview revealed that most users clearly identified three methods as superior to the rest. These methods consistently exhibited desirable tonal qualities and produced outputs free from any artifacts, a significant advantage over the others. Users perceived these top three methods as largely equivalent.

Based on the conducted user study, TreEnhance emerged as a strong competitor among various image enhancement methods. Furthermore, it demonstrated an ability to generate images comparable to modern state-of-the-art techniques.

In addition to these experiments, exploratory tests with probabilistic denoising diffusion models [141] were conducted. Preliminary results indicated that these models excel at preserving the semantic content of images without introducing artifacts. However, diffusion models struggled to replicate the intent of a photo editor, resulting in lower accuracy in terms of performance metrics, especially PSNR and ΔE. One plausible explanation for these outcomes is the comparatively smaller scale of the datasets used in our experiments compared to those employed in the reference paper.

3.3.3 Additional Experiments

The results achieved by TreEnhance in the two distinct tasks are satisfactory, both quantitatively and qualitatively. The method produces high-quality output images by applying fundamental image editing operations, which in turn facilitates a straightforward explanation of the process. During the inference phase, TreEnhance assesses multiple potential sequences of operators, but selecting and applying only the most promising one to the input image.

In this section the memory and time requirements of TreEnhance are presented, and analyzed more in detail. Moreover an ablation study on the network architecture is reported. Finally, the images from the five-k test set produced with different methods, are analyzed, focusing on the presence of artifacts.

Memory and Time Requirements

The generation phase of the training procedure is the most memory-intensive step of TreEnhance, primarily due to the number of MCTS steps, which also corresponds to the number of nodes in the trees. Unlike some other state-of-the-art methods, TreEnhance does not impose any limitations on the resolution of input images. However, to conserve memory (and consequently enable the growth of larger trees), working with resampled images is possible.

After finding the sequence of editing operations 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 need to be adjusted accordingly. Examples of TreEnhance applied to high-resolution images can be observed in Figure 3.5.

The conducted analysis reveals the importance of ensuring a substantial level of exploration during the initial stages of the training procedure. In the early stages, the process shows a breadth-first search behavior, where multiple nodes at the same level are explored until a promising action is discovered. Afterward, the method goes deeper into the promising branches of the tree. This shift occurs as the convolutional neural network (CNN) becomes more adapt at assigning accurate values, enabling the algorithm to select promising actions with higher frequency.

As explained in Section 3.3.1, only a randomized subset of the training set is utilized in each generation. This subset is divided into batches (four in the experiments) to make efficient use of available memory. To speed up the generation phase, trees corresponding to each image in the batch grow concurrently. In the experiments, an Intel® Core™ i7-8700 CPU @ 3.20GHz with 48 GB of memory was employed. For the evaluation step of the generation phase and the optimization phase, an NVIDIA RTX™ 3080Ti graphics card with 12 GB of memory was used.

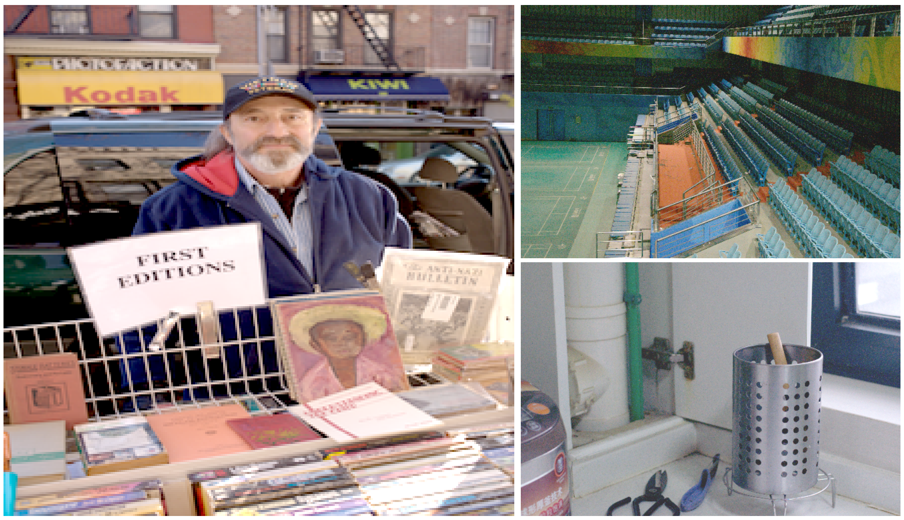


Figure 3.5: Examples of the application of TreEnhance to high resolution test images from the Five-k and LOL datasets.

Network Ablation

In Section 3.3.1, the neural architecture used in this work has been presented. In Table 3.3 different neural network architectures are compared in terms of PSNR on the LOL dataset.

Moreover, a comparison is conducted across different network architectures, including relatively compact networks like ResNet18 and DenseNet121, medium-sized networks like ResNet34 and ResNet50, and the larger ResNet101. As previously mentioned, TreEnhance does not impose any constraints on the choice of the neural architecture used for estimating the probability distribution over actions and the outcome of the enhancement process. Both ResNet18 and DenseNet121 have a comparable number of parameters and FLOPs (computed for images of resolution 256 × 256). These networks exhibit similar performance in terms of PSNR. However, due to the relatively high number of forward passes required by the tree-based strategy and the slightly higher PSNR achieved, ResNet18 is selected as the reference architecture for the method. This decision is made despite its slightly greater parameter count compared to DenseNet.

Interestingly, the performance of larger networks is inferior to that of the smaller ResNet. This phenomenon can be attributed to overfitting, which occurs due to the

Table 3.3: Comparison of different neural architectures on the LOL test set with the Tree-Based inference strategy.

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	20.13

Table 3.4: Comparison of TreEnhance inference strategies on the two dataset considered.

	Five-K			LOL		
Search strategy	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

relatively limited size of the training set. This pattern of overfitting is expected to persist in even larger networks, such as ResNet152, given the same dataset limitations.

Tree-based and Policy-based Enhancement

As detailed in Section 3.3.1, TreEnhance offers two distinct inference strategies during the inference phase: constructing a tree for each input image or utilizing the neural network as a policy directly. Table 3.4 provides a comparison of these two different strategies across the two test datasets. The Tree Search enhancement strategy emerges as the more accurate approach due to its consideration of a broad spectrum of alternatives. In contrast, the Policy-based Search strategy adopts a *greedy* approach, consistently selecting the most promising operation. Consequently, this strategy is faster than the tree-based approach, requires less memory, but exhibits a slight reduction in accuracy. The additional complexity associated with the Tree Search strategy can be controlled by restricting the number of MCTS steps in the inference phase. As depicted in Figure 3.6 (left), the accuracy of the method improves with a higher number of steps. Another parameter tunable during the search is the coefficient c in Equation (3.6). Figure 3.6 (right) indicates that a larger value of c

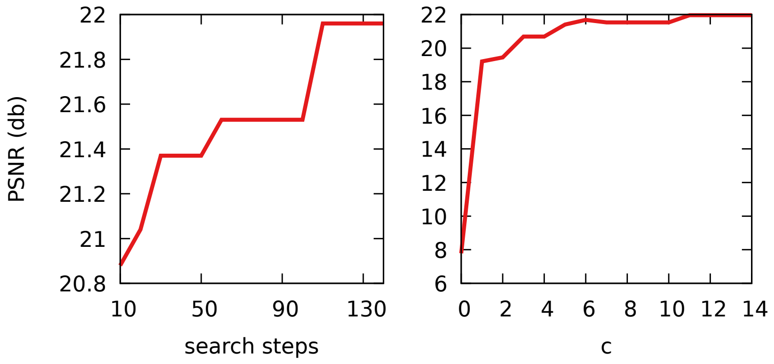


Figure 3.6: Accuracy obtained by the TreEnhance on the LOL test set varying the number of MCTS search steps (left) and the exploration/exploitation coefficient (right).

appears advantageous during the inference phase.

The two inference strategies are examined in terms of time, number of parameters, and FLOPs. Both solutions are built upon the modified version of the ResNet18 architecture previously presented. The architecture contains a total of 11.32M parameters. As detailed in Section 3.3.1, the primary distinction between the two inference phases lies in the fact that the first approach involves tree growth during inference, while the second directly employs the learned policy. The computational load for a forward pass on a 256 × 256 image is 2.38 GFLOPs. For the tree strategy, the overall computational cost depends on the frequency of network evaluations on images linked to nodes, which, in turn, is influenced by the maximum number of steps and the maximum tree depth. In the policy-based strategy, the neural network evaluates the image until the maximum number of editing operators has been applied or the `STOP` action is selected. Table 3.5 compares the computational costs and times of the two search strategies on the Five-K test set with those of other methods from the literature. The average FLOPs per image and the average time per image are provided. The time measurements were taken on a computer equipped 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.

Guided Search

As an additional experiment, TreEnhance was tested as a tool for "reverse engineering" the work of a professional photo editor. For this purpose, a modified version of

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	56.88	1642.95	19.38
Pix2Pix	54.41	18.15	1693.44	23.05
STAR-DCE	0.03	0.02	25.17	23.55
DCE-Net	0.08	5.22	44.72	20.47
HDRNet	0.48	0.17	267.54	22.31
Unfiltering	148.02	2.59	816.56	21.67
Parametric	0.03	0.05	756.93	23.20

Table 3.5: 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.

TreEnhance was adapted to use ground truth information during the inference phase (Equation 3.7). This adjustment allows the MCTS search to retrieve the sequence of operations necessary to transform the input image into a retouched version that matches the ground truth. This approach, referred to as "Guided Search", serves multiple purposes. It not only offers insights into the enhancement process but also provides an upper bound for TreEnhance's performance. Moreover, it could be integrated into a photo editing application to assist both beginners in imitating expert users and professionals in refining and optimizing their enhancement workflows.

The results of this experiment are presented in Table 3.4. The achieved accuracy in this scenario is notably high, reaffirming the effectiveness of MCTS in identifying effective sequences of image editing operations. This experiment suggests that the "Guided Search" strategy could be integrated into educational software, serving as a problem-solving exercise to teach beginners how to enhance low-quality images and guide them in selecting appropriate operators to achieve specific results.

Analysis of the Sequences

The sequences generated by the three inference strategies exhibit correlation. As illustrated in Figure 3.7, all three strategies favor the use of global enhancing operators. Specifically, the most frequently selected operations involve increasing brightness, gamma, and contrast. This tendency is due to the fact that the original LOL test images are characterized by low light conditions, often requiring brightness adjustments. Operations affecting individual channels are more likely to be chosen when the current image is close to the target image.

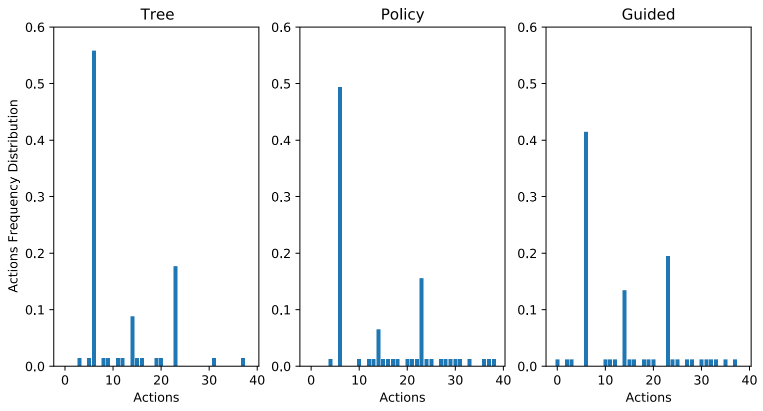


Figure 3.7: Frequency of the operations selected by the three inference strategies applied to the LOL test set. The action numbers reported here follows the order of presentation of subsection 3.3.2

Comparatively, the tree-based and policy-based strategies tend to over-prioritize gamma correction actions across all image channels, in contrast to the guided search strategy (action 6). Guided search displays a more evenly distributed distribution of operations across the three channels, in contrast to the other strategies. This balanced distribution contributes to higher-quality images, thereby enhancing the results.

Artifacts

One of the prominent characteristics of generative models for image enhancement, such as those based on GANs, is their capacity to accurately model color distributions. However, a drawback associated with these models is their tendency to introduce visible artifacts into their outputs. Figure 3.8 presents a comparison between two generative methods (Pix2Pix and CycleGAN) and two reinforcement learning methods (TreEnhance and Exposure) using two Five-K test images.

In both cases, the generative methods exhibit impressive color distribution modeling capabilities; however, their outputs often come with noticeable noise and artifacts. Consequently, perceptual metrics like LPIPS tend to yield relatively unfavorable scores for these types of methods. On the other hand, reinforcement learning methods do not introduce artifacts into the output images, but this advantage might sometimes result in slightly less vibrant colors.

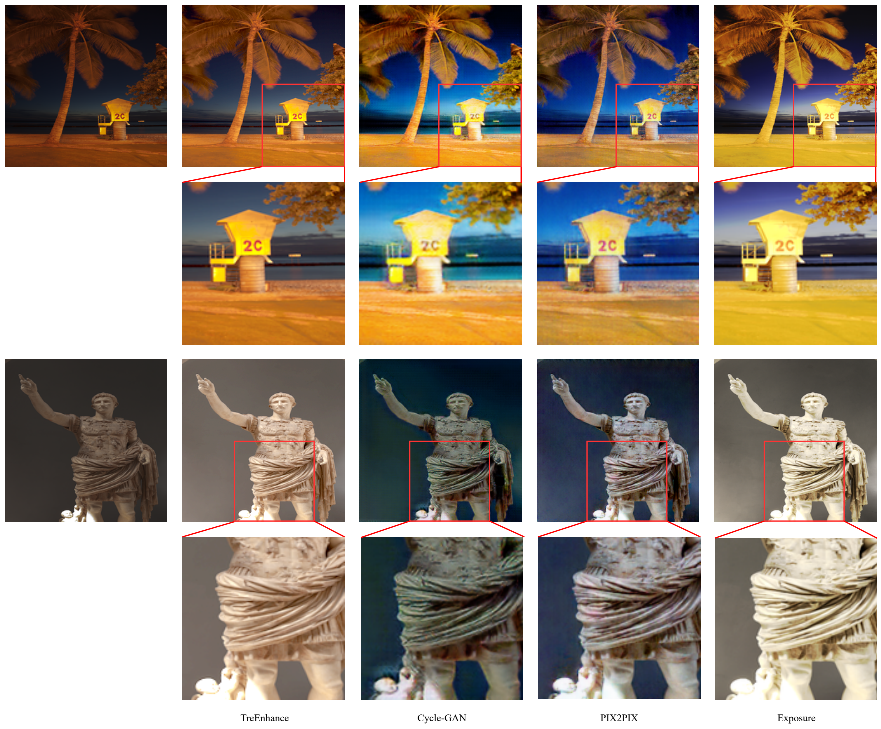


Figure 3.8: Comparison of generative and reinforcement learning methods on two Five-K test images.

3.4 Masked-Based Image Enhancement Through Tree-Search Theory and Deep Reinforcement Learning

Enhancing raw images, such as those directly captured by a DSLR camera sensor, and transforming them into stunning photographs is a formidable task. Professional photo editors dedicate years to mastering and refining this skill. In recent years, the evolution of hardware has facilitated the widespread adoption of deep learning models across various domains, including computational photography. Consequently, photographers now have the opportunity to explore and utilize these solutions embedded within photo editing applications. However, effectively using these software tools remains a complex challenge that demands a profound understanding of the underlying algorithms. This underscores the necessity for deep learning solutions that not only deliver accuracy but are also intuitive and interpretable.

In this section, we present an improved variant of the algorithm introduced in Section 3.3. Unlike TreEnhance, which primarily employed global enhancing operators applied uniformly to the entire image without focusing on specific regions, the method discussed here utilizes a small set of spatial operators. These operators iteratively select areas within the image for enhancement, accompanied by a single enhancing operator based on histogram equalization. Consequently, the algorithm is capable of independently enhancing not just the entire image but also high-frequency details located in smaller regions. This form of enhancement is referred to as targeted enhancement.

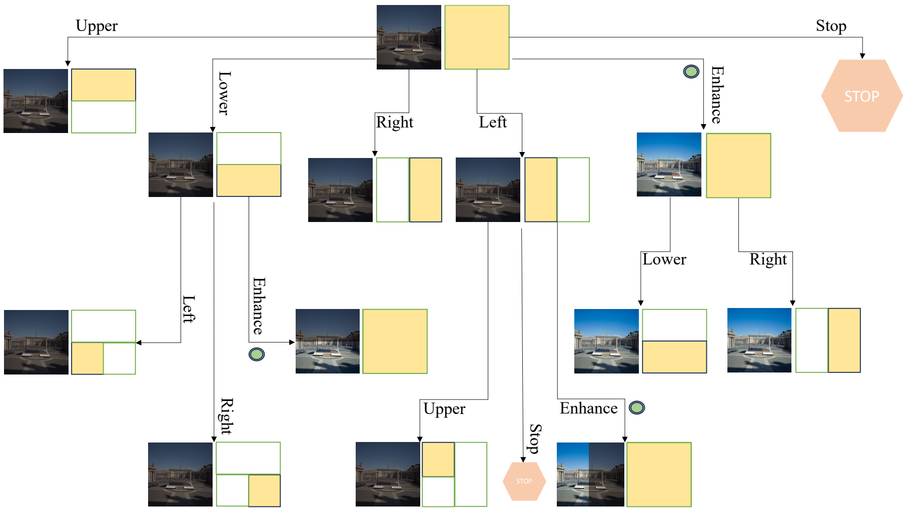


Figure 3.9: Example of the tree traversed by the proposed method. For the sake of clarity, not all the nodes are completely expanded. The root of the tree is the input image and the mask is initialized as true. In the second level of the tree, the six possible operators connect each node to the root (i.e. a selection operator, the enhancement operator, or the stop operator).

In light of these considerations, this section introduces a novel and entirely explainable method for image enhancement. This method integrates principles from tree-search theory and deep reinforcement learning. To be more specific, it systematically identifies regions within the image that require enhancement and subsequently applies a fundamental operator to those areas. This approach not only yields sequences of actions that enhance the quality of input images but also furnishes explanations for its selections. The results obtained applying the method on two datasets demonstrate that this new approach is competitive with image-to-image and parametric black-box methods, despite its simplicity.

The main contributions of this section can be summarized as follows:

- A novel and fully explainable method for image enhancement.
- An extensive experimentation that demonstrates that the method is competitive with state-of-the-art methods on two datasets of images.
- A formalization of the concept of targeted enhancement, where specific parts of the image are processed independently on the others.

Section 3.4.1 provides an in-depth overview of the method. Section 3.4.2 presents the results obtained on two image datasets and conducts a qualitative and quantitative comparison with those obtained by other state-of-the-art methods. Finally, Section 3.4.3 delves into additional experiments conducted to analyze the effectiveness of the method.

3.4.1 Method

Given an input image x and a target image y, the proposed method seeks a sequence of operators that can be applied to x to approximate $\hat{y} \simeq y$. The space of operators is explored to create promising sequences, which are stored in a tree. Each node in the tree contains an image x and a binary mask m that represents a selection of pixels from x. The root of the tree corresponds to the input image, paired with a mask that selects all pixels. Figure 3.9 provides a visual representation of this tree structure. Following the strategy introduced by Cotogni et al. [41], edges between nodes symbolize the application of an operator $a \in \mathcal{A}$, where $\mathcal{A}$ represents the set of possible operators. In this extended version, the method, which initially supported only global operators, has been enhanced to work locally. Specifically, most operators affect only the selection m, with only one operator modifying the pixel values. These operators can be categorized into three distinct groups:

- Spatial Selection Operator: Given the mask $m \in \mathbb{R}^{1\times H\times W}$ and a direction $d \in$ Left, Right, Up, Down, the selection is limited to 50% of the pixels already selected by the mask in the specified direction. For example, if $d =$ Up, the operator selects all the pixels above the barycenter of the current mask m.
- Enhancement operator: This operator applies Contrast Limited Adaptive Histogram Equalization (CLAHE) [133] to the region of the image selected by the mask. Given the mask $m \in \mathbb{R}^{1\times H\times W}$ and the image $x \in \mathbb{R}^{3\times H\times W}$, the new image

is obtained in two steps: *(i)* CLAHE is applied to the pixels selected by the mask m: $x' = \text{CLAHE}(x \circledast m)$, and *(ii)* $x = \alpha x + (1 - \alpha)x'$, where the parameter α is introduced to smooth the transition between the area enhanced by CLAHE and the adjacent pixels.

- Stop Operator: This operator terminates the enhancement sequence. The node reached when using this operator becomes a terminal node and is not expanded further.

The tree is explored using a modified version of the Monte Carlo Tree Search (MCTS) algorithm [151]. In this modified version, a neural network f_θ serves as a function evaluator. This neural network takes as input a tensor $s \in \mathbb{R}^{4\times H\times W}$, which is the concatenation of the mask m and the RGB image x along the channel dimension. The neural network provides two outputs, $\pi, v = f_\theta(s)$. The first output, $\pi(s,\cdot)$, represents the policy, which is a probability distribution over the operators and is used to predict the best operator for the image. The second output, $v(s)$, estimates the expected return r that will be obtained from the current node in the tree.

Starting from the root node (i.e., the input image with all pixels selected), a fixed number n of MCTS iterations are performed to expand the tree. Each iteration involves descending the tree until a new node is discovered and expanded, or until the stop operator is selected, or the maximum depth d is reached. To ensure sufficient exploration, the Upper Confidence Bound (UCB) [156] is utilized to select the operations to descend the tree. After the n iterations, MCTS provides as output the probability distribution of the operators that can be selected from the root node, denoted as $\rho(s,\cdot)$. From this distribution, an operator is sampled as $a \sim \rho(s,\cdot)$ and applied to the image or the mask. The result becomes the new root of the tree, and the procedure is repeated until a terminal node is reached. When this occurs, the return is computed. If the maximum depth d is reached, the computed return is zero. Otherwise, if the Stop operator is selected, the return is computed as:

$$r(s) = e^{-\phi||x-y||^2}, \tag{3.17}$$

where y is the target image, and ϕ is a parameter to be set.

The roots considered by the algorithm, the probability distributions computed during MCTS denoted as $\rho(s,\cdot)$, and the return $r(s)$ are utilized to update the neural network's parameters. This is achieved through the minimization of the following loss function:

$$L = \lambda(v(s) - r(s))^2 - \sum_{a\in A} \rho(s,a)\log\pi(s,a). \tag{3.18}$$

The alternation of the two phases, of self-play via MCTS and neural network optimization, ensure a balanced approach between exploiting the knowledge embedded in the network and exploring new strategies.

During inference, when the tree is explored using MCTS and the Stop operator is chosen, the return is computed as $r(s) = v(s)$ to handle the absence of a target image.

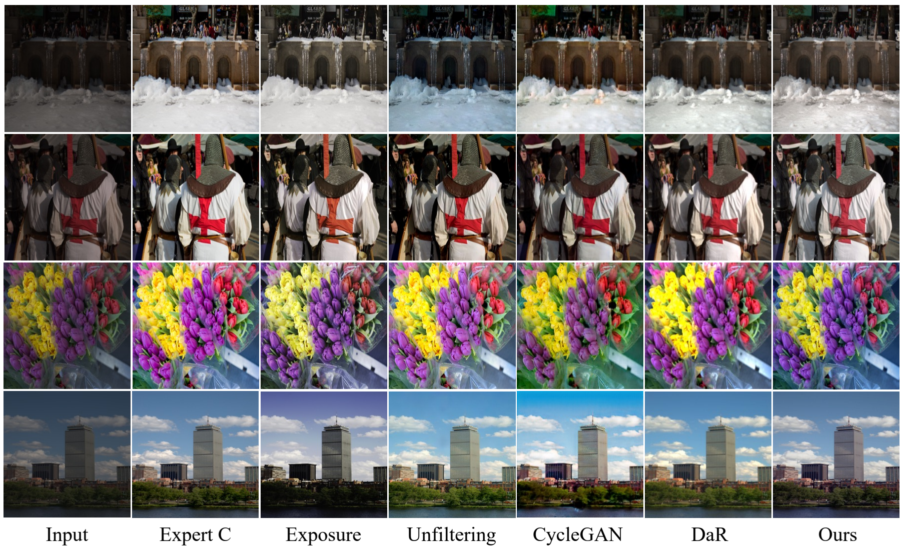


Figure 3.10: Qualitative comparison on Five-K test images.

3.4.2 Results

The proposed method has been evaluated in two distinct tasks: photographic image enhancement on the Five-K dataset and low-light image enhancement on the LOw-Light (LOL) dataset (details on the datasets have been presented in 3.2.1). These tasks differ in the nature of the images to be enhanced. In the former, the input images are directly captured by a DSLR camera sensor, and the enhancement includes both visual improvement and translation into the standard RGB color space. In contrast, the latter task involves input images taken under 'low-light" conditions, aiming to make them appear as if they were captured in well-lit environments.

As detailed in Section 3.4.1, during inference, the tree traversal proceeds without utilizing the ground truth image $\hat{y}$. The obtained sequences, denoted as $[a_0, a_1, \ldots, a_n]$, are then fine-tuned by introducing a gamma correction as the final operation. This step is necessary to restore the standard distribution of lightness (note that histogram

Table 3.6: Comparison of image enhancement methods on the Adobe Five-K dataset.

Method	LPIPS↓	PSNR↑	ΔE ↓	SSIM↑
Unfiltering [12]	0.09	**21.67**	**10.22**	0.88
CycleGan [204]	0.16	19.38	12.95	0.78
DCE-Net [199]	0.13	20.47	11.77	0.85
DCE-Net-Pooling [199]	0.15	20.33	12.13	0.85
Exposure [75]	0.16	18.74	13.57	0.81
DaR [130]	0.10	20.91	13.05	0.86
Proposed	**0.06**	21.56	10.77	0.87

equalization alone cannot achieve this).

Regarding the optimization phase, the neural network weights are updated to minimize the loss function described in Equation (3.18). The optimization algorithm employed is ADAM [113], with a learning rate of 10^{-4}. The number of epochs is set to five. As for the other parameters, the best performance is achieved with the following values: $\alpha = 0.05$, $\phi = 0.05$, $\lambda = 20$, and $n = 250$.

Photographic Image Enhancement

Table 3.6 presents the results achieved by the proposed method in comparison to those of similar methods from the state of the art. Specifically, the primary competitors of the proposed method are algorithms that leverage deep reinforcement learning (Exposure and DaR). As evident from the table, the method surpasses these counterparts in terms of PSNR, ΔE, and secures the overall best performance in LPIPS. Additionally, the proposed method has been also compared against image-to-image translation and parametric techniques such as Unfiltering, Cycle-Gan, and two variants of DCE-Net. Remarkably, despite employing a compact neural network and a single enhancement operator (CLAHE), the proposed method manages to achieve performance levels comparable to, or even superior to, those of competing methods that typically employ extensive neural architectures. Visual comparisons of the results are provided in Figure 3.10.

Low-Light Image Enhancement

Table 3.7 reports the results achieved by the proposed algorithm in comparison to other methods designed for low-light enhancement. The newly introduced solution demonstrates satisfactory results across the assessed metrics. Notably, despite achieving a relatively lower PSNR value (the second-best), the method secures the best result in terms of ΔE and exhibits a competitive SSIM value (the third-best). One weakness of the setup used for the method is that the color distribution in the output is limited to that achievable by multiple applications of CLAHE. Additionally, the application of the operator to small selected areas of the low-light image does not beneficiate of reduced spatial information.

Figure 3.11 reports the results obtained on the LOL dataset. While the method may not achieve the highest accuracy in terms of PSNR, the enhanced images exhibit a visually appealing quality. Notably, the method accurately reproduces the color distribution. However, it faces challenges when dealing with shadows. In contrast to the ground truth, which completely eliminates shadows, the method retains them, resulting in a more natural appearance. The overall quality of the results is further substantiated by the consistently low ΔE value.

Table 3.7: Comparison of state-of-the-art methods on the LoL dataset.

Method ↓	PSNR↑	SSIM↑	ΔE ↓
RRDNet [202]	11.36	0.54	32.63
DSLR [101]	15.23	0.68	23.47
ExCNet [194]	15.80	0.67	21.47
Zero-DCE [69]	16.15	0.70	21.99
RetinexNet [31]	17.16	0.72	18.31
MBLLEN [118]	17.38	0.75	20.48
DeepUPE [168]	17.87	0.73	20.56
RUAS [106]	18.22	0.72	-
LAE-Net [107]	18.30	0.64	-
KinD++ [197]	18.75	0.82	15.13
RetinexDec [119]	**20.23**	**0.84**	13.10
Proposed	19.50	0.81	**12.80**

3.4.3 Additional Experiments

In this section additional studies that has bee performed on the Five-K dataset to better understand the behavior of the proposed method are reported.

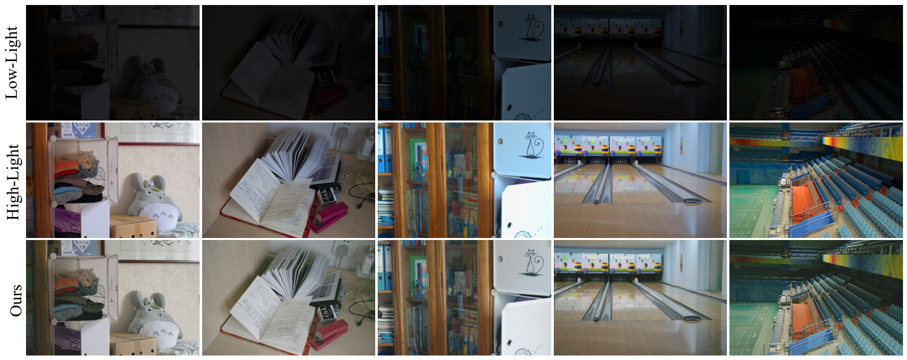


Figure 3.11: Qualitative comparison on LOL test images

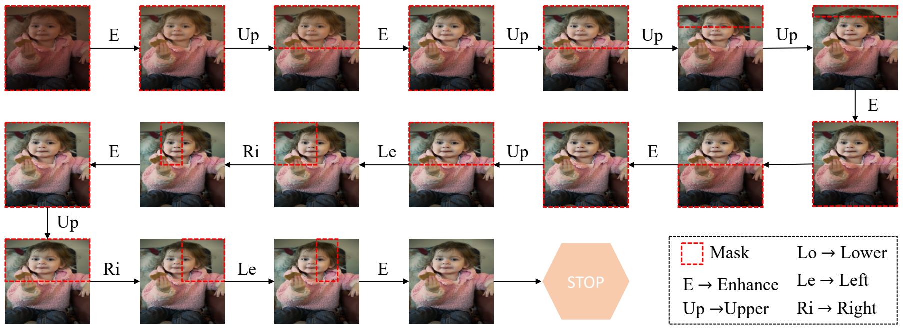


Figure 3.12: Example of a sequence found by the proposed method.

Analysis of the sequences.

One of the key features of the presented method is the ease with which the sequences of operations it generates can be inspected and comprehended. In Figure 3.12, the step-by-step procedure generated for a specific test image is presented. It's worth noting how immediately understandable the method's strategy is, and how effortlessly the regions that have been enhanced can be identified. The process is quite clear: first, CLAHE is applied to the entire image, then the method shifts its focus to refine the upper part of the image (initially the upper half, and subsequently the top border). Afterward, the method enhances the subject's face before concluding the process.

It is important to emphasize that this is just one example, and entirely different sequences could be generated for different input images. However, it illustrates a typical behavior of the method, which has learned to implement a coarse-to-fine

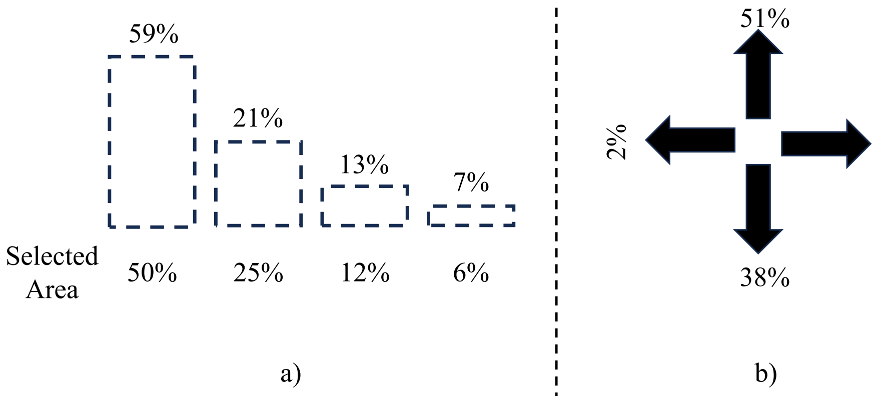


Figure 3.13: (a) Frequency distribution of the areas selected by the algorithm on the Five-K dataset. Given that the mask is initialized as true (100%), the 100% bin has been omitted from the plot to enhance clarity. (b) Frequency distribution of the directions selected by the proposed algorithm on the Five-K dataset. The term “direction” denotes which portion of the area, already marked as true in the mask, needs to be reduced.

enhancement strategy.

For additional analysis, the distribution of areas selected by the proposed algorithm is presented in the left segment of Figure 3.13. As mentioned earlier, since the masks are initially set to True, including the 100% bin in the plot does not enhance its clarity. Notably, the frequency distribution highlights a higher occurrence of larger areas being selected (e.g. 59% for a half-True mask), whereas smaller areas of the mask being set to true are less common. This trend arises from the nature of images, where details tend to occupy fewer pixels and consequently appear less frequently in the image.

In the right part of Figure 3.13, the distribution of directional choices made by the algorithm within the previously selected mask areas is reported. This distribution provides valuable insights into the algorithm’s refined decision-making process. The term “direction” refers to the specific segment within the selected area where the algorithm focuses its enhancement. The distribution shows patterns of pixel reduction, showcasing how the algorithm strategically prioritizes certain regions over others. For example, the most frequently selected direction is “Up.” This tendency can be attributed to the prevalence of open-air scenes in the images from the Five-K dataset, where the sky is prominently visible. Consequently, the algorithm often prioritizes enhancing the sky, which typically represents the brightest area within the image.

Multi-User scenario.

To observe how the proposed method adapts to different enhancement styles, an experiment involving multiple users was conducted. Specifically, the performance of the method was evaluated after training it five times, each time using images retouched by one of the five experts in the Five-K dataset as targets. The five versions obtained were then evaluated using images from another (or the same) expert. The results in terms of PSNR are presented in Table 3.8.

The main diagonal of the table includes the results of the "self-user" experiments, where the same user was used for both training and testing. Fluctuations can be observed among the five experts. This can be attributed to the fact that image enhancement is not solely based on objective content; it also involves a crucial subjective component, which is the visual pleasure and preferences of the photo retoucher. As mentioned earlier, Expert C serves as the reference for the model since their style is considered the most consistent among the five. For instance, the lowest PSNR values occurred in correspondence with Expert D. This could be because Expert D applied a wider variety of styles depending on the subject of the image, compared to the other artists who exhibited a more distinct and recognizable style across the different enhanced images.

In the remaining cells of the table, the results of the "cross-user" experiments are reported for completeness. An important observation from these results is that, for certain experts, the "self-user" PSNR value is not necessarily the highest in the row. This indicates that the proposed method is not only capable of devising a general enhancement strategy based on an expert's style but also capable of applying this strategy to emulate the enhancement of another expert. This use of cross-user strategies allows for further refinement and improvement of the "self-user" sequences.

In other words, the method's ability to adapt its enhancement approach to emulate different styles demonstrates its flexibility and capacity to learn from multiple sources. This versatility enables it to achieve better results by incorporating diverse enhancement strategies, refining its own sequences, and ultimately producing high-quality images that can rival or even surpass the results of individual experts.

3.5 Comparison between Tree-Search Based Methods

In this section, the qualitatively and quantitatively comparison between the results obtained with the methods presented in Section 3.3 and Section 3.4 is reported and commented.

Table 3.8: Results (PSNR) of the multi-user experiment.

		Tested				
		A	B	C	D	E
Trained	A	20.36	21.75	20.77	19.52	18.93
Trained	B	20.15	22.06	20.94	19.90	19.82
Trained	C	20.43	22.49	21.56	20.33	20.37
Trained	D	19.96	21.48	20.44	19.95	20.27
Trained	E	19.43	21.07	20.44	19.78	20.38

Table 3.9: Quantitative comparison between the Tree-Search based methods presented in the thesis on the Five-K and LoL datasets.

	Adobe Five-K				LOL		
	PSNR	SSIM	ΔE	LPIPS	PSNR	SSIM	ΔE
TreEnhance (Tree)	21.24	0.89	11.25	0.06	21.96	0.81	10.82
TreEnhance (Policy)	20.06	0.84	12.44	-	21.86	0.80	10.99
Mask-Based	21.56	0.87	10.77	0.06	19.50	0.81	12.80

3.5.1 Quantitative Analysis

The comparison between the results obtained with TreEnhance and the mask-based method presented in the previous section is detailed in Table 3.9. Specifically, the results of both inference strategies for TreEnhance are provided. Upon examining the outcomes of the photographic-image enhancement task using the Five-K dataset, it is evident that the Mask-Based method outperforms both TreEnhance strategies in terms of PSNR, showcasing superior overall conformity to the target images. Regarding ΔE, both the Tree strategy and the Mask-Based method yield lower values compared to the policy strategy. This observation signifies that these methods achieve a more accurate alignment with the color distribution of the reference images. The lower SSIM value for the mask-based method, as compared to TreEnhance, can be attributed to the algorithm's inherent nature. While the capability to enhance specific image regions yields superior color and luminance outcomes, challenges remain in transitioning between enhanced areas. This concern has been alleviated through the incorporation of the parameter α. Finally, LPIPS scores were notably lower for all methods, suggesting enhanced perceptual similarity.

In the right part of the table the comparison of the results obtained for the low-light image enhancement task on the LOL dataset are reported. From the table is is possible to observe that the strategies from TreEnhance, that work on the whole image (global editing), are more effective than the targeted approach of the mask-based method. Specifically, the policy and tree strategies showed higher PSNR and lower ΔE values, which means they improved the color and appearance of the images more. This suggests that when dealing with tough tasks like low-light enhancement, the overall brightness information is more useful than focusing on smaller brightness details. On the other hand, the mask-based strategy performed really well in terms of SSIM. This is because it does not introduce noise or undesirable changes to the images. It maintains the image quality, which is reflected in its competitive SSIM values. In summary, the results highlight a trade-off between the global and local approaches. While the global methods show better color improvement, the mask-based approach is competitive at keeping the image structure intact. This sheds light on the balance between different enhancement techniques and their impact on image quality.

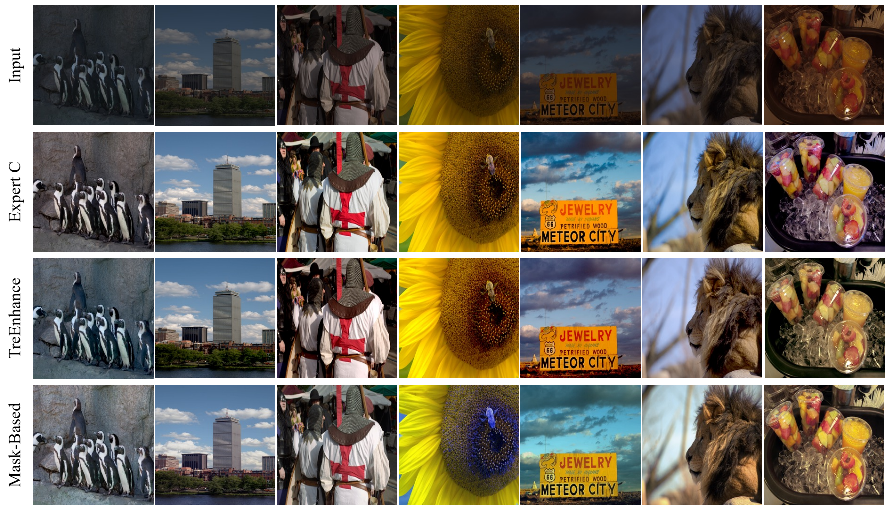


Figure 3.14: Qualitative comparison between Tree-Search based methods on the Five-K dataset

3.5.2 Qualitative Analysis

As an additional analysis, a comparison is drawn between the results achieved using the main TreEnhance strategy (employing the Tree inference strategy) and those

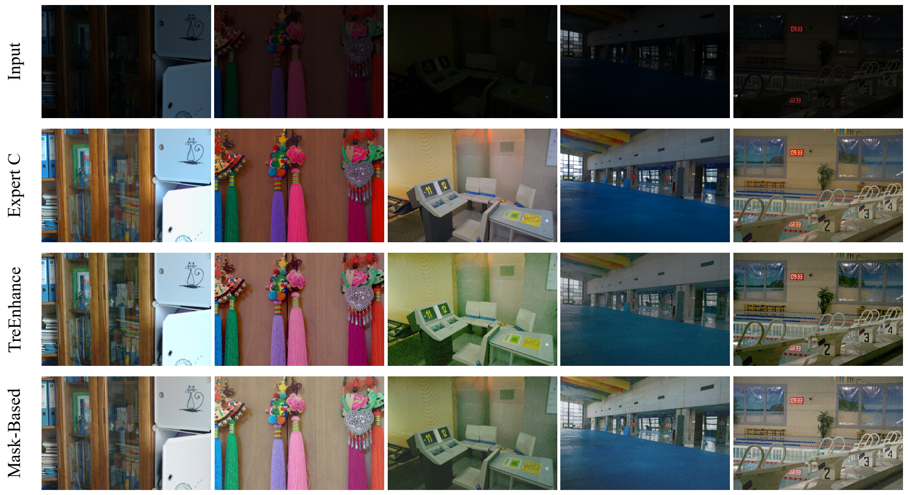


Figure 3.15: Qualitative comparison between Tree-Search based methods on the LOL dataset

obtained from the mask-based method. Figure 3.14 displays a side-by-side comparison of 7 test images generated by both techniques. Notably, the images produced by these methods exhibit considerable resemblance, a characteristic further supported by the alignment of the discussed PSNR, LPIPS, and ΔE values. Nevertheless, certain distinctions do emerge. In the fourth column, both methods accurately enhance the petals of the sunflower, yet the more intricate central part is more adeptly handled by TreEnhance. A comparable scenario unfolds in the subsequent column, where the mask-based method achieves a more authentic sky color in contrast to TreEnhance. Despite these subtle differences, both methods demonstrably perform well on the test set. This underscores the potential of leveraging a fusion of tree-search theory and deep reinforcement learning to enhance low-quality images. This research avenue showcases its efficacy through the successful image enhancement, as corroborated by the comprehensive evaluation of the test images.

Figure 3.15 illustrates the visual comparison carried out on the LOL dataset. This comparison unveils that the images generated using TreEnhance exhibit a clearer and more distinct style compared to those produced by the mask-based method. This enhancement in style is consistent with the elevated metric values, underscoring TreEnhance's capacity to effectively enhance low-light images. While the images rendered by the mask-based method may possess slightly less saturated colors, their structural integrity remains intact, as indicated by the high SSIM value.

A characteristic common to both methods is the absence of artifacts in the images generated for the two test sets. This artifact-free quality is of considerable significance and is guaranteed by both approaches. Furthermore, their white-box nature enhances their effectiveness as options for enhancing low-quality images in scenarios demanding both explainability and artifact-free methods.

3.6 Explaining Image Enhancement Black-Box Methods through a Path Planning Based Algorithm

In many image-to-image methods, a low-quality image is transformed via a neural network into a high-quality version while preserving details and semantic content. While these methods can be highly effective, they come with significant limitations: they typically operate on images of fixed resolutions, may introduce visible artifacts, and produce results that are not easily interpretable, following a black-box approach. This lack of interpretability can hinder their application in various fields, such as medicine and law, where understanding the enhancement steps applied by the algorithm is crucial.

In recent years, artificial intelligence (AI) and deep learning-based methods have demonstrated exceptional performance in numerous domains and applications, including computer vision, natural language processing, and time series analysis, among others. However, a significant challenge associated with AI methods is their inherent lack of interpretability. To address this, various works have been presented to provide explanations for neural networks' predictions, rendering them interpretable for end-users. Examples of eXplainable AI (XAI) algorithms include Grad-Cam [146], DeepLift [149], SHAP [117], DUBLID [100], and others [128, 136, 171, 205]. However, despite these advancements, there remains a need for an XAI algorithm specifically tailored to image enhancement.

In this section, we introduce the first eXplainable AI (XAI) algorithm designed for explaining image enhancement methods. This algorithm offers a step-by-step breakdown of the results generated by black-box image enhancement methods. It accomplishes this by employing a modified A* algorithm to replicate the enhancement process of another method. Through this emulation, it provides users with insights into the enhancement process. This algorithm, referred to as eXIE, is proficient at mimicking the enhancement process of image-to-image translation models for image enhancement. It is not an independent enhancement method but serves the purpose of delivering understandable outputs for existing methods.

eXIE accepts an input image and its translated version, which is obtained using

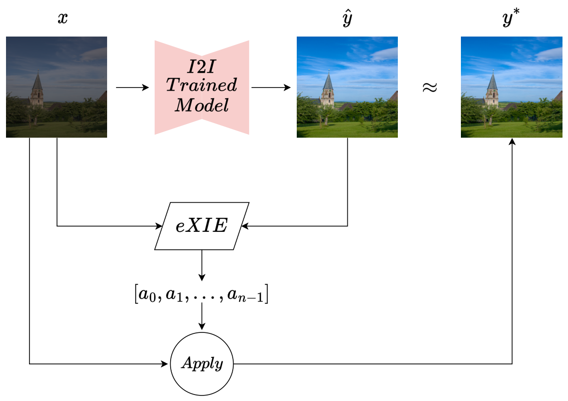


Figure 3.16: Schematic view of eXIE. Given as input a low quality image x, a pretrained model for image enhancement is used to obtain an high quality version $\hat{y}$. Both, the low quality and the high quality versions of the images are used to execute a modified version of the A* algorithm in order to find the shortest sequence of enhancing operators $[a_0, a_1, \ldots, a_{n-1}]$ that emulates the enhancement process. Once this sequence is obtained, each operator is applied to the low quality version to enhance it.

any other image enhancement method. It outputs a sequence of enhancing operators that transform the input image into an approximation of the translated output. To achieve this, eXIE employs a customized version of the A* search algorithm. Figure 3.16 provides an overview of the proposed method's operation.

eXIE finds utility in various application scenarios. For instance, it can serve as a foundational framework for professional photographers. They can employ it to establish a baseline for their work, which can then be fine-tuned and adapted as needed, saving them the effort of starting from scratch. Additionally, it can be a valuable educational tool for beginners, helping them comprehend when and how to apply image processing operators effectively.

To assess the quality of the sequences generated by the proposed method, extensive experiments were conducted on the Five-K dataset. These experiments involved comparing the outputs of several other state-of-the-art methods with their respective versions enhanced by eXIE. The reduction in accuracy due to the inclusion of eXIE was minimal, and in some cases, the method even managed to enhance the underlying enhancement method. Furthermore, eXIE demonstrated impressive results when applied to high-resolution images.

This section presents the following main contributions:

- The eXIE algorithm for the explanation of the output of existing image enhancement methods. This is one of the first works combining path finding and image enhancement.
- eXIE is one of the first explainable algorithms especially designed to obtain a step-by-step interpretation of image enhancement methods outputs.
- A novel heuristic function utilized by the modified A* algorithm that enables the quick discovery of sequences of enhancing operators.
- An in-depth experimentation, in which it is shown how eXIE was able to explain the outputs of several state-of-the-art methods with minimal loss in performance, as well as to reproduce with high fidelity human-retouched images.

This work introduces a novel approach in the field of image enhancement and eXplainable Artificial Intelligence (XAI), aiming to enhance accessibility and comprehensibility for users.

Section 3.6.1 elaborates on the proposed method and its functionality. Section 3.6.2 presents the results obtained by applying the method to images generated by several state-of-the-art image enhancement methods. The acquired sequences are analyzed. In Section 3.6.3, we delve into additional experiments and results, including those conducted on high-resolution images and using expert-retouched images as targets.

3.6.1 Method

The objective of the proposed method, known as eXIE, is to discover an equivalent sequence of enhancing operators capable of emulating the enhancement process of another state-of-the-art method. The search for these sequences is conducted through a modified version of the A* algorithm [70]. This section will begin by introducing the original A* algorithm and detailing the adaptations made to design it to the image enhancement problem. Subsequently, it will conclude by presenting the novel heuristic function specifically defined for eXIE and providing the complete algorithm along with pseudocode.

Fundamentals of A^* algorithm

A* is an efficient pathfinding algorithm capable of finding the shortest path from an initial node to a final node in a graph, if such a path exists. The algorithm utilizes two functions to evaluate nodes:

- The backtrack function $g(x)$, which calculates the length of the path from the starting node to node x.

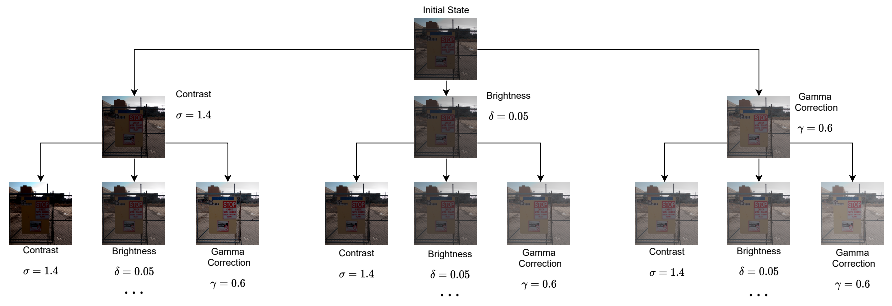


Figure 3.17: Example of the graph traversed by the eXIE algorithm. For space reasons, the considered graph is generated using only three editing operators over all the channels of the images and it is truncated after two levels.

- The heuristic function $h(x)$, estimating the length of the optimal path connecting the current node x to the target node. To ensure solution optimality, the heuristic must be optimistic, i.e., its estimate must not exceed the actual distance.

These two functions are combined, resulting in a value of $f(x) = g(x) + h(x)$ assigned to each node. The algorithm sequentially visits nodes in ascending order of their f values. The search process concludes when the final node is reached.

While defining the backtrack function $g(x)$ is relatively straightforward, careful consideration must be given to designing the heuristic function $h(x)$. A good heuristic function efficiently guides the algorithm toward the correct solution, whereas an incorrect function may lead to infinite loops and disrupt the search process.

The proposed method interprets an image enhancement task as a graph, enabling traversal using the eXIE algorithm. Unlike the A* algorithm, which terminates only upon reaching the final state, two additional stopping criteria for the search process have been established. The subsequent subsections provide a detailed presentation of the graph model, the newly developed heuristic function for graph traversal, and the complete pseudocode of the algorithm.

eXIE

In this work, the graph is composed by nodes that represent images, and edges connecting two nodes represent image processing operators. A link between two nodes in the graph signifies a transformation from one image (node) to another, achieved by applying an editing operator to the former. Figure 3.17 provides an illustration of the graph traversed by the search algorithm.

The initial node is identified by the low-quality image x in need of enhancement.

The final node corresponds to the image ŷ, generated using a different enhancement method. The objective is to discover the shortest sequence of editing operators $[a_0, a_1, \ldots, a_{n-1}]$ that converts x into an image y* closely resembling ŷ (formally, $y^* = a_{n-1}(a_{n-2}(\ldots a_0(x)))$).

The editing operators considered here are a small set of commonly used general filtering functions in image processing. In the following, x_{ijc} represents channel $c \in R, G, B$ of the pixel with coordinates (i, j):

- *Brightness adjustment*:

$$x_{ijc} \rightarrow x_{ijc} + \delta. \tag{3.19}$$

  The selected parameters are $\delta \in \Delta = \{-0.05, +0.05, -0.005, +0.005\}$, and they can be applied to all color channels or a single color channel.

- *Contrast adjustment*:

$$x_{ijc} \rightarrow \mu_c + \sigma \times (x_{ijc} - \mu_c), \tag{3.20}$$

  where μ_c is the average channel value. The variants considered include $\sigma \in \Sigma = \{0.9, 1.4\}$, and the operator can be applied either channel-wise considering all channels or to just one.

- *Gamma Correction*:

$$x_{ijc} \rightarrow (x_{ijc})^{\gamma}. \tag{3.21}$$

  The two values considered for γ are $\gamma \in \Gamma = \{0.6, 1.05\}$, and the transformation can be applied either channel-wise considering all channels or to just one.

Values and operators within the sets Δ, Γ, and Σ have been chosen to enhance the input image with a reasonable number of operator applications. Smaller values would produce similar results but would extend the searching process duration. It is assumed that the input pixel values fall within the range of $[0, 1]$, and all resulting values are clipped to remain within this range. In total, 32 image processing operators were considered.

Heuristic Function

The heuristic function plays a crucial role in the search algorithm by providing an estimate of the distance between a given node and the final node. To ensure the solution's optimality, it's essential for the heuristic function to be optimistic, meaning it should underestimate the actual distance between the nodes. However, if the heuristic function is excessively optimistic, it might slow down the algorithm's progress toward the target.

The heuristic function has been defined by considering how many times one of the operators must be applied to transform a single pixel value into the target value. To be more specific, for each pixel value x_{ijc}, three counters are computed: the Brightness Counter, the Contrast Counter, and the Gamma Correction Counter. The Brightness counter represents the number of times the brightness operator must be applied to x_{ijc} to reach the target value $\hat{\mathrm{y}}_{ijc}$.

$$N^{(B)}_{ijc} = \min_{\delta \in \Delta} \frac{|\mathrm{x}_{ijc} - \hat{\mathrm{y}}_{ijc}|}{|\delta|}. \tag{3.22}$$

The Contrast Counter has been defined similarly. However, it's important to note that there are special cases where it's not possible to transform the pixel value into the desired target using only this operator.

$$N^{(C)}_{ijc} = \begin{cases} \frac{1}{\log \max \Sigma} \log \frac{\hat{\mathrm{y}}_{ijc} - \mu_c}{\mathrm{x}_{ijc} - \mu_c} & \text{if } \mathrm{x}_{ijc} > \mu_c \text{ and } \hat{\mathrm{y}}_{ijc} > \mu_c, \\ \frac{1}{\log \min \Sigma} \log \frac{\hat{\mathrm{y}}_{ijc} - \mu_c}{\mathrm{x}_{ijc} - \mu_c} & \text{if } \mathrm{x}_{ijc} < \mu_c \text{ and } \hat{\mathrm{y}}_{ijc} < \mu_c, \\ \infty & \text{otherwise.} \end{cases} \tag{3.23}$$

Finally, the Gamma Correction Counter is defined as:

$$N^{(G)}_{ijc} = \begin{cases} \frac{1}{\log \max \Gamma} \log \frac{\log \hat{\mathrm{y}}_{ijc}}{\log \mathrm{x}_{ijc}} & \text{if } \mathrm{x}_{ijc} \geq \hat{\mathrm{y}}_{ijc}, \\ \frac{1}{\log \min \Gamma} \log \frac{\log \hat{\mathrm{y}}_{ijc}}{\log \mathrm{x}_{ijc}} & \text{if } \mathrm{x}_{ijc} < \hat{\mathrm{y}}_{ijc}, \end{cases} \tag{3.24}$$

The heuristic function h is defined for the entire image x as an upper bound on the number of times any given operator must be applied to match each pixel with its corresponding target value.

$$h(\mathrm{x}) = \max_{ijc} \min_{a \in [B,C,G]} N^{(a)}_{ijc}. \tag{3.25}$$

Regarding the backtrack function $g(\mathrm{x})$, it counts the number of times an operator has been applied to the initial image to obtain the image x. To limit the algorithm's search time, two additional modifications have been introduced. The search terminates when the difference between the actual and target nodes falls below a predefined threshold $\|\mathrm{x} - \hat{\mathrm{y}}\| < \tau$. Moreover, it can also terminate when the number of explored nodes exceeds a set limit L. In such cases, the visited node closest to the target is selected, and the path (i.e., the sequence of enhancing operators) from the root to this node is obtained as output, along with the enhanced image. It has been observed that when the number of visited nodes exceeds the value of $L = 7000$, the accuracy of the output images tends to be very stable. In accordance with the maximum number of explored nodes L, the threshold value has been set to $\tau = 2$. These choices have been made to strike a balance between the quality of the output image and the time required to explore the graph.

The pseudo-code for the whole procedure is reported in Algorithm 1.

Algorithm 1 eXIE

```
Input: image x,
target $\hat{\mathrm{y}}$,
set of image operators $\mathcal{A}$,
target threshold $\tau$,
maximum number of visited nodes $L$.

Output: sequence of operators $\langle a_0, a_1, \ldots a_{n-1} \rangle$ taken from $\mathcal{A}$, such that
$a_{n-1}(a_{n-2}(\ldots a_0(\mathrm{x})))$ is approximately equal to $\hat{\mathrm{y}}$.

$O \leftarrow \{\mathrm{x}\}$                                              ▷ Open set
$C \leftarrow \emptyset$                                                ▷ Closed set
$g(\mathrm{x}) \leftarrow 0$
$\mathrm{path}(\mathrm{x}) \leftarrow \langle\rangle$
while $|C| < L$ do
    $\mathrm{x}_{\mathrm{current}} \leftarrow \arg\min_{\mathrm{x}' \in O} g(\mathrm{x}') + h(\mathrm{x}')$
    $C \leftarrow C \cup \{\mathrm{x}_{\mathrm{current}}\}$
    $O \leftarrow O \setminus \{\mathrm{x}_{\mathrm{current}}\}$
    if $\|\mathrm{x}_{\mathrm{current}} - \hat{\mathrm{y}}\| < \tau$ then
        return $\mathrm{path}(\mathrm{x}_{\mathrm{current}})$
    end if
    for $a \in \mathcal{A}$ do
        $\mathrm{x}_{\mathrm{new}} \leftarrow a(\mathrm{x}_{\mathrm{current}})$
        $g(\mathrm{x}_{\mathrm{new}}) \leftarrow g(\mathrm{x}_{\mathrm{current}}) + 1$
        Compute $h(\mathrm{x}_{\mathrm{new}})$ according to (3.25)
        $\mathrm{path}(\mathrm{x}_{\mathrm{new}}) \leftarrow \mathrm{path}(\mathrm{x}_{\mathrm{current}}) \oplus \langle a \rangle$      ▷ append $a$
        $O \leftarrow O \cup \{\mathrm{x}_{\mathrm{new}}\}$
    end for
end while
$\mathrm{x}_{\mathrm{best}} \leftarrow \mathrm{argmin}_{\mathrm{x}' \in C \cup O} \|\mathrm{x}' - \hat{\mathrm{y}}\|$
return $\mathrm{path}(\mathrm{x}_{\mathrm{best}})$
```

Table 3.10: Comparison of state-of-the-art enhancement methods on the Adobe Five-K dataset with and without eXIE.

Metric	Method	Original	eXIE	Metric	Method	Original	eXIE
LPIPS↓	Exposure [75]	0.16	0.13	$\Delta E\downarrow$	Exposure [75]	13.57	13.06
	CycleGan [204]	0.16	0.14		CycleGan [204]	12.95	13.88
	DaR [130]	0.10	0.11		DaR [130]	13.05	11.09
	Pix2Pix [80]	0.09	0.12		Pix2Pix [80]	8.84	11.03
	HDRNet [61]	0.08	0.10		HDRNet [61]	9.53	10.66
	Star-DCE [199]	0.08	0.09		Star-DCE [199]	8.45	9.75
	Parametric [13]	0.07	0.10		Parametric [13]	8.52	9.86
	TreEnhance [42]	0.06	0.10		TreEnhance [42]	11.25	11.73
	DCE-Net [69]	0.13	0.12		DCE-Net [69]	11.77	12.06
	DCE-Net-Pooling [69]	0.15	0.13		DCE-Net-Pooling [69]	12.13	12.47
PSNR↑	Exposure [75]	18.74	19.45	SSIM↑	Exposure [75]	0.81	0.83
	CycleGan [204]	19.38	19.38		CycleGan [204]	0.78	0.82
	DaR [130]	20.91	21.00		DaR [130]	0.86	0.86
	Pix2Pix [80]	23.05	21.41		Pix2Pix [80]	0.86	0.86
	HDRNet [61]	22.31	21.55		HDRNet [61]	0.89	0.87
	Star-DCE [199]	23.55	22.41		Star-DCE [199]	0.89	0.88
	Parametric [13]	23.20	22.26		Parametric [13]	0.90	0.88
	TreEnhance [42]	21.24	20.32		TreEnhance [42]	0.89	0.87
	DCE-Net [69]	20.47	20.14		DCE-Net [69]	0.85	0.84
	DCE-Net-Pooling [69]	20.33	19.83		DCE-Net-Pooling [69]	0.85	0.83

3.6.2 Results

In this section, the results obtained by applying eXIE to the methods presented in the previous section are presented. The performance, both quantitatively and qualitatively, with and without the application of the proposed algorithm, has been compared. The final part of this section analyzes the explainable aspect of eXIE by inspecting the sequences of enhancing operators produced by the method.

Quantitative results

Table 3.10 compares the metrics computed on the output images of the considered methods and those computed on the images produced by eXIE.

The results demonstrate that eXIE can accurately emulate the original methods. There is only a slight loss in terms of the four metrics considered (PSNR, ΔE, LPIPS, and SSIM), and in some cases, the images generated by eXIE even outperform the originals. This is attributed to eXIE's design, which focuses on preventing the introduction of artifacts, a task more challenging than reproducing correct enhancements. This effect is particularly notable for methods like Exposure, CycleGAN, and DaR.

Significant differences in accuracy, specifically in terms of PSNR and ΔE, are observed only for the most precise methods. Nevertheless, this relatively minor decrease in accuracy is often acceptable in many applications in exchange for the explainability offered by eXIE. Across all experiments, differences in terms of LPIPS and SSIM remain negligible, underscoring the high accuracy of the proposed algorithm in preserving image content.

Qualitative Results

From the qualitative analysis of the images produced by eXIE shown in Figure 3.18 it is possible to notice how eXIE is able to emulate the enhancement process of the original methods with high fidelity. For instance, by looking at the images produced by eXIE on CycleGan (column 6) it is possible to see that the artifacts have been removed and the color balance for the images obtained by eXIE is better than the version produced by using the image-to-image translation method. Similarly, for DaR (column 3) and Exposure (column 2), it is possible to observe that eXIE is able to obtain better saturation and contrast in the final image with respect to the result of the original models.

When applied to the best models, such as Star-DCE (column 1), Pix2Pix (column 4), HDR (column 5), and Parametric (column 7), eXIE accurately replicates their results. These findings underscore the potential of eXIE as a tool for explaining and enhancing the output of existing image enhancement methods.

The selection of editing operators has been carefully made to prevent the introduction of artifacts in the output image. Unlike other image-to-image translation methods that rely on complex models and may generate undesirable changes, the method employs a straightforward approach to image enhancement. The operators utilized in eXIE are fundamental and well-established in the field of image processing, enhancing the method's comprehensibility and control over the results. Consequently, the proposed approach can generate high-quality output images with minimal accuracy loss and a high level of interpretability.

One of the key features of the proposed solution is its ability to preserve the semantic content of the image. Even when the target method introduces distortion in the output image, the enhancing operators designed for eXIE do not significantly alter the image's content. The "Brightness Adjustment","Contrast Adjustment" and "Gamma correction" operators works by modifying the color curve while preserving the image's structure.

Sequence inspection

For further analysis, the sequences generated by eXIE (Figure 3.19) were examined. The sequences produced by the considered state-of-the-art methods exhibit consider-

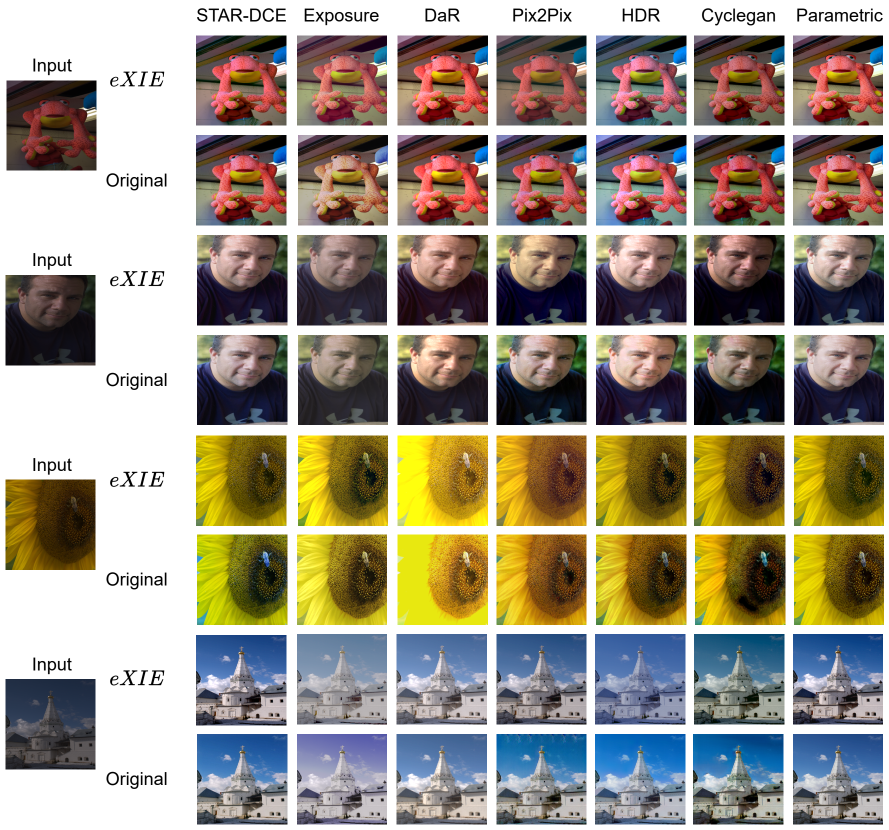


Figure 3.18: Comparison of the images obtained by eXIE and by the original methods.

able heterogeneity and follow different orders of operator application.

Typically, the initial operators applied are those that globally increase the image's brightness. These operators are applied to all pixels across the three color channels of the image. Subsequently, when the image achieves a satisfactory overall balance, more nuanced operators are applied. For example, in the HDRNet column, the figure demonstrates how the algorithm determines that, following the initial brightness adjustment across the entire image, the best subsequent action is to reduce the value of the red channel, resulting in a refinement of the color distribution.

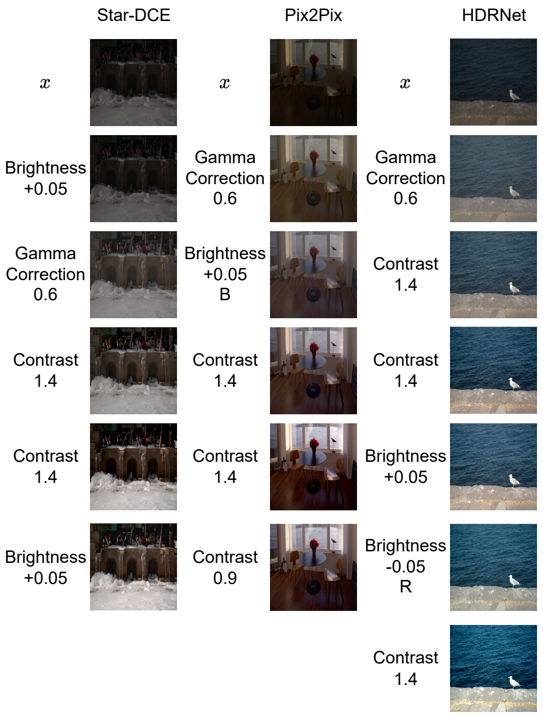


Figure 3.19: Example of the sequences obtained with the application of eXIE on Star-DCE, Pix2Pix and HDRNet.

3.6.3 Additional Experiments

In this section, two additional experiments that were conducted are reported. In the first experiment, eXIE was applied to low-resolution images to generate sequences of editing operators. Subsequently, these sequences were applied to the original resolution images. The primary goal of this experiment was to speed up the sequence generation process, as the sequences were later applied to images of their original sizes without any limitations. In the second experiment, the ground truth images from the Five-K dataset have been used as the target. This case study aimed to assess the capability of the presented method in explaining the manual enhancement procedures performed by expert photographers.

Enhancement of high resolution images via UNET

Many image-to-image translation methods, such as Pix2Pix or CycleGAN, operate with images of fixed dimensions. When dealing with the enhancement of very high-resolution images to avoid introducing artifacts, it is crucial to maintain consistent spatial dimensions across all layers of the model. To address this challenge, we employed the method on significantly lower-resolution versions of the input images and subsequently applied the generated sequences to the original high-resolution images. This approach requires minimal computational resources in terms of memory and time and does not compromise the output quality.

To elaborate further, when provided with a high-resolution image from the Five-K dataset, we first resize it to a very low resolution (e.g., 32 × 32). This low-resolution version serves as input to a specially designed convolutional neural network based on the UNet [137]. eXIE is then applied to these resulting low-resolution images, and the sequence of operators identified is subsequently applied to the original high-resolution input image. Figure 3.20 illustrates this approach.

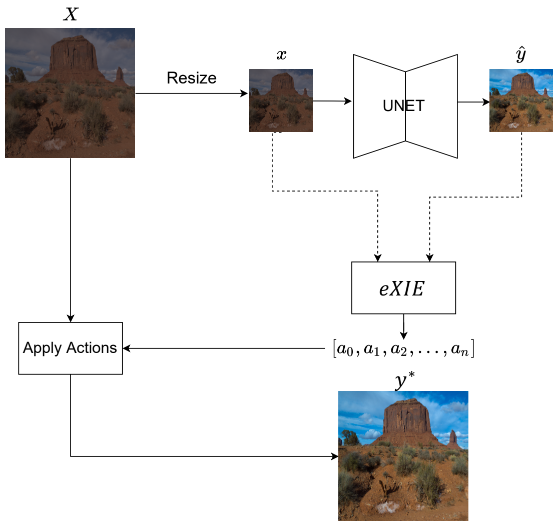


Figure 3.20: The high resolution input image X, is resized obtaining a low resolution version x. This image is enhanced using the previously trained UNet. This architecture provide as output the image ŷ. The images x and ŷ are used to execute the eXIE algorithm and obtaining the sequence of enhancing operators to enhance the high resolution image X and obtaining Ŷ.

The neural network utilized in this experiment consists of two primary components: an encoder and a decoder. In the encoder, each of the four blocks is comprised of a convolutional layer with a kernel size of 4×4 and a stride of 2, followed by batch normalization and the application of the leaky ReLU activation function. The decoder, on the other hand, comprises four blocks, each containing a transposed convolutional layer with a kernel size of 4×4 and a stride of 2, followed by batch normalization, ReLU activation, and dropout (with a dropout rate of $p = 0.05$). The output layer of the architecture employs a sigmoid activation function to constrain the output pixel values within the range of $[0, 1]$.

During the training of the network, the high-resolution input images X from the Five-K training dataset are resized to obtain a lower-resolution version x. These lower-resolution images are then provided as input to the network, resulting in the generation of enhanced versions $\hat{\mathrm{y}} = f(\mathrm{x}; \theta)$. The images $\hat{\mathrm{y}}$ are compared to the target images y using binary cross-entropy (BCE) as the loss function.

$$BCE = -\frac{1}{HWC} \sum_{c=0}^{C} \sum_{i=0}^{W} \sum_{j=0}^{H} \mathrm{y}_{jic} \cdot \log(\hat{\mathrm{y}}_{jic}) + (1 - \mathrm{y}_{jic}) \cdot \log(1 - \hat{\mathrm{y}}_{jic}). \tag{3.26}$$

The model was trained using standard data augmentation techniques, including cropping, resizing, random flips, and rotations, applied to the image pairs. Training was conducted for 600 epochs with a batch size of 32. Mini-batch gradient descent and the AdamW optimizer were used to update the parameters of the UNet model, with a learning rate of 5e-3. The learning rate was decayed by a factor of 0.1 every 100 epochs, starting from the 200th epoch.

After training, the network was employed to enhance the low-resolution versions of the images from the Five-K test set. Subsequently, these enhanced images were processed by eXIE in conjunction with the original low-resolution inputs.

Table 3.11: Results of the experiment with the neural network for low-resolution enhancement (on low-resolution images) and the application of eXIE (on high-resolution images).

Metric	UNet on Low-Res	eXIE on High-Res
PSNR↑	23.36	22.09
ΔE ↓	8.55	10.31
SSIM↑	0.93	0.88

Table 3.11 provides a summary of the results obtained from the application of eXIE to the low-resolution images. The first column displays the performance of the neural model on the low-resolution images, while the metrics in the second column are computed on the high-resolution images. Remarkably, eXIE exhibits strong performance even when applied to very high-resolution images, such as the original ones from the Five-K dataset.

The metrics computed on the images enhanced using eXIE demonstrate its effectiveness. A comparison with the results presented in Table 3.10 reveals that eXIE outperforms other methods like Exposure, CycleGan, and Dar in enhancing high-resolution images. The performance metric values closely resemble those obtained with HDRNet.

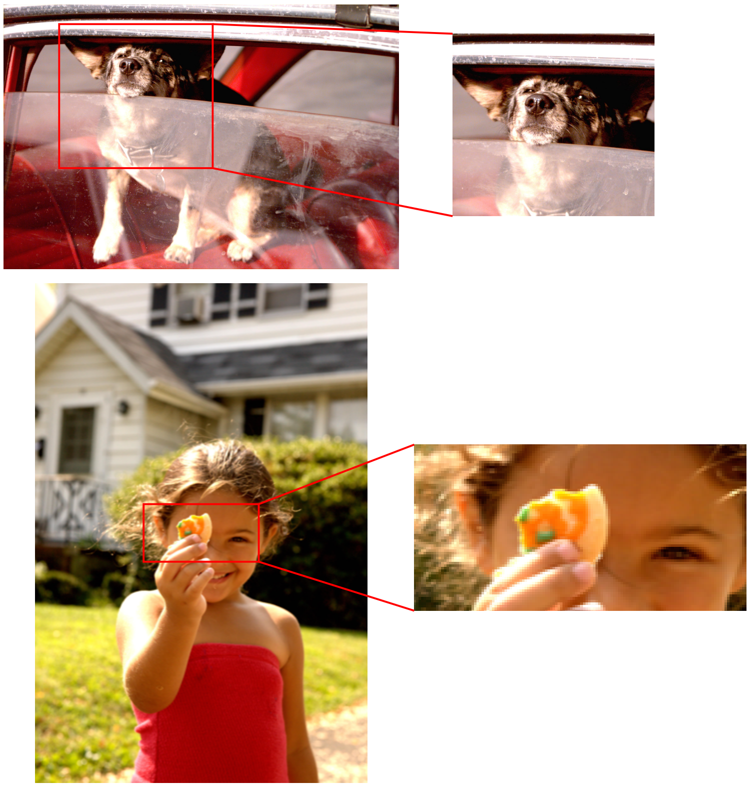


Figure 3.21: Example of the application of eXIE on two high resolution image from Five-K test set after being processed by the UNet.

Table 3.12: Results of the application of eXIE on the test images of the Five-K dataset using the images of the 5 available experts as target.

Metric	LPIPS↓	PSNR↑	$\Delta E\downarrow$	SSIM↑
expA	0.01	27.76	5.83	0.92
expB	0.01	29.14	5.16	0.95
expC	0.02	25.44	7.43	0.91
expD	0.01	28.02	5.73	0.94
expE	0.02	26.50	6.69	0.93

Analyzing the images reported in Figure 3.21, it becomes evident that there are no noticeable artifacts, and the color balance applied by eXIE is accurate. Furthermore, upon closer examination of the details, it is apparent that the visual content of the image is faithfully preserved. This observation is corroborated by the high SSIM value reported in the previous table.

Case Study: Human Target

In the last experiment, eXIE was employed to reverse engineer the work of an expert photo editor. The objective was to replicate the image enhanced by a human expert as a sequence of elementary editing operations. This scenario has educational applications, enabling beginner photo editors to learn how to achieve specific editing effects by observing how eXIE breaks them down into a sequence of operations.

To accomplish this, eXIE was applied to the images in the Five-K dataset with the aim of reproducing the versions enhanced by the experts. The results of this experiment are summarized in Table 3.12.

The results in Table 3.12 demonstrate that eXIE is capable of providing high-quality images that replicate the enhancement process applied by the experts in the Five-K dataset. Furthermore, the values reported for the considered metrics are exceptionally high, confirming the algorithm's ability to emulate not only the enhancement process of image-to-image translation models but also the sequence of operations chosen by human experts.

Analyzing the distributions of the operations composing the sequences selected by eXIE (Figure 3.22), it is evident that the most frequently chosen operations are those that modify all the color channels. These distributions are quite similar across all the experts and even for the UNet, with only minor differences in the frequency of single-channel operations (particularly in the case of brightness).

Upon closer examination of the action probability distributions for each individual expert, it becomes apparent that there are general trends and specific preferences. For example, when observing Expert A's action distribution, it is noticeable that brightness and gamma correction over all three channels have nearly equal probabilities, indicating that these two actions are often interchangeable for Expert A.

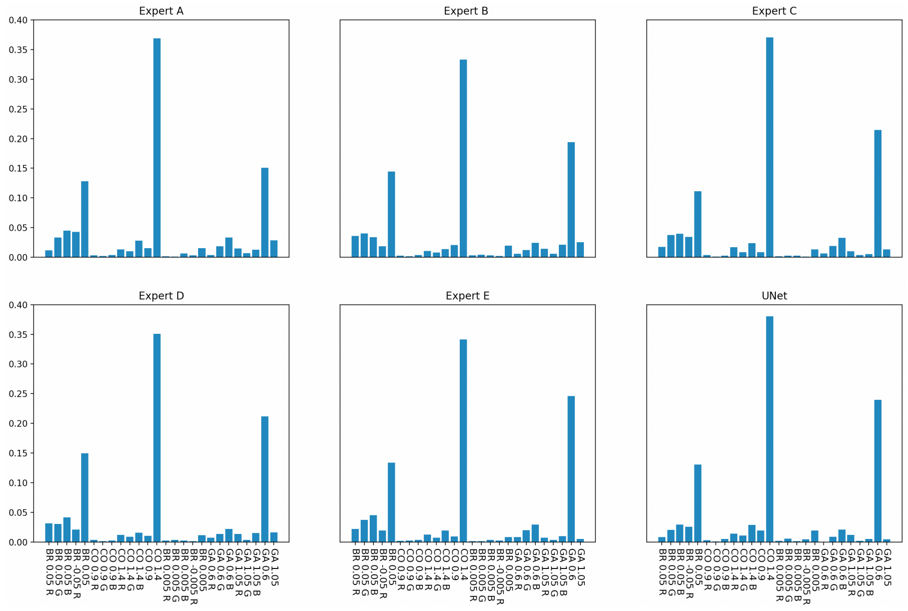


Figure 3.22: Actions distributions of the sequences obtained applying eXIE to replicate the enhancing process applied by Five-K experts and the UNet.

3.7 Summary

This chapter introduces three artifacts-free white-box methods for image enhancement. These approaches present effective solutions within the realm of fully interpretable methods for enhancing images. Specifically, in Section 3.3, the TreEnhance method is proposed as a lightweight, fully automatic, and explainable global image enhancement technique. Its solutions are inherently explainable, being composed of a sequence of simple image processing operations. Similarly, Section 3.4 outlines a mask-based image enhancement method. Differently from TreEnhance, this approach operates locally on smaller image regions, involving a single enhancement operator.

These methods exhibit strong performance in two image enhancement tasks using the Adobe Five-K and Low-Light datasets, as extensively detailed in Section 3.5.

Additionally, a novel white-box explainability method is introduced. This method employs a new heuristic function to generate interpretable sequences of image processing operators, emulating the enhancement process of image-to-image black-box translation methods.

This emerging family of image enhancement methods presents a promising trajectory in the realm of Explainable Computer Vision techniques. Their inherent generalizability allows slight operator adjustments for application in diverse computational photography tasks, such as image retargeting and deblurring. Furthermore, their interpretability positions them as valuable tools for integration into post-processing software. These tools not only enhance low-quality images but also guide novice photo-retouchers approaching complex tasks with confidence.

4 Tackling Photo Authorship Attribution and Photographic Style Transfer tasks: a Data-Driven approach

Professional photographs are distinguished by their high technical precision and profound capacity for conveying messages. The aesthetic merit of professional photographers' work is determined not solely by their subject choices but also by the expressiveness and uniqueness of their individual styles. Iconic photographers produce their shots with a distinct style, elevating photography to an art form. Certain photographers exhibit preferences for particular color tones, contrasts, shades, and more, rendering their photographs instantly recognizable. Nevertheless, identifying the author of a photograph solely based on its style presents an immensely challenging task. This challenge arises from the fact that a photograph's style results from a blend of various factors that are intricately related to its content.

The automatic classification of paintings, a closely related task, has garnered substantial attention in recent years [1, 18, 19, 85]. Conversely, only a limited number of studies have delved into the recognition of photographic styles [159]. One of the primary reasons behind this discrepancy is the lack of extensive datasets containing photographs from a diverse array of photographers, each possessing a clearly defined photographic style.

4.1 Introduction

In this chapter, PhotoStyle60, a dataset composed of 5708 images taken by 60 different photographers characterized by unique personal styles, is proposed. Additionally, a subset version of the dataset containing images from 10 clearly recognizable photographers is reported.

The dataset's potential was explored in two applications: authorship attribution and photographic style transfer. For authorship attribution, state-of-the-art image recognition models were trained and fine-tuned to categorize images from the dataset. For photographic style transfer, previously published neural style transfer models were trained to transfer the photographic style from the dataset to general content images sourced from the Five-K dataset[23]

Automatic recognition of a professional photographer from a photograph can serve

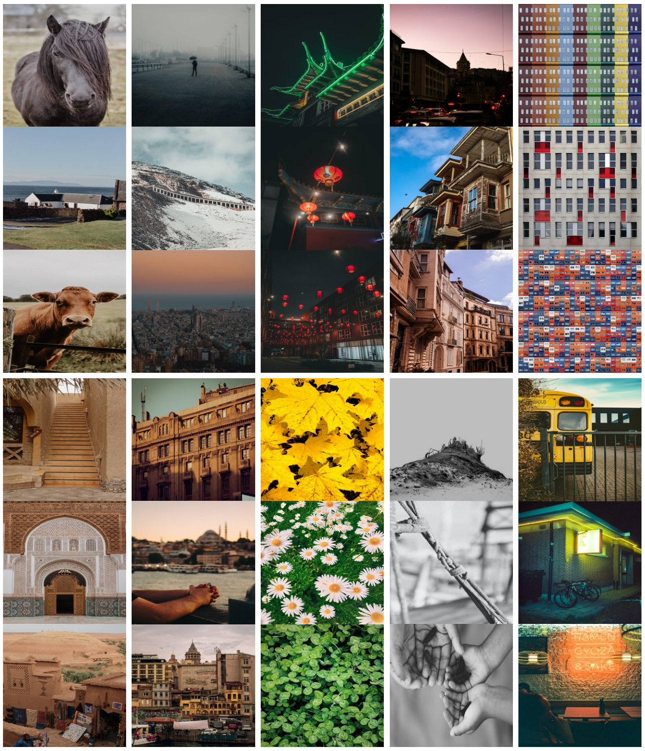


Figure 4.1: Examples of photographs included in the PhotoStyle60 dataset. Each triplet is representative of the style of one of the photographers in the dataset.

various purposes. It can aid in preserving intellectual property rights by identifying unauthorized use or appropriation of authorship. Furthermore, it can contribute to the development of accurate online image search engines that rely on photographs rather than manual keyword inputs. Additionally, considering the realm of universal beauty, a dataset like PhotoStyle60 could facilitate user studies aimed at objectively identifying universal aesthetic standards that unite the success of some of the most iconic contemporary photographers.

Additionally, a novel neural style transfer algorithm is proposed, which can transfer a photographer's style while maintaining the visual content of the image without introducing glitches or artifacts. In contrast to existing methods [59, 73, 97, 99, 167], which typically use a single style image for style transfer, the proposed approach leverages multiple style images from a single photographer. Building upon the foundation laid by [59], the method automates the selection of multiple photographs from an expert photographer to facilitate a broader style transfer to content images. This adaptation draws inspiration from the principles of single-style transfer solutions [72, 109, 169]. In the realm of education, this efficient style transfer method serves as a valuable tool for novice photographers. It enables them to easily adjust their photos to emulate specific styles, aiding in the exploration and comprehension of iconic photographers' unique visual languages. For commercial photography and advertising, such a tool holds immense potential for creating visually appealing images by replicating universally appreciated styles of iconic photographers.

To assess the effectiveness of the proposed algorithm, a comprehensive comparison was conducted against other style transfer methods through an extensive user study. The proposed solution ranked first in terms of the quality of the resulting images and the fidelity of the transferred style. Furthermore, it outperformed other methods by preserving the image content with fewer undesirable artifacts.

The main contributions of the chapter are:

- PhotoStyle60, a novel dataset designed for computational photography applications, comprising 5708 images contributed by 60 distinct photographers, each exhibiting a unique style, along with its reduced variant.
- A comprehensive series of experiments demonstrating the effectiveness of state-of-the-art techniques in authorship attribution and photographic style transfer.
- The introduction of an innovative photographic style transfer method that surpasses the current state of the art by incorporating multiple reference photographs.

Section 4.2 describes in detail the proposed PhotoStyle60. Section 4.3 presents the method proposed for multi-image style transfer. The results obtained on the new dataset with existing and new methods are reported Section 4.4. Finally, Section 4.5 concludes the chapter with new considerations and observations.

4.2 PhotoStyle60

To investigate the influence of personal photographic styles in digital photography, PhotoStyle60 was curated, a substantial dataset encompassing photographs contributed by 60 photographers. This collection includes renowned artists like Charlie Hamilton James, Ami Vitale, David LaChapelle, Irving Penn, Maurizio Lima, among others. For each photographer, a curated selection of their most distinctive works was sourced from the internet. Additionally, a set of 16 semi-professional photographers who have shared their works under the Creative-Common License was incorporated (see

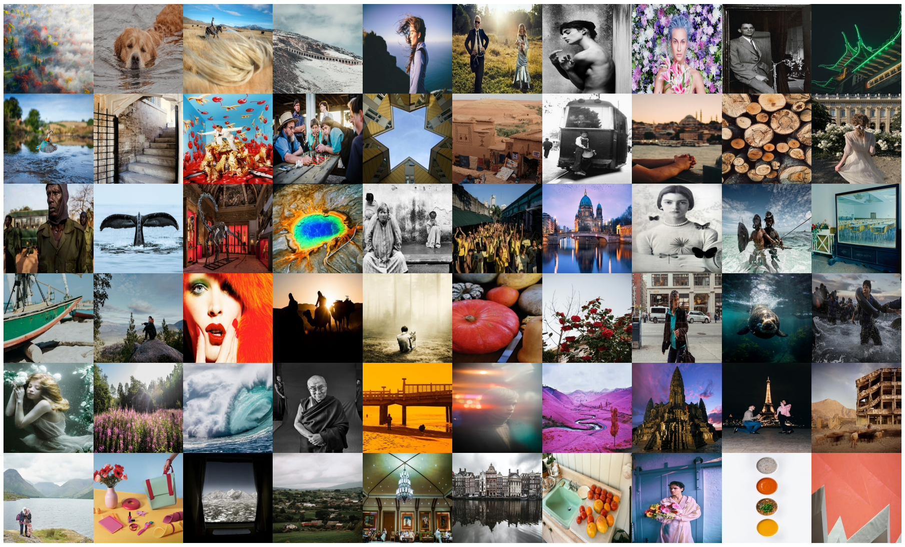


Figure 4.2: Example of images from the PhotoStyle60. Each of the images belong to one of the photographer in the dataset.

Figure 4.1). The complete list of photographers, along with the corresponding image counts, is presented in Table 4.1. Figure 4.2 further showcases a representative image from each of these photographers, demonstrating their unique and individualistic photographic styles.

The dataset exhibits a range of photos per photographer, varying from a minimum of 36 to a maximum of 252 images, with an average count of approximately 95 images. The photographers were chosen based on the presence of a distinct and recognizable style, with some specializing in specific genres like fashion or nature, while others displayed versatility across a broad spectrum of subjects.

All images within the dataset are high-quality digital photographs, and no scanned prints were included. Rigorous **manual** inspections were conducted to ensure the absence of watermarks, signatures, logos, borders, or any extraneous graphical elements. Subsequently, ImageMagick[110], an open-source tool, was employed for dataset preprocessing, which involved resizing the images to a standardized resolution of 256×256 pixels and conversion to the standard RGB color space (sRGB).

A major concern in collecting the dataset has been the relationship between style and content. In fact, in attributing the authorship of a photograph, both concepts may be useful. To verify that authors cannot be identified solely from the content of their

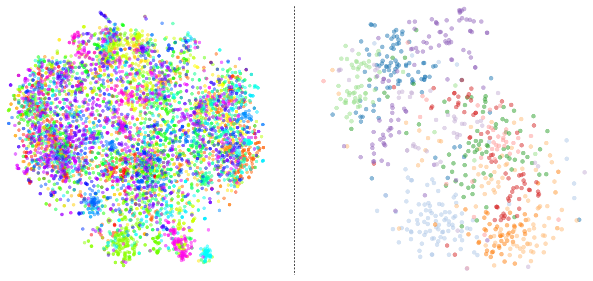

(a) (b)

Figure 4.3: Result of the application of t-SNE on PhotoStyle60 (a) and on PhotoStyle10 (b) features extracted with a SwinTransformer pretrained on ImageNet data.

photographs, neural network trained on ImageNet data to categorize the image content was employed and the t-SNE was applied to the features extracted from the dataset. Results are shown in Figure 4.3.

One of the primary considerations during dataset curation pertained to the intricate interplay between photographic style and content. In the context of attributing authorship to a photograph, both style and content may carry significance. To assess whether authors can be exclusively identified based on the content of their photographs, an experiment was conducted.

A pretrained neural network, that had been trained on ImageNet data for the purpose of content categorization, has been employed for extracting high level features. Then, the t-SNE (t-distributed stochastic neighbor embedding) algorithm was applied to the features. The outcomes of this analysis are visually presented in Figure 4.3.

Table 4.1: List of all the photographers in the dataset with the number of photographs.

Photographer	#photographs	Photographer	#photographs	Photographer	#photographs	Photographer	#photographs
Adam Senatori	36	Semi-professional 1	137	Ami Vitale	95	Semi-professional 2	90
Andy Bardon	99	Annie Leibovitz	99	Antigone Kourakou	100	Arjun Mark	49
Arnold Newman	97	Semi-professional 3	80	Charlie Hamilton James	46	Semi-professional 4	166
David LaChapelle	98	Dina Litovsky	78	Dirk Bakker	68	Semi-professional 5	252
Elliot Erwitt	64	Semi-professional 6	116	Semi-professional 7	228	Semi-professional 8	97
Fabio Bucciarelli	96	Frans Lanting	92	Gabriele Galimberti	89	George Steinmetz	90
Gianni Barengo Gardin	95	Hanna Reyes Morales	79	Ilhan Eroglu	55	Irving Penn	100
Jimmy Nelson	72	Julia Fullerton-Batten	81	Semi-professional 9	90	Lucy Foster	75
Mario Testino	95	Martin Edstrom	93	Martin Stranka	98	Semi-professional 10	106
Semi-professional 11	167	Matilde Pernille	38	Mauricio Lima	37	Michaela Skrovanova	68
Mira Nedyalkova	49	Nora Lorek	100	Paul Nicklen	96	Raghunath Ray Chowdhry	99
Semi-professional 12	113	Rich Gilligan	81	Richard Mosse	46	Robert Clark	85
Scander Aidoudi	57	Simon Norflok	95	Simon Roberts	98	Simone Bramante	99
Stefano De Luigi	99	Semi-professional 13	95	Tasneem Alsultan	89	Semi-professional 14	227
Willian Eggleston	93	Zuzu Valla	49	Semi-professional 15	109	Semi-professional 16	118

The t-SNE analysis revealed numerous overlapping clusters, implying that different photographers can capture similar content while employing entirely distinct photographic styles.

4.2.1 PhotoStyle10

To gain a deeper understanding of the challenges associated with recognizing photographic styles, a smaller-scale version of the dataset was created. Specifically, a subset comprising 722 photographs from ten photographers, each known for their distinct and easily recognizable style was curated. This reduced subset serves not only to alleviate computational demands and enhance training efficiency when resources are limited but also to improve the quality of learned features, given the clearly distinct styles represented.

Figure 4.3 (b) illustrates that, even with their distinctive styles, t-SNE struggles to completely differentiate all ten photographers based solely on the features depicting the content of their photographs.

Additional high-resolution images from both the PhotoStyle60 and PhotoStyle10 have been included in the supplementary materials.

4.3 Multi-Image Style Transfer Method

Style transfer methods based on feature matching entail combining a style image with a content image to transfer the style from one to the other. This process involves a neural model that extracts feature maps from both images across various layers. These feature maps guide an optimization procedure, resulting in an output image that retains the content while adopting the desired style.

In the realm of paintings, a single image often suffices to represent an entire style, thanks to the significance of primitive and easily identifiable features, such as individual brush strokes. Consequently, previous style transfer methods typically focused on a single image to convey the target style.

In contrast, in the domain of photography, distinguishing style from content is challenging, and using a single image to encapsulate a photographer's style is insufficient. Photographic styles tend to be subtler compared to those found in paintings, and forcibly transferring such styles, by increasing specific coefficients, often leads to the emergence of artifacts.

For these reasons, an innovative multi-image style transfer approach, that leverages multiple photographs automatically selected to represent the style of a target photographer, is proposed. Unlike feature matching methods relying on a single style image, the method aims to convey a more general photographic style derived from

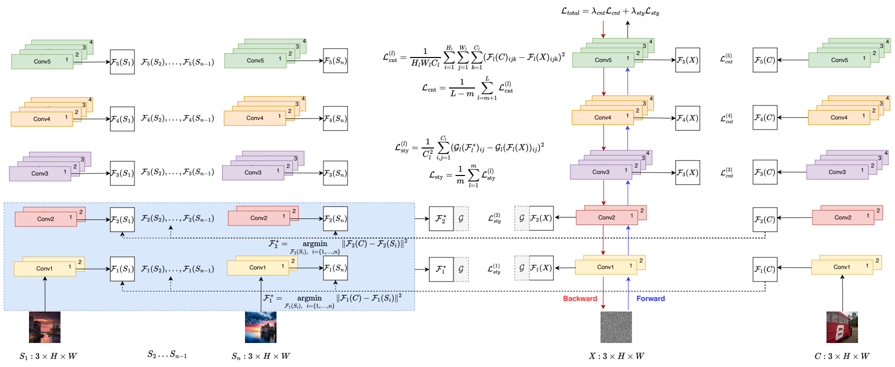


Figure 4.4: Method overview with $m = 2$. In the left part of the figure, the selection procedure of the candidate style feature maps is reported . On the right, the extraction of the content feature maps is presented. In the central part of the figure the image refinement step is shown. The random initialized image X is forwarded to the VGG19 extracting the five feature maps. The first two $\mathcal{F}_1(X)$ and $\mathcal{F}_2(X)$ are projected in a new feature space via the computation of the Gram matrix $\mathcal{G}$. Then the distance among these feature maps and the projected style feature maps is computed. For the high-level feature maps, $\mathcal{F}_3(X)$, $\mathcal{F}_4(X)$ and $\mathcal{F}_5(X)$ the distance is computed with respect to the content feature maps obtained forwarding the content image to the VGG19 network. Once the $\mathcal{L}_{total}$ has been computed, the pixels of the image X are updated via backpropagation.

a combination of multiple style images. This approach also offers the advantage of being computationally efficient, requiring no additional training.

The main idea principle behind the proposed style transfer method is rooted in the role of early-stage layers within convolutional neural networks. These layers are primarily responsible for recognizing low-level patterns such as edges, lines, textures, and color information, as established in previous research [157]. Subsequent layers then amalgamate and enhance these features to generate more intricate representations encompassing higher-level characteristics and semantic information. Given these insights, the proposed approach prioritizes the initial feature maps extracted from the neural network as the most informative with regard to the photographic style. In contrast, the feature maps from the final stages predominantly describe the image's content. Consequently, the first set of low-level feature maps has been considered as the pivotal layers for transferring the style, while the high-level feature maps are inclined towards preserving the content. This design choice enables to produce high-quality output images that prominently exhibit the photographic style, all while faithfully preserving the image's content. Notably, one of the significant challenges in employing photographs for feature-matching style transfer methods is the potential for artifacts to manifest in the output image due to disparities between the subjects in the style and content images.

Given a set of n style images $S_i \in \mathbb{R}^{H\times W\times 3}$ and a content image $C \in \mathbb{R}^{H\times W\times 3}$, the method aims to generate an output image $X \in \mathbb{R}^{H\times W\times 3}$ that retains the same content as C but is rendered in the photographic style represented by $S_1,\dots,S_n$. To achieve this, a pretrained convolutional neural network ϕ designed for image recognition is employed. The input to this neural network is both the content image C and the n style images S_i, resulting in L sets of feature maps for each image. Here, L denotes the number of convolutional blocks. The feature maps associated with the style images and the content image are denoted as $\mathcal{F}_l(S_i)$ and $\mathcal{F}_l(C)$, respectively, with l ranging over $1,2,\dots,L$. Similarly, a randomly initialized image X is passed through ϕ, yielding $\mathcal{F}_l(X)$ for all feature maps, each with dimensions of $H_l \times W_l \times C_l$.

As previously discussed in this section, the initial m feature maps extracted play a crucial role in style transfer, while the remaining $L-m$ primarily focus on preserving the image's semantic content. Given the goal of transferring a more general photographic style, as opposed to an image-specific style, the style feature map is selected from the n available options for each of the m levels. To be more specific, the one with the closest $\mathcal{L}_2$ distance is selected as follows:

$$\mathcal{F}_l^\star = \underset{\mathcal{F}_l(S_i),\ i=\{1,\dots,n\}}{\operatorname{argmin}} \|\mathcal{F}_l(C) - \mathcal{F}_l(S_i)\|^2. \tag{4.1}$$

Then, the style maps $\mathcal{F}_1^\star,\dots,\mathcal{F}_m^\star$ are aligned with their corresponding feature maps from X, denoted as $\mathcal{F}_1(X),\dots,\mathcal{F}_m(X)$. The remaining maps $\mathcal{F}_{m+1}(X),\dots,\mathcal{F}_L(X)$ are matched to the content feature maps at the same levels, $\mathcal{F}_{m+1}(C),\dots,\mathcal{F}_L(C)$. Given that the goal was to transfer a general photographic style using multiple images from the same photographer, it's important to note that the selected style feature maps $\mathcal{F}_1^\star,\dots,\mathcal{F}_m^\star$ can originate from m distinct style images.

For the low-level feature maps, the matching is executed by transforming them into a new feature space using the Gram matrix $\mathcal{G} \in \mathbb{R}^{C_l\times C_l}$, where C_l represents the number of channels in the feature map at layer l. When considering the image X and its corresponding feature map $\mathcal{F}_l(X)$ extracted at layer l, the associated Gram matrix is defined as follows:

$$\mathcal{G}_l(\mathcal{F}_l(X)) = \frac{1}{H_l W_l} \sum_{i=1}^{H_l} \sum_{j=1}^{W_l} \mathcal{F}_l(X)_{ij} \cdot \mathcal{F}_l(X)_{ij}^T. \tag{4.2}$$

This matrix captures the relationships among the information contained in each channel of the feature maps within a new feature space, enhancing the expression of image style, as introduced in [59]. The matching process involves minimizing the style loss function $\mathcal{L}_{\text{sty}}$:

$$\begin{aligned}\mathcal{L}_{\text{sty}}^{(l)} &= \frac{1}{C_l^2}\sum_{i,j=1}^{C_l}(\mathcal{G}_l(\mathcal{F}_l^\star)_{ij} - \mathcal{G}_l(\mathcal{F}_l(X))_{ij})^2,\\ \mathcal{L}_{\text{sty}} &= \frac{1}{m}\sum_{l=1}^{m}\mathcal{L}_{\text{sty}}^{(l)}.\end{aligned} \tag{4.3}$$

For the remaining $L - m$ feature maps, the content loss function was computed as:

$$\begin{aligned}\mathcal{L}_{\text{cnt}}^{(l)} &= \frac{1}{H_l W_l C_l}\sum_{i=1}^{H_l}\sum_{j=1}^{W_l}\sum_{k=1}^{C_l}(\mathcal{F}_l(C)_{ijk} - \mathcal{F}_l(X)_{ijk})^2,\\ \mathcal{L}_{\text{cnt}} &= \frac{1}{L-m}\sum_{l=m+1}^{L}\mathcal{L}_{\text{cnt}}^{(l)}.\end{aligned} \tag{4.4}$$

The final loss function is computed as

$$\mathcal{L}_{\text{total}} = \lambda_{\text{cnt}}\mathcal{L}_{\text{cnt}} + \lambda_{\text{sty}}\mathcal{L}_{\text{sty}}. \tag{4.5}$$

with λ_{cnt} and λ_{sty} serving as weighting factors for the content and style loss, which are necessary to balance the contributions of these two loss components. The objective of the loss function minimization is twofold. First, it aims to reduce the disparity between the style-feature representation of the image X and the feature maps associated with the style. Second, it strives to minimize the dissimilarity between the high-level content representation of the content-feature maps and the high-level feature maps of the image X. The image X is generated through the iterative minimization of the total loss function $\mathcal{L}_{\text{total}}$. To achieve this, the L-BFGS optimization algorithm [203] was employed. The refinement process continues until the optimizer converges or a fixed number of steps S is reached. Figure 4.4 provides a schematic representation of the approach. The pseudocode for the proposed neural style transfer method is outlined in Algorithm 2. It comprises two primary functions: the selection function and the transfer function. The selection function identifies the m candidate style feature maps, denoted as $\mathcal{F}_l^\star$. These maps are subsequently concatenated with the remaining $L - m$ content feature maps, and the resulting list of feature maps is passed to the transfer function. The transfer function takes the input image X and iteratively enhances it by aligning the style and content feature maps until convergence or until the maximum number of iterations, here set to $S = 3000$, is attained. The outcome of this function is the refined image X, imbued with the style from m images.

Algorithm 2 PyTorch-style pseudocode for our neural style transfer algorithm.

```
function SELECTION($S$,$C$,$\phi$,$L$,$m$):                ▷ $S$: $n \times 3 \times H \times W$; $C$: $1 \times 3 \times H \times W$
    $\mathcal{F}_1(S),\dots,\mathcal{F}_L(S) = \phi(S)$                 ▷ $\mathcal{F}_i(S)$: $n \times C_l \times H_l \times W_l$
    $\mathcal{F}_1(C),\dots,\mathcal{F}_L(C) = \phi(C)$                 ▷ $\mathcal{F}_i(C)$: $1 \times C_l \times H_l \times W_l$
    sel=[]
    for j in range($m$) do
        idx=argmin($\|\mathcal{F}_j(C) - \mathcal{F}_j(S)\|^2$)
        sel.append($\mathcal{F}_j(S)$[idx])
    end for
    sel+=[$\mathcal{F}_j(C)$ for j in range($m+1$,$L$)]
    return sel
end function

function TRANSFER ($X$,$C$,$S$,$\phi$,$m$,$L$,OPTM,$\lambda_{sty}$,$\lambda_{cnt}$):      ▷ $X$:1x3xHxW
    sel=selection($S$,$C$,$\phi$,$L$,$m$)
    $S = 0$
    while !=converged or $S < 3000$:  do
        optimizer.zero_grad()
        $\mathcal{F}_1(X),\dots,\mathcal{F}_L(X) = \phi(X)$
        sty$_{loss}$,cnt$_{loss}$= 0, 0
        for  j in range($m$) do
            sty$_{loss}$+=MSE($\mathcal{G}(\mathcal{F}_j(X))$,$\mathcal{G}$(sel[j]))
        end for
        sty$_{loss}$/=$m$
        for  j in range($m+1$,$L$) do
            cnt$_{loss}$+=MSE($\mathcal{F}_j(X)$,sel[j])
        end for
        cnt$_{loss}$/=($L - m$)
        tot$_{loss}$=$\lambda_{sty}$ sty$_{loss}$ + $\lambda_{cnt}$ cnt$_{loss}$
        tot$_{loss}$.backward()
        optimizer.step()
        $S$+=1
    end while
    return X
end function
```

Table 4.2: Comparison of state-of-the-art methods on dataset, PhotoStyle60.

Architecture	Params (M)	Batch Size	From Scratch		Fine Tuning		Transfer Learning	
			ACC@1	ACC@5	ACC@1	ACC@5	ACC@1	ACC@5
AlexNet[91]	61.10	128	25.54	55.20	50.82	76.96	44.63	72.91
VGG11[153]	132.86	128	25.80	57.52	58.56	83.75	46.43	74.98
VGG13[153]	133.05	128	25.19	56.49	58.13	84.35	47.46	76.78
VGG16[153]	138.36	128	21.07	50.73	55.46	82.20	45.92	75.49
ResNet18[71]	11.69	192	35.42	64.23	56.84	85.81	47.38	75.49
ResNet34[71]	21.80	192	35.34	65.00	59.93	83.49	45.74	74.20
ResNet50[71]	25.56	192	29.58	58.30	61.22	86.16	51.85	76.61
ResNet101[71]	44.55	192	23.13	52.19	63.46	88.22	52.45	78.07
ResNet152[71]	60.19	128	22.01	48.67	63.71	88.13	50.99	78.25
WideResNet[186]	68.88	128	35.60	65.18	65.09	87.79	46.17	73.09
SwinTransf[108]	87.77	32	29.49	60.79	**68.01**	89.08	**58.99**	**84.69**
ViT base 16[52]	86.57	128	**38.69**	**68.10**	66.64	87.96	52.11	77.30
ViT base 32[52]	88.22	128	36.20	64.23	62.34	84.01	48.32	75.41
ViT large 16[52]	304.33	64	34.05	64.57	65.61	87.70	55.96	81.26
MaxVit[161]	30.92	96	30.87	62.42	67.67	**90.37**	48.67	76.96

4.4 Experimental Results

To evaluate the distinctiveness of the styles associated with different photographers, two specific tasks have been evaluated using the dataset: (i) authorship attribution of the photographs and (ii) the accuracy of the model in recognizing the style of a particular photographer that has been transferred to a random image from the Five-K dataset. Additionally, the new multi-image style transfer method is tested.

4.4.1 Authorship Attribution.

For the authorship attribution task, a set of state-of-the-art classification models was selected to assess the dataset's quality. In this experiment, the dataset was randomly divided into training (80%) and test (20%) sets. Three different scenarios were considered: *From scratch*, models were initialized with random weights, *Transfer Learning*, only a 2-layer MLP classification head was trained using features extracted from the last layer of a frozen pretrained model and *Fine Tuning*, where pretrained models, trained on the ImageNet dataset, were fine-tuned. The performance of each model was evaluated using the Top 1 accuracy (ACC@1) and Top 5 accuracy (ACC@5) metrics. The results of the experiments are presented in Table 4.2. The *Fine Tuning* scenario yielded the best results, with the SwinTransformer model[108] achieving an ACC@1 of 68%. Furthermore, the experiment was repeated on the subset PhotoStyle10. Table 4.3 demonstrates a significant improvement in performance, particularly in the *Fine Tuning* scenario, where an ACC@1 of 97% was achieved.

Table 4.3: Comparison of state-of-the-art methods on the subset of ten photographers, PhotoStyle10.

Architecture	Params (M)	Batch Size	From Scratch		Fine Tuning	
			ACC@1	ACC@5	ACC@1	ACC@5
AlexNet[91]	61.10	128	60.38	95.25	86.38	100.00
VGG16[153]	138.36	128	57.65	93.63	90.88	100.00
ResNet152[71]	60.19	128	50.13	88.00	95.63	100.00
WideResNet[186]	68.88	128	63.50	95.88	94.00	100.00
SwinTransf[108]	87.77	32	63.88	96.25	**97.50**	100.00
ViT large 16[52]	304.33	64	**69.88**	95.88	93.75	100.00
MaxVit[161]	30.92	96	66.25	**96.25**	93.75	**100.00**

The results from both the *From Scratch* and *Fine Tuning* experiments on the PhotoStyle10 dataset demonstrate the ability to distinctly identify the author of a photograph, even when its content bears resemblance to photographs taken by other photographers. However, as illustrated in Table 4.2, this task becomes notably more challenging when applied to the entire dataset. The overlap in photograph content can lead to a situation where a larger number of professional photographers may employ similar styles, making authorship attribution more intricate.

4.4.2 Style-Transfer.

The aim of the style transfer task is to assess the effectiveness of transferring photographers' styles to images that are not included in the dataset. For this experiment, content images from the Five-K dataset have been chosen and employed photographs from the dataset as style images. The capacity of various state-of-the-art methods was evaluated to carry over the styles of the photographers. The compared style transfer methods include: (i) Gatys et al. [59] (ii) Li et al. [97] (iii) Wang et al. [167] (iv) Hong et al. [73] (v) Li et al. [99] (vi) Luan et al. [115] (vii) Zhu et al. [204] (viii) Li et al. [98].

Methods (i)-(v) are feature matching approaches that do not necessitate specific style training. In these cases, a random set of 100 images from the Five-K dataset to serve as content images, each paired with the most akin style photograph from each of the 60 photographers in the dataset, have been carefully selected. To achieve this, a content image C from the Five-K dataset and n style images S_i from a given photographer have been employed. All of these images were processed through a pretrained ResNet152, extracting their last-layer features. Following this, the style image S have been identified as the one that exhibited the highest similarity to the provided content image C, based on the cosine similarity between their feature vectors. Consequently, for each of these five methods, collection of 100 images for each of the

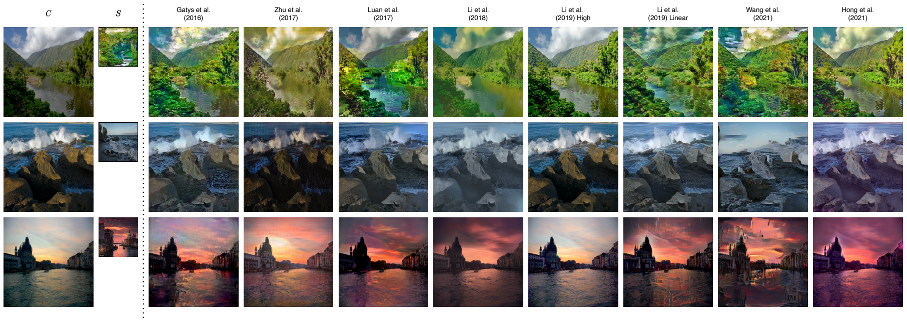


Figure 4.5: Comparison of the results obtained by using eight neural style transfer methods. Each of these methods is used to transfer the style of a photograph from the PhotoStyle60 to the most perceptual similar image from the Five-K dataset.

60 photographers have been generated. These images have been then tested using the top-performing method from the author attribution experiment, the SwinTransformer. The objective was to classify the photographer's style, thus evaluating the effectiveness of the style transfer methods. The results, as demonstrated in Table 4.4, show that the best method is the one proposed by Li et al.'s model [99] as the most proficient style transfer model. Notably, due to the substantial time requirements for training 60 models, one for each style, methods (vi)-(viii) were intentionally omitted from this final experiment. However, Figure 4.5 shows the outcomes obtained by testing each of the eight style transfer algorithms on the dataset. An examination of the figure reinforces the conclusion drawn from Table 4.4, affirming the effectiveness of Li et al.'s method [99]. This approach adeptly transfers the photographic style of the input image S while meticulously preserving the content, as evident from the crisp presentation of style nuances in the output images. In contrast, although the other methods do accomplish style transfer, they tend to introduce visible artifacts in the output images. These artifacts can significantly compromise image quality and alter patterns that should ideally have been preserved throughout the style transfer process.

4.4.3 Multi-Image Style Transfer

In style transfer method proposed above, a VGG19 neural architecture ϕ have been employed (which consists of five convolutional blocks separated by pooling layers). The first two blocks encompass two convolutional layers each, while the remaining three are composed by a sequence of four convolutional layers. An Extensive analysis of the method has led to identify the optimal configuration for effectively transferring

Table 4.4: Performances of the Swin Transformer trained on PhotoStyle60 and tested to recognize the author of the images obtained using five methods from the state of the art by applying the corresponding style transfer algorithm to 100 images from the Five-K dataset.

Method	ACC@1	ACC@5	Prec@1	Recall@1
Gatys et al. [59]	18.74	41.03	30.81	18.83
Li et al. [97]	6.92	22.75	9.47	6.92
Wang et al. [167]	6.48	23.55	11.01	6.48
Hong et al. [73]	5.62	18.90	8.32	5.62
Li et al. [99]	**62.03**	**85.82**	**68.13**	**62.03**

photographic style while simultaneously preserving image content. This configuration involves setting $m = 2$, designating the first two feature layers for style and reserving the remaining three for content preservation. In Section 4.4.3, a detailed examination of the impact of different choices of m on the results is proposed.

Qualitative results

In this section, the qualitative results of the style transfer method are reported. A different combinations of hyperparameters that modulate the contributions of different losses are tested to validate the effectiveness of the method. The most successful configuration, resulting in the generation of the most realistic and natural images, includes setting $\lambda_{cnt} = 10^5$ and $\lambda_{\text{sty}} = 45$. The algorithm have run for $S = 3000$ iterations. The outcomes are displayed in Figure 4.6. From a qualitative perspective, it is evident that the transferred style is prominently visible in most of the content images, while maintaining image quality without severe artifacts. For the second and third photographers in the figure, although the transferred style is apparent in the content images, some of them may feature subjects that are not typically associated with a particular photographer, resulting in images of varying aesthetic appeal. In contrast, for the first and last photographers, the style transfer is remarkably evident. For instance, the last photographer exhibits a distinct preference for a gloomy atmosphere, which is consistently reflected in all the content images.

User Study

The objective of this user study was to evaluate the quality of the model results under different aspects: (i) Quality: determine if the obtained image is realistic in shapes, colors and content with respect to the original image (ii) Style: assess if the style of

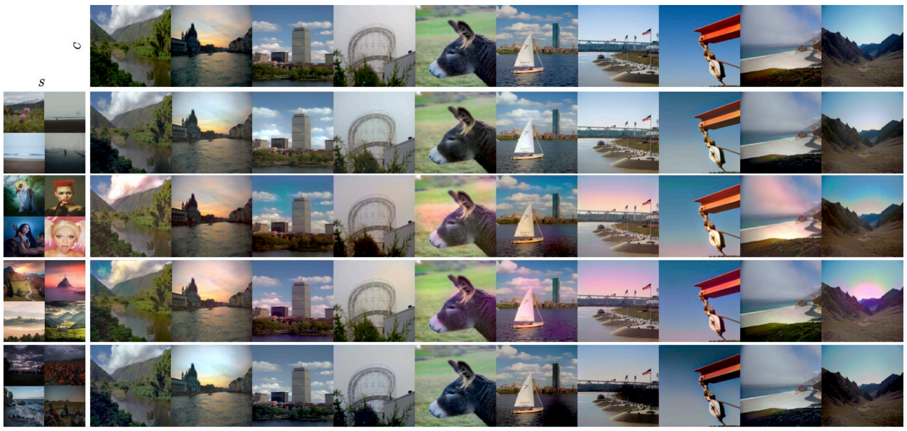


Figure 4.6: Results of the new style transfer method on different content images and styles. The original content images are provided in the first line, while the style is represented on the left by the four most representative photographs of four sample photographers from the dataset.

the photographer has been transferred to the content image. (iii) Artifacts: verify if the obtained image has visible artifacts. In this last case a lower score means a better image presenting a reduced or null number of artifacts.

In this study, the novel proposed model and the methods outlined in Section 4.4.2, representing the state-of-the-art techniques for the style-transfer task, were included. To conduct these experiments, 25 individuals who had no involvement in shaping the model have been asked to perform an user study. Participants were presented with a test in which 20 different content images were displayed both before and after style transfer from the proposed dataset. The style was presented to users as a collection of the four most representative photographs from a specific photographer and was transferred to each of the 20 selected images using all five style transfer methods listed in Table 4.5. This resulted in a total of 100 images for evaluation. To prevent users from recognizing patterns between the images, the images were presented in random order. Participants were asked to rate each resulting image on a scale from 1 to 5 for each of the categories listed previously (quality, style, artifacts). As indicated by the user study results in Table 4.5, the proposed method outperforms the other methods significantly in terms of faithfully reproducing the content of the original image while maintaining a realistic appearance. Additionally, it excels in style transfer and prevents the generation of artifacts. In comparison, other methods like [97, 99] demonstrated similar results in terms of quality and transferred style, with the former exhibiting a higher level of artifacts. Finally, the method proposed in [167] received the lowest scores in the user study. Additional details and examples of screenshots of the user study are available in the supplementary materials.

Table 4.5: Results of the User Study

Method	Quality↑	Transferred Style↑	Artifacts↓
Ours	**3.89**	**3.09**	**2.09**
Li et al. [97]	2.85	2.76	3.15
Wang et al. [167]	1.37	1.85	4.62
Hong et al. [73]	2.52	2.46	3.21
Li et al. [99]	2.98	2.82	2.31

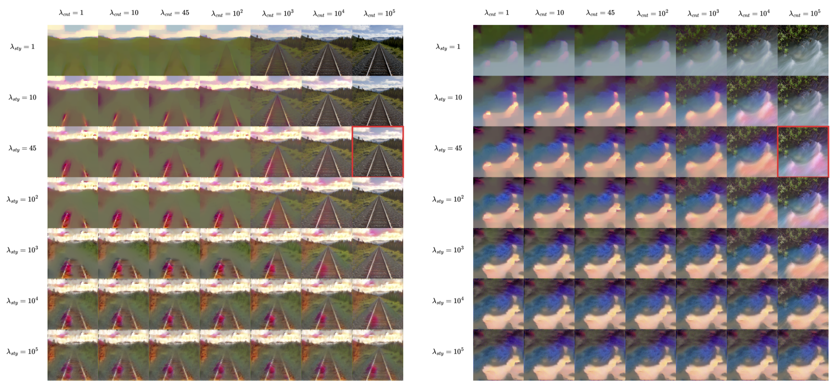


Figure 4.7: Result of the grid search performed on the hyperparameters considered in the proposed method, λ_{sty} and λ_{cnt}.

Ablation Study

Finally, an ablation study was conducted to assess how the choice of feature maps to be transferred influences the resulting image quality in the proposed method. The outcomes of these experiments are illustrated in Figure 4.8. As reported in the figure, increasing the number of $\mathcal{F}_l^{\star}$ feature maps, progressing from $m = 1$ in the second row to $m = 4$ in the last, enhances the style transfer process but leads to visible image artifacts due to the significant dissimilarity between the content and style image subjects. Notably, it is important to observe that the best qualitative results are

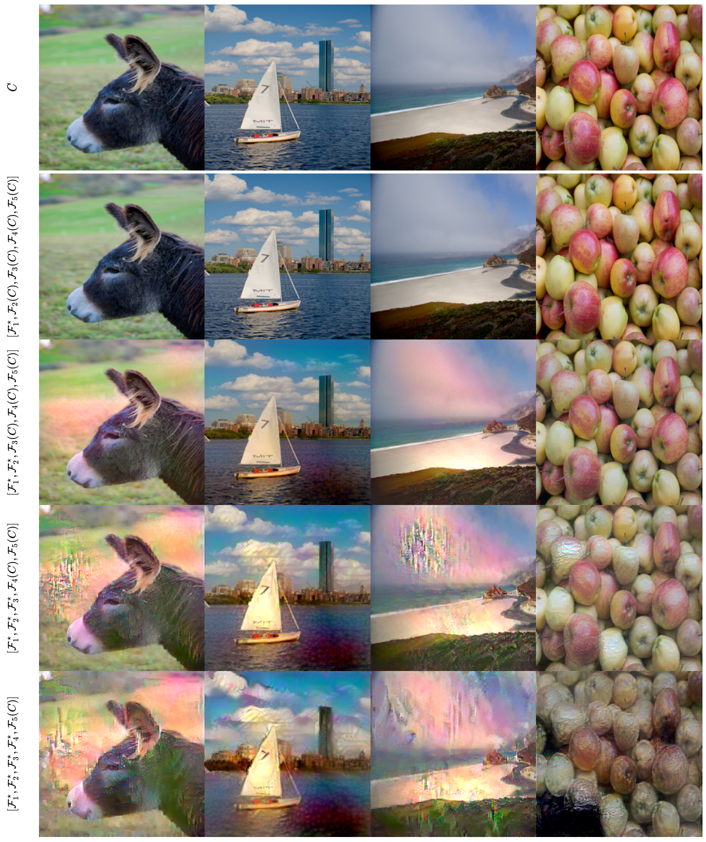

C
$[\mathcal{F}_1^\star, \mathcal{F}_2(C), \mathcal{F}_3(C), \mathcal{F}_4(C), \mathcal{F}_5(C)]$
$[\mathcal{F}_1^\star, \mathcal{F}_2^\star, \mathcal{F}_3(C), \mathcal{F}_4(C), \mathcal{F}_5(C)]$
$[\mathcal{F}_1^\star, \mathcal{F}_2^\star, \mathcal{F}_3^\star, \mathcal{F}_4(C), \mathcal{F}_5(C)]$
$[\mathcal{F}_1^\star, \mathcal{F}_2^\star, \mathcal{F}_3^\star, \mathcal{F}_4^\star, \mathcal{F}_5(C)]$

Figure 4.8: Comparison between the different combinations of $\mathcal{F}_l^\star$ and $\mathcal{F}_l(C)$ feature maps in the proposed method.

achieved by considering only the first two feature maps (i.e., $m = 2$) as representatives of the style. This outcome aligns with the underlying principle of the proposed method. Specifically, as the number of style feature maps increases, the image's content degrades in favor of the style. Consequently, maintaining the initial layers for the style and the later layers for the content yields the most favorable outcomes for transferring photographic style while preserving the image's semantic content.

Hyperparameter Analysis

Finally, a study was also carried out to investigate the influence of the hyperparameters λ_{sty} and λ_{cnt} on the quality of the generated images. The outcomes of this hyperparameter exploration, conducted on two images from the Five-K dataset with style transferred from two photographers in the dataset, are depicted in Figure 4.7. The values of λ_{sty} and λ_{cnt} were systematically varied across the range $1, 10, 45, 10^2, 10^3, 10^4, 10^5$.

The results illustrate that when the style weight significantly outweighs the content weight, the image quality is compromised, manifesting a pronounced presence of artifacts and blurriness. With an increase in the content weight, the semantic content of the image is progressively preserved, and when it becomes exceedingly high, the style ceases to be discernible in the resulting image. As depicted in the figure, the chosen hyperparameters, highlighted in red, yield images that effectively preserve content without introducing artifacts, all while retaining a clearly visible style.

4.5 Summary

This chapter introduced PhotoStyle60, a meticulously curated dataset comprising 5708 images contributed by 60 distinct photographers. The proposed dataset was thoughtfully assembled to encompass a wide spectrum of photographic styles, marking it as the most extensive collection in this regard. The results of an exhaustive series of experiments, addressing two main research domains, authorship attribution and photographic style transfer, have been presented.

Finally, an innovative multi-image style transfer technique that outperformed alternative deep learning-based neural style transfer algorithms, have been presented. It shown high performance in preserving the image's semantic content, transferring the style faithfully, and mitigating the occurrence of undesirable artifacts in the resulting image.

4.6 Additional Materials

4.6.1 User Study Details

Here, the details of the web application utilized for conducting the user study are reported. In Figure 4.12, a selection of screenshot examples that were presented to the study participants are reported. These screenshots are organized in a row with three columns: (i) The first column displays the content image, (ii) The second column exhibits a grid containing four examples of style images intended for transfer onto the content image, (iii) The final result achieved through the style transfer algorithm.

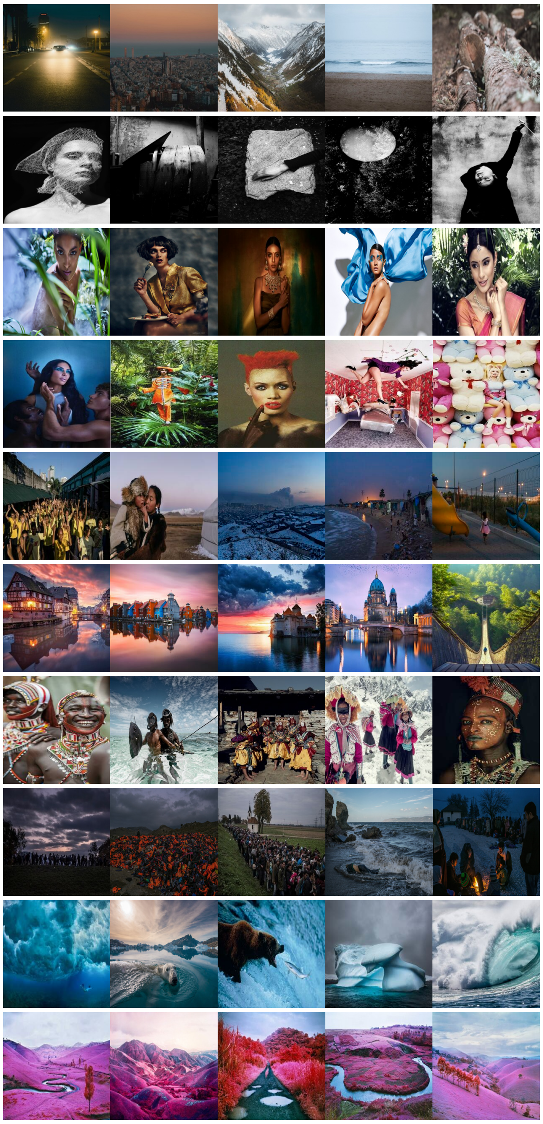


Figure 4.9: Example of images from the PhotoStyle10. Each group of 5 images belong to one of the ten photographer in the subset.

The image dimensions, especially for the content and result images, were thoughtfully chosen to enable users to discern even minute details and potential artifacts. Below each image, three boxes prompt the user to assign a rating from 1 to 5 based on three metrics: the overall quality of the result, the fidelity of the style match to the target, and the presence of any artifacts. Adjacent to each box, a description guides the user on how to interpret the rating scale for these three metrics. At the top of the web application, a progress bar indicates the number of images remaining for evaluation. In proximity to the progress bar, two buttons are provided: the first facilitates navigation to the next comparison, while the second opens a tutorial elucidating the evaluation process in detail, supported by multiple examples.

4.6.2 Additional PhotoStyle60 Images

In this section, additional images from the proposed dataset and its reduced version are reported. In Figure 4.9, a selection of images from the reduced version of the dataset, the PhotoStyle10 dataset, comprising five images per photographer that highlight their distinctive photographic styles, is shown. The photographer selection criteria focused on style diversity rather than content, resulting in a wide array of styles among the ten featured photographers. As demonstrated in the results section, this approach allowed to illustrate the potential of identifying photographers based on their unique photographic styles.

Finally, high-resolution examples from the dataset are reported in Figure 4.10 and Figure 4.11.

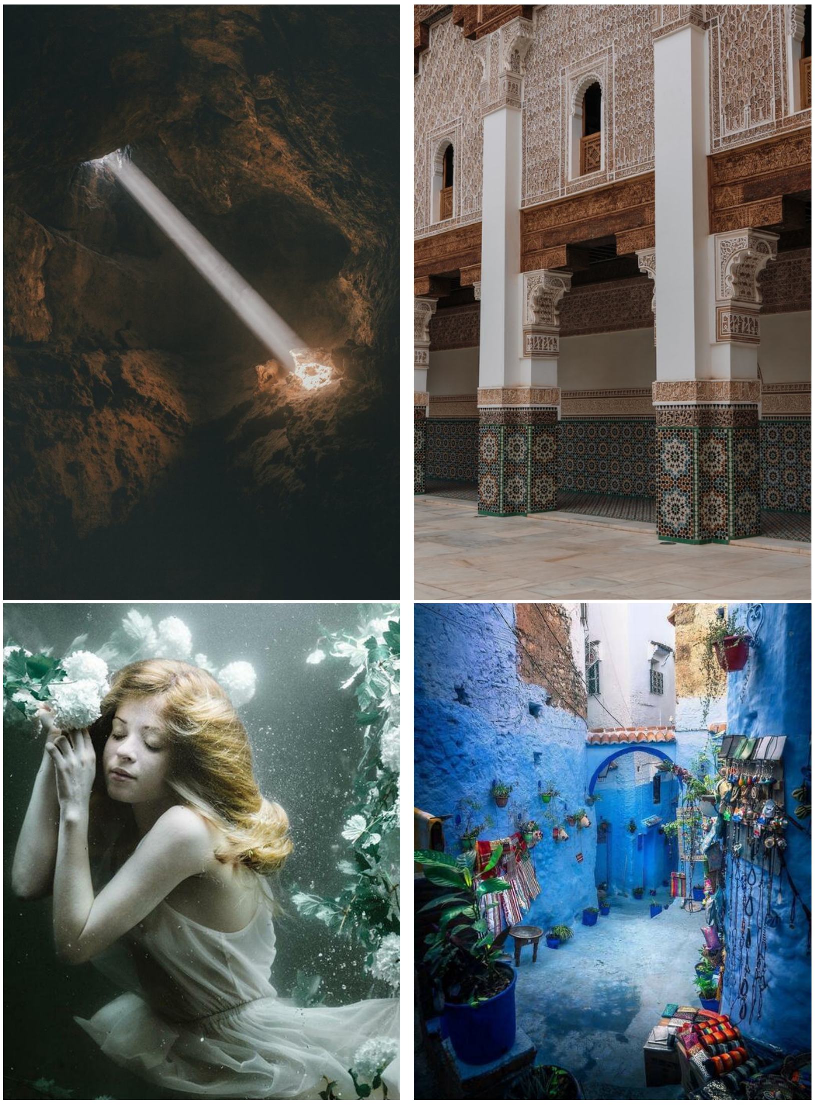


Figure 4.10: Example of 4 high resolution images from PhotoStyle60

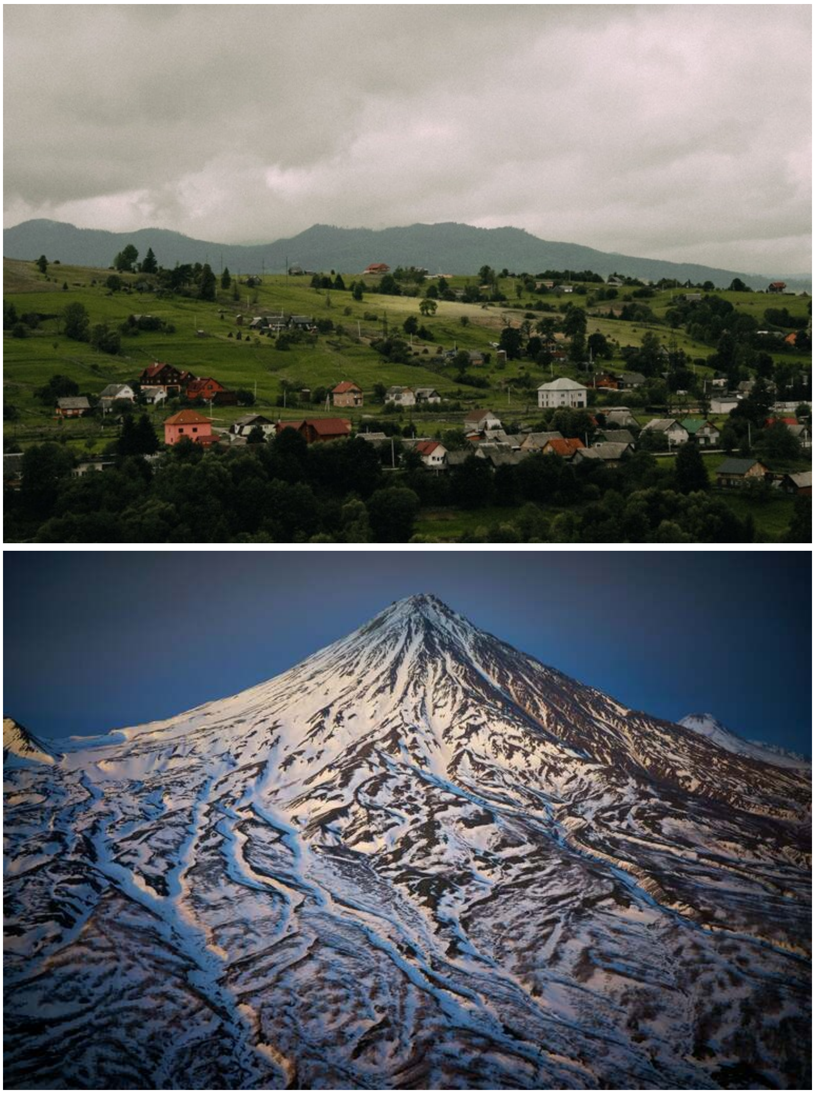
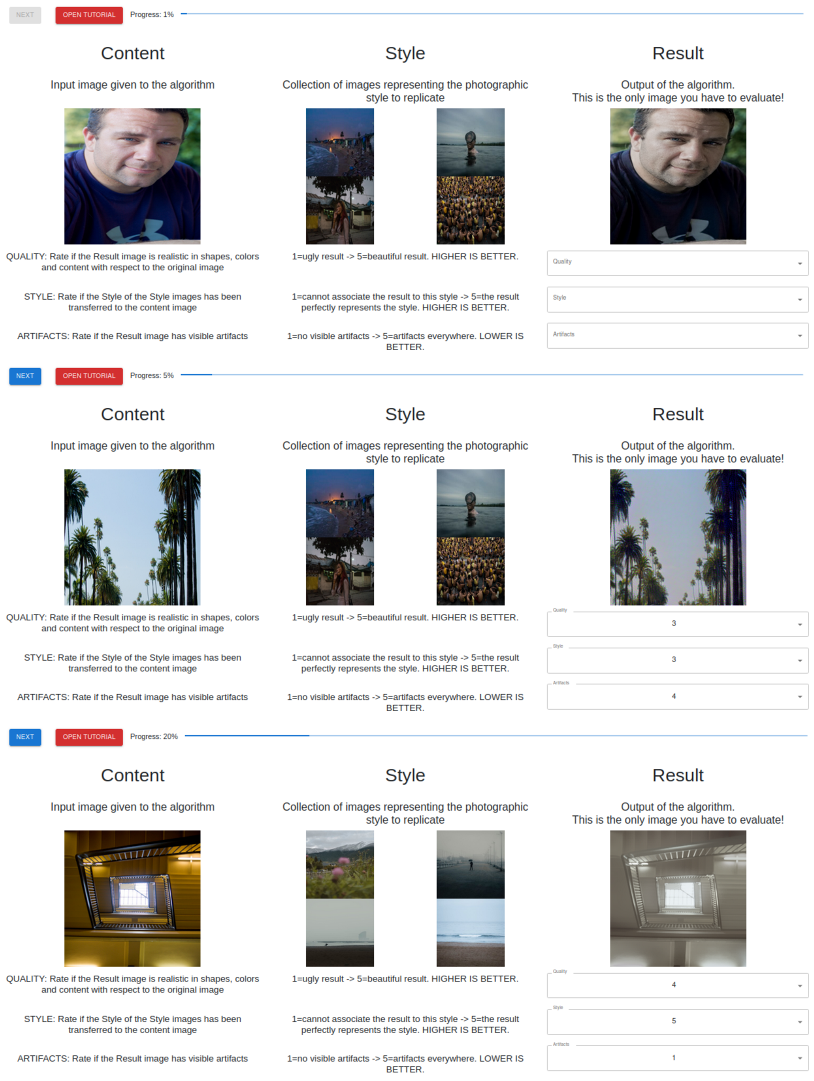


Figure 4.11: Example of 2 high resolution images from PhotoStyle60



Content

Input image given to the algorithm

Style

Collection of images representing the photographic style to replicate

Result

Output of the algorithm.
This is the only image you have to evaluate!

QUALITY: Rate if the Result image is realistic in shapes, colors and content with respect to the original image | 1=ugly result -> 5=beautiful result. HIGHER IS BETTER. | Quality

STYLE: Rate if the Style of the Style images has been transferred to the content image | 1=cannot associate the result to this style -> 5=the result perfectly represents the style. HIGHER IS BETTER. | Style

ARTIFACTS: Rate if the Result image has visible artifacts | 1=no visible artifacts -> 5=artifacts everywhere. LOWER IS BETTER. | Artifacts

Content

Input image given to the algorithm

Style

Collection of images representing the photographic style to replicate

Result

Output of the algorithm.
This is the only image you have to evaluate!

QUALITY: Rate if the Result image is realistic in shapes, colors and content with respect to the original image | 1=ugly result -> 5=beautiful result. HIGHER IS BETTER. | Quality: 3

STYLE: Rate if the Style of the Style images has been transferred to the content image | 1=cannot associate the result to this style -> 5=the result perfectly represents the style. HIGHER IS BETTER. | Style: 3

ARTIFACTS: Rate if the Result image has visible artifacts | 1=no visible artifacts -> 5=artifacts everywhere. LOWER IS BETTER. | Artifacts: 4

Content

Input image given to the algorithm

Style

Collection of images representing the photographic style to replicate

Result

Output of the algorithm.
This is the only image you have to evaluate!

QUALITY: Rate if the Result image is realistic in shapes, colors and content with respect to the original image | 1=ugly result -> 5=beautiful result. HIGHER IS BETTER. | Quality: 4

STYLE: Rate if the Style of the Style images has been transferred to the content image | 1=cannot associate the result to this style -> 5=the result perfectly represents the style. HIGHER IS BETTER. | Style: 5

ARTIFACTS: Rate if the Result image has visible artifacts | 1=no visible artifacts -> 5=artifacts everywhere. LOWER IS BETTER. | Artifacts: 1

Figure 4.12: Example of screens presented to the user during the user study.

5 Color Constancy through Offset Equivariant Neural Networks

Recent studies have concentrated on the concept of equivariance in neural networks [88, 134, 182]. In essence, a network, or one of its components, is considered equivariant concerning a group of transformations if applying one of these transformations to the input results in a predictable change in the output. For instance, convolutions exhibit equivariance with respect to translations in the plane, meaning that a translation in the input corresponds to an equivalent translation in the output.

Equivariance is an relevant concept in neural networks theory since it enables the imposition of an inductive bias [123]. For example, in convolutional networks, equivariance allows for the exploitation of the knowledge that objects and patterns may manifest in various locations within an image [90, 94]. In the field of graph neural networks, it makes the model independent from the order used to label the vertices in graphs [122]. Furthermore, in 3D processing, equivariance extends the applicability of convolutions to 3D meshes [44].

In the field of computer vision, various models of convolutional networks have been introduced to achieve equivariance concerning geometric transformations such as translations, rotations, scalings, and reflections [39, 40]. This advancement has facilitated remarkable achievements in various applications [36, 45, 46, 47, 111, 126, 175, 176, 177, 178, 179].

5.1 Introduction

In this chapter, the problem of achieving equivariance with respect to*photometric* transformations, such as those caused by variations in the lighting conditions or in the acquisition device, is analyzed, For instance, a neural network, equivariant with respect to photometric transformations, would obtain accurate predictions even when the images during inference are acquired with a device that is different to the one producing training images, or when facing a new unforeseen kind of illumination. These kinds of variations are often dealt with the application of normalization techniques (e.g. color balancing) at the cost of discarding potentially useful information [11].

A novel framework for designing neural networks that exhibit equivariance to a specific group of photometric transformations, is presented. Specifically, the transformations under consideration involve adding a uniform offset to the input data. In the case of grayscale images, this offset corresponds to variations in brightness, while for color images (when using an appropriate color space), it corresponds to alterations in the illuminant color of the scene.

Figure 5.1 illustrates the desired outcome when applying a neural network trained for image inpainting. In this scenario, a change in the illuminant's color in the input image should lead the network to produce an output that is identical to the original output but modified by the same change in illuminant color. The proposed architecture, tailored specifically to address color constancy challenges, aims to ensure robustness against rare and previously unseen illuminants. It can be applied and tested in various contexts, including image recognition to achieve invariance with respect to global lighting conditions, or automatic image editing tasks like inpainting [68], compositing [160], and more, allowing these methods to adapt automatically to the illumination present in the input image.

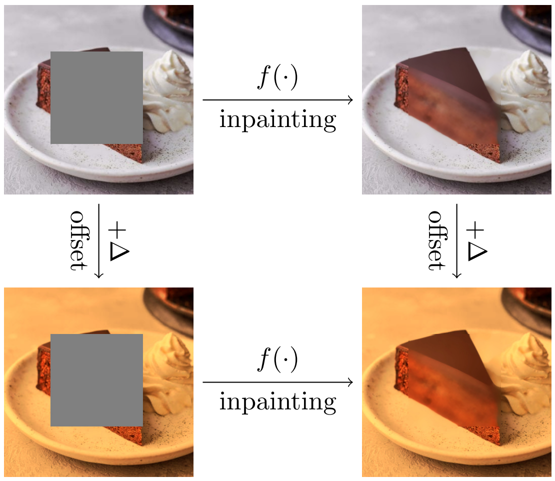


Figure 5.1: Representation of offset equivariance in the case of inpainting of color images. Under the hypothesis that f is an offset equivariant neural network implementing image inpainting, a change of illuminant and the application of f can occur in any order without affecting the result.

Section 5.2 introduces the concept of offset equivariance along with its inherent properties. Additionally, it presents a methodology for converting conventional neural networks into equivariant ones. In Section 5.3, presents how offset equivariance can be leveraged to attain equivariance in the context of changes in illumination. Section 5.4 presents the results of the application of such networks to the color constancy problem. The section ends presenting two additional experiments performed on image recognition and automatic image editing task. Finally, in Section 5.5, outcomes of these experiments and outline potential directions for future research are discussed.

5.2 Offset equivariant networks

This section introduces the concept of *offset equivariant* functions, presents their fundamental characteristics, and employs them in the formulation of layers within offset equivariant neural networks.

In essence, an *offset equivariant* network possesses the capability to preserve the overall "level" of the input signal. To illustrate, if every element of the input vector x uniformly increases by Δ, the components of the output likewise increase by the same magnitude. Formally, this property is encapsulated by the following equation, which must hold true for all $\mathrm{x} \in \mathbb{R}^m$ and all $\Delta \in \mathbb{R}$:

$$f(\mathrm{x} + 1_m\Delta) = f(\mathrm{x}) + 1_n\Delta, \tag{5.1}$$

where $f : \mathbb{R}^m \to \mathbb{R}^n$ represents a neural network (or one of its constituent layers). Moreover, 1_m and 1_n refer to vectors consisting of m and n unitary components, respectively. For instance, when x represents a grayscale image, uniformly elevating the brightness of all pixels by Δ results in an equivalent increase in the output.

In the case of color images, enforcing Equation (5.1) is not particularly beneficial because photometric transformations can impact the color channels differently. When dealing with color images, input and output features are organized into separate groups, typically one group for each color channel. In this context, offset equivariance is defined as the property of preserving uniform variations of features within the same group. This can be expressed as follows:

$$f(\mathrm{x} + G_m\boldsymbol{\Delta}) = f(\mathrm{x}) + G_n\boldsymbol{\Delta}, \tag{5.2}$$

$\boldsymbol{\Delta} \in \mathbb{R}^g$ represents a vector of offsets, with one offset for each group. Additionally, G_m is an $m \times g$ matrix that signifies the assignment of the m input features to the g

groups:

$$(G_m)_{ij} = \begin{cases} 1 & \text{if the } i\text{-th feature is in the } j\text{-th group,} \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

Likewise, the $n \times g$ matrix G_n represents how the n output features are assigned to the g groups. Please note that the definition of equivariance relies on these assignment matrices. For the sake of simplicity, throughout the rest of this chapter, it is assumed that compatible assignments are consistently used when dealing with multiple equivariant functions.

An important characteristic of offset equivariant functions is their closure under functional composition. If both $f_1 : \mathbb{R}^m \to \mathbb{R}^n$ and $f_2 : \mathbb{R}^p \to \mathbb{R}^m$ are offset equivariant functions, their composition $h = f_1 \circ f_2$ also exhibits offset equivariance. This property is a direct consequence of Equation (5.2):

$$\begin{aligned} h(\mathrm{x} + G_p\mathbf{\Delta}) &= f_1(f_2(\mathrm{x} + G_p\mathbf{\Delta})) = f_1(f_2(\mathrm{x}) + G_m\mathbf{\Delta}) \\ &= f_1(f_2(\mathrm{x})) + G_n\mathbf{\Delta} = h(\mathrm{x}) + G_n\mathbf{\Delta}. \end{aligned} \tag{5.4}$$

While linear combinations, in general, do not preserve offset equivariance, affine linear combinations accomplish precisely that. Given the equivariant functions f_1, f_2, ..., f_k and the coefficients $\alpha_1, \alpha_2, \ldots, \alpha_k$, the function h defined as:

$$h(\mathrm{x}) = \sum_{i=1}^{k} \alpha_i f_i(\mathrm{x}), \tag{5.5}$$

is offset equivariant if and only if $\sum_{i=1}^{k} \alpha_i = 1$. In fact:

$$\begin{aligned} h(\mathrm{x} + G_m\mathbf{\Delta}) &= \sum_{i=1}^{k} \alpha_i f_i(\mathrm{x} + G_m\mathbf{\Delta}) \\ &= \sum_{i=1}^{k} \alpha_i \left(f_i(\mathrm{x}) + G_n\mathbf{\Delta}\right) \\ &= \left(\sum_{i=1}^{k} \alpha_i f_i(\mathrm{x})\right) + \left(\sum_{i=1}^{k} \alpha_i\right) G_n\mathbf{\Delta} \\ &= h(\mathrm{x}) + G_n\mathbf{\Delta}. \end{aligned} \tag{5.6}$$

These properties suggest a potential strategy for designing offset equivariant networks, involving defining them as compositions and combinations of the offset equivariant layers detailed in the following sections.

5.2.1 Linear layers

Most neural architectures incorporate one or more linear layers, which are prevalent in various network types, including multi-layer perceptrons [139], convolutional neural networks [90], graph neural networks [145], attention mechanisms [164], and many more. Linear layers are characterized by a weight matrix W of size $n \times m$ and a bias vector b with dimension n:

$$f(\mathrm{x}) = W\mathrm{x} + \mathrm{b}. \tag{5.7}$$

The introduction of an offset leads to:

$$f(\mathrm{x} + G_m\mathbf{\Delta}) = W\mathrm{x} + WG_m\mathbf{\Delta} + \mathrm{b} = f(\mathrm{x}) + WG_m\mathbf{\Delta}, \tag{5.8}$$

which, combined with Equation (5.2), gives us

$$WG_m\mathbf{\Delta} = G_n\mathbf{\Delta}. \tag{5.9}$$

Since the above must hold true independently on $\mathbf{\Delta}$, offset equivariance for linear layers with the linear constraint can be summarized as:

$$WG_m = G_n, \tag{5.10}$$

which is feasible if and only if all the groups assigned features by G_n also have at least one feature assigned by G_m. It iss worth noting that the equivariance of f does not depend on the bias vector b, which can remain as a free parameter. In practice, W could be reparametrized to satisfy the constraint (5.10). Alternatively, it is possible to work with a standard linear layer and project W onto the constraint after each backpropagation iteration. The projected weights $\hat{W}$ can be computed as follows:

$$\hat{W} = W - (WG_m - G_n)G_m^+, \tag{5.11}$$

where G_m^+ is the pseudoinverse of G_m. Due to the regular structure of G_m, its pseudoinverse has a simple form, given by $G_m^+ = N^{-1}G_m^T$, where N is the diagonal matrix with elements representing the number of features in each group. The weights of the linear layer are updated by alternating between the standard backpropagation steps and the projection defined in (5.11).

5.2.2 Convolutional layers

Since convolutions are linear operators, they can be treated in the same way as discussed in the previous section. Here, in particular, the two-dimensional case is considered, but the generalization to other numbers of dimensions is straightforward. Given a set of coefficients $\mathcal{W} \in \mathbb{R}^{k\times k\times n\times m}$, organized as a $k\times k$ array of linear operators $\mathcal{W}_{ij} \in \mathbb{R}^{n\times m}$, the convolution for the input image x is denoted as:

$$f(\mathrm{x}) = \mathcal{W} \star \mathrm{x}. \tag{5.12}$$

The pixel value at location (i, j) is the m-dimensional vector x_{ij} and the n-dimensional output value at location (i, j) is defined as:

$$f(\mathrm{x})_{ij} = \sum_{s=1}^{k}\sum_{t=1}^{k} \mathcal{W}_{st}\mathrm{x}_{i+s-1,j+t-1}. \tag{5.13}$$

Let x' be the image obtained by adding the offset $G_m\boldsymbol{\Delta}$ to each pixel of x, then the corresponding output value is:

$$\begin{aligned} f(\mathrm{x}')_{ij} &= \sum_{s=1}^{k}\sum_{t=1}^{k} \mathcal{W}_{st}\mathrm{x}'_{i+s-1,j+t-1} \\ &= \sum_{s=1}^{k}\sum_{t=1}^{k} \mathcal{W}_{st}(\mathrm{x}_{i+s-1,j+t-1} + G_m\boldsymbol{\Delta}) \\ &= \sum_{s=1}^{k}\sum_{t=1}^{k} \mathcal{W}_{st}\mathrm{x}_{i+s-1,j+t-1} + \left(\sum_{s=1}^{k}\sum_{t=1}^{k} \mathcal{W}_{st}\right) G_m\boldsymbol{\Delta} \\ &= f(\mathrm{x})_{ij} + \left(\sum_{s=1}^{k}\sum_{t=1}^{k} \mathcal{W}_{st}\right) G_m\boldsymbol{\Delta}. \end{aligned} \tag{5.14}$$

Therefore, in order to achieve offset equivariance it is required to have:

$$\left(\sum_{s=1}^{k}\sum_{t=1}^{k} \mathcal{W}_{st}\right) G_m\boldsymbol{\Delta} = G_n\boldsymbol{\Delta}, \tag{5.15}$$

for all $\boldsymbol{\Delta} \in \mathbb{R}^g$, which corresponds to the following linear constraint:

$$\left(\sum_{s=1}^{k}\sum_{t=1}^{k} \mathcal{W}_{st}\right) G_m = G_n. \tag{5.16}$$

A projection $\hat{\mathcal{W}}$ satisfying (5.16) is the following:

$$\hat{\mathcal{W}}_{ij} = \frac{1}{k^2}\left(\mathcal{W}_{ij} - (\mathcal{W}_{ij}G_m - G_n)G_m^{+}\right). \tag{5.17}$$

and the weights of the convolutional layer are updated by alternating the usual back-propagation steps and the projection defined in (5.17).

To preserve offset equivariance, padding strategies that rely on the input data, such as replicating values along the border, should be used. Padding with zeros or other constant values would hinder equivariance. Strided and dilated convolutions, on the other hand, do not require any special treatment in this context.

5.2.3 Pooling layers

Standard average and max pooling are already offset equivariant operators, under the same constraint on padding imposed by convolutions.

5.2.4 Group pooling and non-linear layers

Activation functions and other non-linear layers, such as dropout [155] and batch normalization [79], are not offset equivariant. However, they can be made equivariant by incorporating auxiliary *group pooling functions*. A group pooling function is an equivariant function $\varphi : \mathbb{R}^m \to \mathbb{R}^g$ that aggregates feature values by group. In the context of the definition (5.2), an additional requirement is introduced: that G_n is a $g \times g$ identity matrix. This requirement ensures that for group pooling functions:

$$\varphi(\mathrm{x} + G_m \mathbf{\Delta}) = \varphi(\mathrm{x}) + \mathbf{\Delta}. \tag{5.18}$$

Examples of group pooling functions include taking the average over the groups, finding the maximum, and determining the minimum. Another group pooling function involves the linear combination of features, provided that the coefficients satisfy the constraint (5.10).

Group pooling functions can be employed to render offset equivariant functions from any other function. Suppose f is any function from $\mathbb{R}^m \to \mathbb{R}^n$, and let φ_1 and φ_2 be two group pooling functions. Then, the function h, defined as:

$$h(\mathrm{x}) = f(\mathrm{x} - G_m \varphi_1(\mathrm{x})) + G_n \varphi_2(\mathrm{x}), \tag{5.19}$$

is offset equivariant. The demonstration follows immediately from (5.2) and (5.18):

$$\begin{aligned} h(\mathrm{x}+G_m\boldsymbol{\Delta}) &= f(\mathrm{x}+G_m\boldsymbol{\Delta}-G_m\varphi_1(\mathrm{x}+G_m\boldsymbol{\Delta})) \\ &\quad + G_n\varphi_2(\mathrm{x}+G_m\boldsymbol{\Delta}) \\ &= f(\mathrm{x}+G_m\boldsymbol{\Delta}-G_m\varphi_1(\mathrm{x})-G_m\boldsymbol{\Delta}) \\ &\quad + G_n\varphi_2(\mathrm{x})+G_n\boldsymbol{\Delta} \\ &= f(\mathrm{x}-G_m\varphi_1(\mathrm{x}))+G_n\varphi_2(\mathrm{x})+G_n\boldsymbol{\Delta} \\ &= h(\mathrm{x})+G_n\boldsymbol{\Delta}. \end{aligned} \tag{5.20}$$

An example is the application of this technique to the ReLU activation function, one of the most widely used components of deep neural networks. Let φ be a group pooling function. Then, applying (5.19) to ReLUs with $\varphi_1=\varphi_2=\varphi$ results in:

$$\begin{aligned} h(\mathrm{x}) &= \mathrm{relu}(\mathrm{x}-G_m\varphi(\mathrm{x}))+G_m\varphi(\mathrm{x}) \\ &= \max(\mathrm{x}-G_m\varphi(\mathrm{x}),0)+G_m\varphi(\mathrm{x}) \\ &= \max(\mathrm{x},G_m\varphi(\mathrm{x})). \end{aligned} \tag{5.21}$$

In particular, when φ is a linear function this resembles a maxout network [64] (in fact the maxout activation function is naturally offset equivariant).

5.3 Illuminant equivariance

The addition of a constant offset is not the typical transformation applied to color images. Multiplication is more useful since, according to the von Kries diagonal model [165], it corresponds to the chromatic variation caused by a global change in the illuminant:

$$\begin{pmatrix} y_r \\ y_g \\ y_b \end{pmatrix} = \begin{bmatrix} I_r & 0 & 0 \\ 0 & I_g & 0 \\ 0 & 0 & I_b \end{bmatrix} \cdot \begin{pmatrix} x_r \\ x_g \\ x_b \end{pmatrix}, \tag{5.22}$$

where x_r, x_g, x_b is the color of a pixel in the linear RGB space under a canonical neutral illuminant, and where y_r, y_g, y_b is the color of the same pixel under the illuminant I_r, I_g, I_b.

A logarithmic transformation

$$x'_c = \log\frac{1}{x_c},\ \ y'_c = \log\frac{1}{y_c},\ \ I'_c = \log\frac{1}{I_c},\ \ c \in \{r, g, b\}, \tag{5.23}$$

makes the model additive:

$$\begin{pmatrix} y'_r \\ y'_g \\ y'_b \end{pmatrix} = \begin{pmatrix} I'_r \\ I'_g \\ I'_b \end{pmatrix} + \begin{pmatrix} x'_r \\ x'_g \\ x'_b \end{pmatrix}. \tag{5.24}$$

In this logarithmic RGB color space, offset equivariance corresponds to equivariance with respect to the transformation caused by a change in the illuminant.

When performing the transformation, there is one detail that must be observed: if the input component is exactly zero, then the transformed one would be $+\infty$. To prevent this, all values in the original linear space are clipped from below to a small positive value ϵ (in all the experiments, $\epsilon = 2 \times 10^{-4}$ has been set).

5.4 Applications

To evaluate the effectiveness of offset equivariant networks they have been tested on the illuminant estimation task. Then, to verify their ability of generalization to other computer vision tasks, they have been applied to other two applications: image recognition and image inpainting. For each of these applications, one or more Convolutional Neural Network architectures from existing literature have been selected and then adapted them to create their equivariant versions. Both the standard and equivariant models were trained using reference datasets. To assess their performance, the comparisons on reference test sets have been conducted. Notably, a data augmentation procedure have been introduced for assessing robustness, which involved applying a random change in the global illuminant to the test images. Importantly, this illuminant augmentation was never employed during the training phase.

To introduce equivariance into the networks, the original components were replaced with the ones outlined in Section 5.2. Specifically, three feature groups corresponding to the red, green, and blue color channels were established, and assignment matrices were employed to distribute an equal number of features into each group.

$$(G_m)_{ij} = \begin{cases} 1 & \text{if } \lceil \frac{3i}{m} \rceil = j, \\ 0 & \text{otherwise.} \end{cases} \tag{5.25}$$

This requires that the number of features is a multiple of three. In instances where modifications were necessary, the original architectures were adjusted by aligning the number of features at each layer to the nearest multiple of three.

For convolutions and linear layers, the constraint (5.10) was enforced by projecting the weights accordingly. Pooling layers remained unaltered. ReLUs and batch normalization layers were adapted by employing Equation (5.19) with average group pooling.

The next sections offer further insights and present the outcomes achieved for each task.

5.4.1 Illuminant Estimation

Illuminant estimation constitutes a crucial stage in many computational color constancy methods [62]. It involves predicting the color of the light source(s) in the scene depicted in an image. Once acquired, the estimated illuminant color can be discarded to replicate the color constancy capability of the human visual system. The outcome is a modified image in which colors appear as if the scene were captured under a standard neutral illuminant. To achieve this correction, most methods simply reverse the transformation defined by the von Kries diagonal model (5.22).

In the context of illuminant estimation, offset equivariance ensures that the prediction model behaves consistently with respect to changes in illumination. Consider two images, x and x', representing the same scene captured under the illuminants I and I'. In the logarithmic RGB space, according to the von Kries model (5.22), it is stated that $x'_{ijc} = x_{ijc} - I_c + I'_c$. Under this assumption, the estimates $\hat{\mathrm{I}}$ and $\hat{\mathrm{I}}'$ produced by an offset equivariant model would maintain the difference between the actual and predicted illuminants ($\mathrm{I} - \mathrm{I}' = \hat{\mathrm{I}} - \hat{\mathrm{I}}'$). Consequently, this implies that the estimation error is independent of the actual color of the illuminant in the scene ($\mathrm{I} - \hat{\mathrm{I}} = \mathrm{I}' - \hat{\mathrm{I}}'$).

The usage of Convolutional Neural Networks for illuminant estimation has been extensively explored in previous research [10, 15, 114, 129]. In this study, experiments using the Color Cerberus model [144] have been conducted. This CNN is designed to take a 64×64 image as input and produce k illuminant estimates as output (in these experiments $k = 1$). The network begins by augmenting each pixel with the global average, resulting in a six-channel image. This image then undergoes a series of four modules, each consisting of a 3×3 convolution followed by a 2×2 max-pooling operation. Finally, a 1×1 convolution layer and two fully connected layers map the image to a three-dimensional illuminant estimate. All convolutional and fully connected layers are followed by the ReLU activation function.

The conversion of the original Cerberus model into its offset equivariant version is straightforward: convolutions, ReLUs, and fully connected linear layers have been replaced with their offset equivariant counterparts. The input image is transformed into the logarithmic RGB space, and the output estimate is converted back into the linear RGB space.

NUS dataset

The dataset used for this task is the one proposed by researchers from the National University of Singapore (NUS) [34]. This dataset has been specifically designed for studying computational color constancy and comprises 1853 images captured using nine different commercial cameras The dataset includes indoor and outdoor scenes in which the authors inserted a color chart to enable the computation of a reliable ground truth for the illuminant's color (a common practice is to cover the chart with a black patch during the experiments).

The original and the equivariant versions of the Color Cerberus are trained using the procedure described in the reference paper. These two models were evaluated through three-fold cross-validation on the entire dataset, using the Reproduction Angular Error ([55]) as the performance measure:

$$E(\hat{\mathrm{I}}, \mathrm{I}) = \arccos\left(\frac{\hat{I}_r / I_r + \hat{I}_g / I_g + \hat{I}_b / I_b}{\sqrt{3\left(\hat{I}_r^2 / I_r^2 + \hat{I}_g^2 / I_g^2 + \hat{I}_b^2 / I_b^2\right)}} \right). \tag{5.26}$$

Results are reported in Table 5.1. The accuracy of the two models is very similar, with the original version achieving a slightly lower median error, but only for the original test images without distortions. To assess the robustness of the trained models,

		Med. angular error (deg)		
Network	# par.	S 0.0	S 0.5	S 0.9
Original	206 345	**1.93**	11.5	30.3
Equivariant	196 668	2.03	**2.30**	**3.44**

Table 5.1: Comparison of original and offset equivariant color Cerberus on NUS images. Performance are measured in terms of median reproduction angular error and computed by three-fold cross validation on the whole dataset. Three levels of distortions are considered: no distortion ($S = 0.0$), moderate ($S = 0.5$) and strong distortion ($S = 0.9$).

a test is conducted in which the test images have been distorted by applying a global artificial illuminant with random hue and variable saturation. The results are shown in Figure 5.2. The equivariant version exhibits a slow degradation in accuracy, remaining quite good even for extreme illuminants (the median angular error is about 3.4 when the saturation is 0.9). In contrast, the original version clearly suffers from the degradation of the input (at saturation 0.9, the median error is about 30 degrees).

The stability of the equivariant color Cerberus becomes evident when examin-

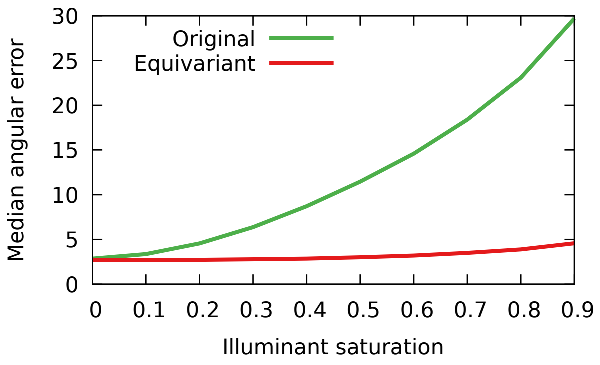


Figure 5.2: Median angular reproduction error on NUS images under an illuminant with random hue and variable saturation.

ing Figure 5.3, where the outputs of the two variants are visually compared under increasing levels of distortion.

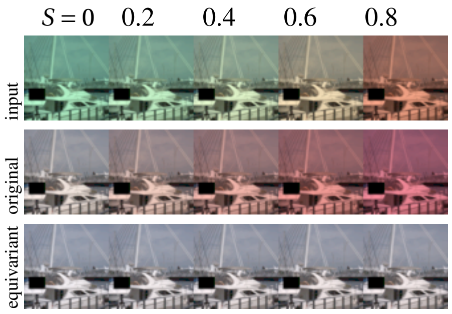


Figure 5.3: Comparison of color corrections obtained by removing the illuminant estimated by the original and the equivariant Cerberus. The image is taken from the NUS dataset and has been distorted by the introduction of an artificial illuminant with saturation varying from 0.0 (no distortion) to 0.8. For rendering purposes, the values of the pixels have been stretched, and the sRGB gamma has been applied to all images.

5.4.2 Image recognition

The second application under analysis is image classification, which has been one of the most extensively explored applications of convolutional neural networks. Numerous critical advancements in deep learning have emerged while addressing this problem.

One recurring objective in image classification is to achieve robustness to variations in the input. Data augmentation [132] is a widely employed technique to attain this robustness. In this experiment, the study focuses on the robustness of offset equivariant convolutional networks concerning substantial variations in the global color distribution, even when trained without specific data augmentation techniques.

To examine this, the experiment employs the ResNet [71] family of convolutional networks. This family of architectures is considered one of the highest-performing options for image classification and has been extensively studied. ResNets encompass convolutions, ReLU activation functions, batch normalization layers, and a final average pooling layer, followed by a fully connected layer. All these building blocks can be made offset equivariant using the procedures outlined in the previous sections. The only conversion that poses some complexity is that of the residual blocks, from which the architecture derives its name:

$$\mathrm{y} = f(\mathrm{x}) + \mathrm{x}. \tag{5.27}$$

Residual blocks directly combine the function f (which can be made equivariant) and the identity function (which is trivially equivariant). However, it's important to note that the sum of two equivariant functions is not equivariant itself. In fact, Equation (5.5) requests that the coefficients in the linear combination sum up to one. To maintain offset equivariance, the following modified residual blocks are defined:

$$\mathrm{y} = f(\mathrm{x}) + \mathrm{x} - G_m\varphi(\mathrm{x}), \tag{5.28}$$

with the average group pooling as activation function φ.

To ensure that features are uniformly assigned to the three groups, the number of channels computed by each convolution is adjusted to the closest multiple of three. Additionally, the number of output scores, computed by the final fully connected layer, is set to triple the number of classes. This allows one score to be computed for each class and for each group of features. These modifications result in an offset equivariant version of the original ResNet.

Offset equivariance can be employed to achieve illuminant invariance under the von Kries model (5.22). The input image is assumed to be in the sRGB color space. Firstly, the sRGB gamma is removed, and the image is converted into the log RGB space using Equation (5.23). Subsequently, the converted image is normalized using the mean and standard deviations of the log RGB values computed from the training set. The CNN is then applied, producing a set of $k \times 3$ scores, where k represents the number of classes. These scores are reduced to a k-dimensional vector by averaging over the groups, and finally, the softmax operator is applied to obtain posterior probabilities for the k classes.

More precisely, consider $Z \in \mathbb{R}^{k\times 3}$ as the output of the equivariant CNN for an image x, where (z_{i1}, z_{i2}, z_{i3}) denotes the triplet of scores for class i. In the log RGB space, a change in illuminant corresponds to adding the same offset I to all the pixels. Since the network is offset equivariant, the scores for the modified image will be $(z_{i1} + I_1, z_{i2} + I_2, z_{i3} + I_3)$. The posterior probability p_i for class i is calculated by applying the softmax operator to the average of these scores:

$$\begin{aligned} p_i &= \frac{e^{\frac{1}{3}(z_{i1}+I_1+z_{i2}+I_2+z_{i3}+I_3)}}{\sum_{j=1}^{k} e^{\frac{1}{3}\left(z_{j1}+I_1+z_{j2}+I_2+z_{j3}+I_3\right)}} \\ &= \frac{e^{\frac{1}{3}(I_1+I_2+I_3)} e^{\frac{1}{3}(z_{i1}+z_{i2}+z_{i3})}}{e^{\frac{1}{3}(I_1+I_2+I_3)} \sum_{j=1}^{k} e^{\frac{1}{3}\left(z_{j1}+z_{j2}+z_{j3}\right)}} \\ &= \frac{e^{\frac{1}{3}(z_{i1}+z_{i2}+z_{i3})}}{\sum_{j=1}^{k} e^{\frac{1}{3}\left(z_{j1}+z_{j2}+z_{j3}\right)}}, \end{aligned} \tag{5.29}$$

which does not depend on I. These final probability estimates can be utilized for predicting the most probable class label or as part of the cross-entropy classification loss during training.

CIFAR-10

The first dataset under consideration is the Canadian Institute for Advanced Research-10 dataset (CIFAR-10) [89]. The CIFAR-10 dataset is a subset of a larger dataset consisting of tiny images, each measuring 32 × 32 pixels. This labeled subset comprises 10 distinct classes of objects: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. These classes are non-overlapping, meaning each image portrays an object belonging to just one class. The dataset includes 6000 images for each of these ten classes, resulting in a total of 60000 images. The dataset has been partitioned into a training set (50k images) and a test set (10k images) by the dataset's authors.

The experiment conducted in the original ResNet paper, which pertains to the ResNet family of networks, was replicated on the CIFAR-10 dataset. This experiment specifically concentrated on the 20-layer network, which includes nine residual blocks. The original model was trained following the same configuration as described in the paper, resulting in a closely matching level of accuracy (8.67% classification error on the test set). The experiment was then repeated using an offset equivariant version of the network, which led to a very similar test error rate (8.83%). To assess the ability of the offset equivariant network to withstand significant variations in the illuminant, the evaluation step has been repeated by distorting the input image as follows:

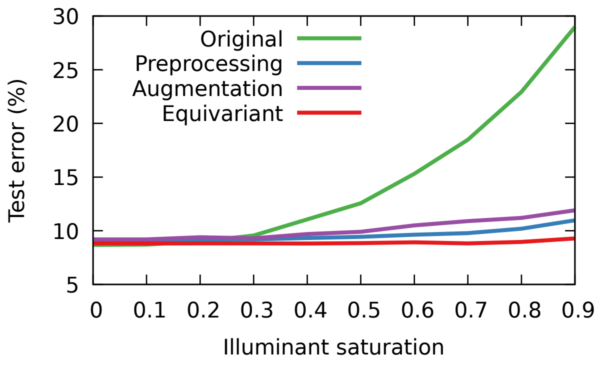


Figure 5.4: Error rates of original and equivariant ResNet-20 on CIFAR-10 test images under an illuminant with random hue and variable saturation. For the original network, the results obtained with preprocessing and data augmentation are also reported.

1. the sRGB gamma is removed;
2. a random illuminant color is generated in the HSV color space by randomly selecting the hue (H), setting the value (V) to its maximum value, and leaving the saturation (S) as a tunable parameter. This allows for adjustment of the degree of chromatic distortion;
3. the illuminant color is converted to the RGB space and applied to the whole image (Equation (5.22));
4. the sRGB gamma is restored.

Figure 5.4 reports the test accuracies obtained by the original and the equivariant model, as a function of the saturation of the generated illuminant. The comparison includes the performance obtained using two alternative strategies to enhance the original network's robustness to changes in illumination:

- Data augmentation, involving color jittering during training, was employed. Several tests were performed with various degrees of random brightness, contrast, hue, and saturation adjustments and the combination that yielded the best average performance on the test set was selected.
- Preprocessing involved applying an automatic color balancing algorithm to both the training and test images. Among the various methods in the state of the art, the algorithm for Quasi-unsupervised color constancy [10] was employed. This

choice was motivated by the fact that it does not necessitate information about the actual color of the illuminant in the scene for training, and it can function in the standard sRGB color space. Importantly, it does not rely on the device RAW space, which is the case for most methods. The online version of this algorithm pretrained on the ImageNET data in the unsupervised scenario was used.

The plot illustrates that for small saturation values, the performance of various networks is quite similar. However, as the saturation exceeds 0.3, the test error of the original ResNet begins to rise, diverging steadily from the error rate observed under neutral illumination ($S = 0$). In contrast, the offset equivariant version exhibits remarkable stability. It maintains the same error rate (within ±0.2%) for saturation values up to 0.8. Only at S = 0.9 does a noticeable decrease in accuracy occur (9.38% error), and it's only at S = 1.0 that the performance significantly degrades due to the complete elimination of information from one or more color channels (28.17% test error, not depicted in the plot for clarity). Both preprocessing and data augmentation proved to be valid strategies, but the offset equivariant version outperforms them. It does not require time-consuming parameter tuning, as data augmentation does, and it eliminates the need for training an additional complex model for preprocessing (the one used comprises more than 54 million learnable parameters).

A summary of the test results is provided in Table 5.2. The slight difference in the number of parameters arises from adjusting the channel count to ensure they are multiples of three.

		Test error (%)		
Strategy	# par.	S 0.0	S 0.5	S 0.9
Original	270410	**8.67**	12.6	29.0
Preprocessing	270410	9.19	9.43	11.0
Augmentation	270410	9.20	9.91	11.9
Equivariant	267294	8.88	**8.85**	**9.29**

Table 5.2: Comparison of original and offset equivariant ResNet-20 on CIFAR-10 test images. For the original ResNet, the versions trained with preprocessing and data augmentation are also compared. For the version with preprocessing only the parameters in the ResNet have been counted. The preprocessing module includes about 54 millions of extra parameters. Three levels of distortions are considered: no distortion ($S = 0.0$), moderate ($S = 0.5$) and strong distortion ($S = 0.9$).

Figure 5.5 illustrates how the scores and predictions obtained with the original ResNet-20 are noticeably influenced by the introduction of an artificial illuminant, even when its saturation is mild. In contrast, the output of the equivariant ResNet-20 remains stable across the entire spectrum of saturation values.

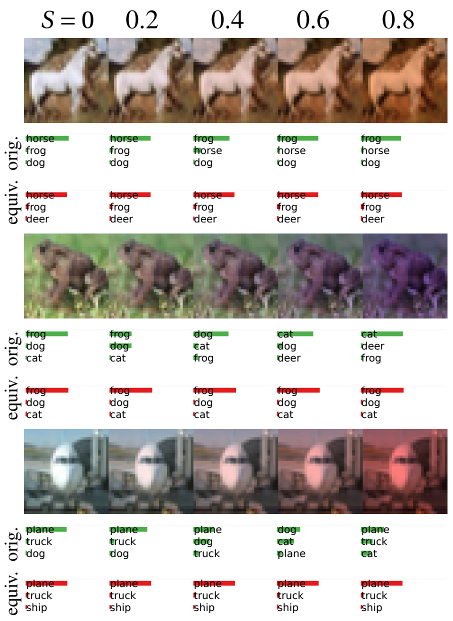


Figure 5.5: Comparison of the predictions obtained on three images taken from the CIFAR-10 test set. Each image has been distorted by the introduction of an artificial illuminant with saturation varying from 0.0 (no distortion) to 0.8. The top three classes predicted by the original and the equivariant ResNet-20 are reported below each image. The lenghts of the bars is proportional to the posterior probabilities estimated by the networks.

ILSVRC 12

For the second experiment on image recognition, the dataset introduced for the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC-12) [140] has been employed. This dataset consists of 1000 classes, carefully chosen to avoid any overlap found in the original ImageNet dataset [48], where classes are organized hierarchically. The training images, amounting to approximately 1.2 million, were directly extracted from the ImageNet dataset. In contrast, the validation set (50K) and the test set (100K) were collected from various search engines like Flickr and subsequently filtered following the ImageNet guidelines.

		Test error (%)		
Strategy	# par.	S 0.0	S 0.5	S 0.9
Original	25.6M	**24.59**	26.09	31.17
Preprocessing	25.6M	24.95	25.32	26.53
Augmentation	25.6M	28.09	29.21	33.91
Equivariant	28.8M	24.85	**24.92**	**25.71**

Table 5.3: Comparison of original and offset equivariant ResNet-50 on ILSVRC12 test images. Results obtained by using preprocessing (without counting the extra parameters) and data augmentation on top of the original ResNet-50 are also reported. Three levels of distortions are considered: no distortion ($S = 0.0$), moderate ($S = 0.5$) and strong distortion ($S = 0.9$).

Based on the experiments presented in the previous section, offset equivariant version of ResNet-50 has been trained. This version retains the same structure as the original ResNet-50 [71], with the incorporation of the equivariant layers introduced earlier.

The training procedure followed (the same as the original paper outlined by He et al. for the ImageNet dataset) included data augmentation techniques like random resizing and cropping, random horizontal flips, etc. However, due to hardware limitations (a single Nvidia RTX 3080ti graphics card with 12 GB of memory), the batch size has been reduced to 48. Subsequently, the model has been tested on the ILSVRC-12 test set and a test error of 24.85% has been obtained. To ensure a fair comparison, the training procedure was repeated with the original version of ResNet-50, maintaining the same batch size, and achieved a very similar test error of 24.59%. For a detailed comparison of the results and the number of parameters, please refer to Table 5.3.

To evaluate the robustness of both the equivariant ResNet-50 and the original ResNet-50 the illuminant has been varied, following the same procedure applied in the CIFAR-10 experiment. Figure 5.6 illustrates the error rates as functions of saturation. For small saturation values, the behavior of the two networks is quite similar. However, as saturation reaches 0.3, the error curve for the original ResNet-50 begins to rise, indicating a degradation in performance. In contrast, the equivariant ResNet-50 remains stable up to saturation values of 0.8, where it exhibits a test error of 24.83% $\pm$ 0.3, and then exhibits a slight increase at $S = 0.9$ (25.71%). To provide a comprehensive comparison, the same preprocessing and data augmentation strategies, mentioned earlier for the CIFAR-10 dataset experiments, have been applied. When preprocessing was applied, the results obtained were satisfactory with less than 1%

additional error compared to the equivariant network. However, the results with data augmentation were not as satisfactory. This could be attributed to the fact that the same augmentation parameters optimized for the CIFAR dataset have been employed, as a complete tuning specific to ILSVRC would have been computationally intensive.

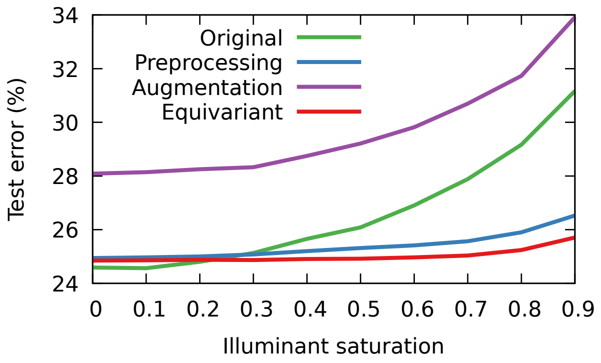


Figure 5.6: Error rates of original and equivariant ResNet-50 on ILSVRC12 test images under an illuminant with random hue and variable saturation. For the original network, the results obtained with preprocessing and data augmentation are also reported.

5.4.3 Inpainting

Image inpainting involves the task of filling in the missing portions of an incomplete image with pixels that seamlessly blend into the rest of the picture, creating a natural and coherent visual result. In the realm of image inpainting, Generative Adversarial Networks (GANs) represent the state of the art. For these experiments, the Context Encoder architecture, originally proposed by Pathak et al. [131], has been employed. Both the generator and discriminator in this architecture are sequential convolutional networks that consist of convolutional layers, ReLU activation functions, Leaky ReLU activations [120], and batch normalization layers. The generator concludes with a final hyperbolic tangent layer, while the discriminator finishes with a sigmoid activation.

To make the generator offset equivariant, its individual components have been replaced, following the guidelines outlined in Section 5.2. Additionally, the number of channels have been adjusted to ensure they were multiples of three. The preprocessing pipeline involved converting the input image into the logarithmic RGB space, and the final inpainted image was converted back into the linear RGB space. Since the output values already resided within the $[0,1]$ range, they were scaled to fit within the range [-1, 1] without the need for an additional hyperbolic tangent layer. The discriminator was retained in its original form without any modifications.

Paris Street View dataset

To analyze the image inpainting task, the Paris Street View Dataset [50] was considered. This dataset comprises images obtained from Google Street View, a vast repository of street-level images. The dataset contains nearly 10 000 images for each of the 12 cities under consideration (Paris, London, Prague, Barcelona, Milan, New York, Boston, Philadelphia, San Francisco, San Paulo, Mexico City, and Tokyo), as well as suburbs of Paris. For the experiment, the focus relied solely on the subset comprising 6492 images representing the city of Paris, divided into 6392 training images and 100 test images. Following the practices outlined in the original paper, all images were resampled to dimensions of 128×128 pixels, with the central 64×64 pixels being intentionally removed.

Both the original and modified Context Encoders were trained, and their performance was assessed on the test set. Results, measured in terms of reconstruction error using the Peak Signal-to-Noise Ratio (PSNR), are presented in Table 5.4. The performance of the two variants closely aligns, and for the scenario without distorted test images, their performance matches that which was reported in the original paper.

		PSNR (dB)		
Network	# par.	S 0.0	S 0.5	S 0.9
Original	71.14M	17.65	18.44	19.05
Equivariant	69.94M	**17.67**	**18.68**	**19.78**

Table 5.4: Comparison of original and offset equivariant Context Encoder for image inpainting. Performance are measured in terms of PSNR (higher is better). Three levels of distortions are considered: no distortion ($S = 0.0$), moderate ($S = 0.5$) and strong distortion ($S = 0.9$).

The test was repeated by distorting the test images with the application of a global artificial illuminant of random hue, and variable saturation. The procedure was the same used for illuminant estimation and image recognition tasks. Figure 5.7 shows the results obtained. In this scenario, performance improves with increasing saturation levels. However, this effect is primarily a consequence of the reduced dynamic range caused by highly saturated illuminants. Nevertheless, it is worth noting that the offset equivariant version outperforms the original version at high saturation levels.

Figure 5.8 illustrates the performance of the two models on one of the test images. Given the nature of the problem, both variants tend to align with the color distribution influenced by the illuminant. However, the equivariant version consistently fills the

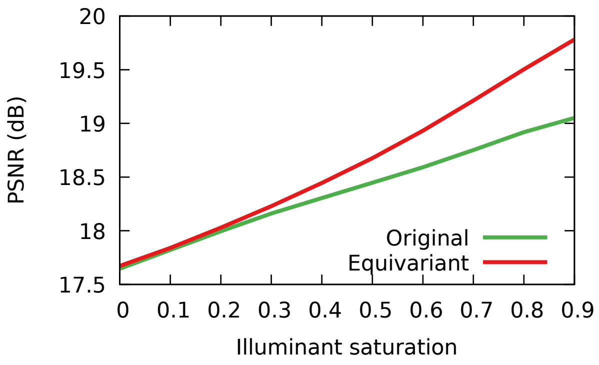


Figure 5.7: Reconstruction error (PSNR) obtained on the test images of the Paris Street View dataset, under an illuminant with random hue and variable saturation.

center with the same pattern, while the original version produces different patterns as the saturation levels change.

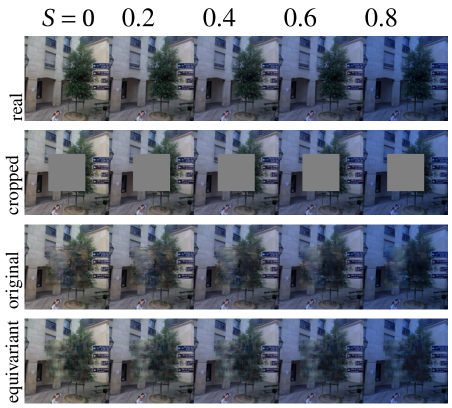


Figure 5.8: Comparison of image inpainting obtained with the original and the equivariant Context Encoder. The image is taken from the Paris Street View dataset and has been distorted by the introduction of an artificial illuminant with saturation varying from 0.0 (no distortion) to 0.8.

5.5 Summary

In this chapter, the concept of offset equivariant networks has been introduced, along with a framework for constructing them. Their applications in achieving invariance or equivariance with respect to changes in illuminant color have been demonstrated across various scenarios. In all experiments, offset equivariant networks have not only maintained the performance of the original architectures under standard conditions but have also showcased improvements when dealing with strongly non-neutral illuminants.

Unlike some approaches that rely on learned data, the consistency of offset equivariant networks is guaranteed by their mathematical structure. Therefore, they offer a preferable alternative, particularly in scenarios where robustness to rare and unusual lighting conditions is a critical requirement.

This chapter has primarily focused on color images and equivariance concerning changes in illuminant color. Future investigations may explore the application of such networks in other domains, such as multi-spectral and hyper-spectral imaging.

6 Conclusions

The objective of this thesis was to introduce an innovative approach, combining deep learning and reinforcement techniques, for addressing computational photography challenges. Specifically, three key challenges were identified and thoroughly examined in the chapters of this thesis: **(i)** the enhancement of low-quality images through artifacts-free fully interpretable algorithms, **(ii)** the lack of a photographic dataset for photo authorship attribution and photographic style transfer and **(iii)** the improvement of state-of-the-art convolutional architectures, particularly concerning color constancy in the presence of changing scene illuminants.

In Chapter 3, a novel, artifact-free, white-box methods for image enhancement has been proposed. Specifically, in Section 3.3, TreEnhance, a tree-search-based approach for image enhancement has been presented. Leveraging principles from tree-search theory and deep reinforcement learning, TreEnhance generates understandable sequences of global image processing operators. When applied to low-quality input images, these sequences significantly improve their visual content. This algorithm has been evaluated in two image enhancement scenarios: photographic image enhancement, using the Five-K dataset, and low-light image enhancement, with the Low-Light dataset. TreEnhance demonstrated impressive performance, surpassing image-to-image translation methods. In fact, in a user study assessing enhancement quality and artifact visibility, TreEnhance ranked first. However, one limitation of this method is its application of global operators to the entire input image. While highly effective for enhancing low-light images, where a uniform transformation can enhance the final result, it may be less effective in cases where specific areas of the image need individual adjustments (evident from color curve shapes after enhancement). To address such scenarios, a mask-based method in Section 3.4 has been introduced. This algorithm, structurally similar to TreEnhance, utilizes tree-search theory to iteratively select smaller image areas and applies a histogram equalization operator to these regions. This enhanced version outperformed TreEnhance on the Five-K dataset, particularly when a more localized enhancement style is required. Finally, in Section 3.6, the first explainability method for image enhancement was presented. Using a modified version of the A^* path planning algorithm, this method emulates the enhancement process of black-box image-to-image enhancement methods. Moreover, in some cases, it can even improve visual results by removing artifacts introduced by translation methods. It is important to note that all the algorithms in Chapter 3 ensure artifact-free output images, resulting in high-quality perceptual images.

To address the lack of a comprehensive photographic dataset containing recent images, Chapter 4 introduces PhotoStyle60. This dataset comprises 5708 images contributed by 60 different professional and amateur photographers. Additionally, a subset of this dataset, called PhotoStyle10, has been introduced. PhotoStyle10 consists of images from the 10 photographers in the original dataset who exhibit distinct and easily recognizable styles. Both PhotoStyle60 and PhotoStyle10 were validated using state-of-the-art methods across two key applications: photo authorship attribution and photographic style transfer. Moreover, a new multi-image style transfer method was presented and tested, aiming to transfer the style of one of the photographers in the dataset to a photo from the Five-K dataset. This innovative method achieved the top ranking in a user study, demonstrating its effectiveness in replicating a photographer's general photographic style based on their most iconic shots.

In Chapter 5, a novel convolutional model known as the "Offset Equivariant Neural Network" has been introduced. This architecture is purpose-built to address scenarios where shifts in scene illuminants occur. Unlike state-of-the-art convolutional neural networks, which exhibit performance degradation when the illuminant in test images differs from that in the training set, the proposed model displays consistent behavior, with only a slight performance drop in cases of extreme illuminant shifts. While primarily designed for color constancy, this model was also tested on image recognition and image inpainting tasks. The results achieved by this model not only surpassed those of state-of-the-art models trained under standard augmentations but also outperformed models trained with significant augmentation in illuminant variations. This underscores that the model's primary strength lies not in learning dependencies on specific data characteristics but rather in its ability to remain independent of the actual illuminant conditions during training.

6.1 Future Works

The results obtained in this thesis open up several promising directions for future research and expansion of the contributions presented. For instance, the tree-search based methods introduced in Section 3.3 and Section 3.4 can be adapted to address additional computational photography tasks such as image denoising, deblurring, deraining, retargeting, and more. Achieving this adaptation simply involves adjusting the set of editing operators to align with the specific requirements of the task at hand. Furthermore, the mask-based method outlined in Section 3.4 offers room for improvement through the incorporation of additional spatial selection techniques or by leveraging saliency/segmentation maps to detect and enhance individual object instances directly, rather than entire image regions. An exciting prospect stemming from the method introduced in Section 3.6 is the potential development of a completely interpretable approach based on path-planning. This approach could employ heuristic search techniques that involve the neural network as a function approximator. This innovative fusion of path-planning theory and deep reinforcement learning represents a promising direction for future research and development.

The dataset introduced in Chapter 4 could be expanded to include photographs contributed by beginner photographers. This expansion would enable a triple categorization of the dataset, differentiating images captured by professional, amateur, and beginner photographers. Additionally, the multi-image style transfer method could find application in a scenario where photos taken by beginners are enhanced to match the style of professional photographers. Furthermore, by leveraging one of the tree-search-based methods outlined in Chapter 3, it would be feasible to identify a sequence that transitions the style of photos edited by a beginner photographer to resemble that of an expert photographer. This system could be incorporated into an educational tool, accessible within post-processing software, offering valuable learning opportunities for photographers at different skill levels.

Until now, the concept of equivariance has primarily been confined to spatial transformations applied to images. However, based on the results obtained, the novel family of architectures introduced in Chapter 5 holds the potential for validation in other image domains, such as hyperspectral or 3D images. Notably, the equivariant layers presented in this work are versatile enough to replace the original layers in any neural network. This extension into new image domains highlights the significance of developing neural networks capable of maintaining robustness even when subjected to photometric transformations during inference.

All the solutions presented to address the challenges that inspired the development of this thesis are merely the inception of what promises to be an exciting journey of future advancements and innovations in the field of computational photography.

Bibliography

[1] Rao Muhammad Anwer, Fahad Shahbaz Khan, Joost Van De Weijer, and Jorma Laaksonen. Combining holistic and part-based deep representations for computational painting categorization. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 339–342, 2016.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Sugata Banerji and Atreyee Sinha. Painting classification using a pre-trained convolutional neural network. In *Computer Vision, Graphics, and Image Processing: ICVGIP 2016 Satellite Workshops, WCVA, DAR, and MedImage, Guwahati, India, December 19, 2016 Revised Selected Papers*, pages 168–179. Springer, 2017.

[5] Nikola Banić, Karlo Koščević, and Sven Lončarić. Unsupervised learning for color constancy. *arXiv preprint arXiv:1712.00436*, 2017.

[6] Nikola Banić and Sven Lončarić. Green stability assumption: Unsupervised learning for statistics-based illumination estimation. *Journal of Imaging*, 4(11):127, 2018.

[7] K Barnard. A comparison of computational color constancy algorithms-part ii: experiments with image data. *IEEE Trans. Pattern. Anal. Mach. Intell.*, 23(11):1209–1221, 2001.

[8] Elissavet Batziou, Konstantinos Ioannidis, Ioannis Patras, Stefanos Vrochidis, and Ioannis Kompatsiaris. Artistic neural style transfer using cyclegan and fabemd by adaptive information selection. *Pattern Recognition Letters*, 165:55–62, 2023.

[9] Georgios Batzolis, Jan Stanczuk, Carola-Bibiane Schönlieb, and Christian Etmann. Conditional image generation with score-based diffusion models. *arXiv preprint arXiv:2111.13606*, 2021.

[10] Simone Bianco and Claudio Cusano. Quasi-unsupervised color constancy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12212–12221, 2019.

[11] Simone Bianco, Claudio Cusano, Paolo Napoletano, and Raimondo Schettini. Improving CNN-based texture classification by color balancing. *Journal of Imaging*, 3(3):33, 2017.

[12] Simone Bianco, Claudio Cusano, Flavio Piccoli, and Raimondo Schettini. Artistic photo filter removal using convolutional neural networks. *Journal of Electronic Imaging*, 27(1):011004, 2017.

[13] Simone Bianco, Claudio Cusano, Flavio Piccoli, and Raimondo Schettini. Learning parametric functions for color image enhancement. In *International Workshop on Computational Color Imaging*, pages 209–220. Springer, 2019.

[14] Simone Bianco, Claudio Cusano, Flavio Piccoli, and Raimondo Schettini. Personalized image enhancement using neural spline color transforms. *IEEE Transactions on Image Processing*, 29:6223–6236, 2020.

[15] Simone Bianco, Claudio Cusano, and Raimondo Schettini. Color constancy using CNNs. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 81–89, 2015.

[16] Simone Bianco, Claudio Cusano, and Raimondo Schettini. Single and multiple illuminant estimation using convolutional neural networks. *IEEE Transactions on Image Processing*, 26(9):4347–4362, 2017.

[17] Simone Bianco, Francesca Gasparini, and Raimondo Schettini. Consensus-based framework for illuminant chromaticity estimation. *Journal of Electronic Imaging*, 17(2):023013–023013, 2008.

[18] Simone Bianco, Davide Mazzini, Paolo Napoletano, and Raimondo Schettini. Multitask painting categorization by deep multibranch neural network. *Expert Systems with Applications*, 135:90–101, 2019.

[19] Simone Bianco, Davide Mazzini, and Raimondo Schettini. Deep multibranch neural network for painting categorization. In *Image Analysis and Processing-ICIAP 2017: 19th International Conference, Catania, Italy, September 11-15, 2017, Proceedings, Part I 19*, pages 414–423. Springer, 2017.

[20] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[21] Gershon Buchsbaum. A spatial processor model for object colour perception. *Journal of the Franklin institute*, 310(1):1–26, 1980.

[22] Marco Buzzelli, Joost van de Weijer, and Raimondo Schettini. Learning illuminant estimation from object recognition. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 3234–3238. IEEE, 2018.

[23] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo 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.

[24] Weiwei Cai and Zhanguo Wei. Piigan: generative adversarial networks for pluralistic image inpainting. *IEEE Access*, 8:48451–48463, 2020.

[25] Yuanhao Cai, Xiaowan Hu, Haoqian Wang, Yulun Zhang, Hanspeter Pfister, and Donglai Wei. Learning to generate realistic noisy images via pixel-level noise-aware adversarial training. *Advances in Neural Information Processing Systems*, 34:3259–3270, 2021.

[26] Yuanhao Cai, Jing Lin, Xiaowan Hu, Haoqian Wang, Xin Yuan, Yulun Zhang, Radu Timofte, and Luc 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*, pages 17502–17511, 2022.

[27] Vlad C Cardei and Brian V Funt. Committee-based color constancy. In *Color imaging conference*, pages 311–313. Citeseer, 1999.

[28] Vlad C Cardei, Brian V Funt, and Kobus Barnard. White point estimation for uncalibrated images. In *Color Imaging Conference*, pages 97–100, 1999.

[29] Yoav Chai, Raja Giryes, and Lior Wolf. Supervised and unsupervised learning of parameterized color enhancement. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 992–1000, 2020.

[30] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3291–3300, 2018.

[31] Wei Chen, Wang Wenjing, Yang Wenhan, and Liu Jiaying. Deep retinex decomposition for low-light enhancement. In *British Machine Vision Conference*, 2018.

[32] Yingshu Chen, Tuan-Anh Vu, Ka-Chun Shum, Sai-Kit Yeung, and Binh-Son Hua. Time-of-day neural style transfer for architectural photographs. In *2022 IEEE International Conference on Computational Photography (ICCP)*, pages 1–12. IEEE, 2022.

[33] Zhe Chen, Wenhai Wang, Enze Xie, Tong Lu, and Ping Luo. Towards ultra-resolution neural style transfer via thumbnail instance normalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 393–400, 2022.

[34] Dongliang Cheng, Dilip K Prasad, and Michael S Brown. Illuminant estimation for color constancy: why spatial-domain methods work and the role of the color distribution. *JOSA A*, 31(5):1049–1058, 2014.

[35] Dongliang Cheng, Brian Price, Scott Cohen, and Michael S Brown. Effective learning-based illuminant estimation using simple features. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1000–1008, 2015.

[36] Hsiu-Sen Chiang, Mu-Yen Chen, and Yu-Jhih Huang. Wavelet-based eeg processing for epilepsy detection using fuzzy entropy and associative petri net. *IEEE Access*, 7:103255–103262, 2019.

[37] Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

[38] Wei-Ta Chu and Yi-Ling Wu. Image style classification based on learnt deep correlation features. *IEEE Transactions on Multimedia*, 20(9):2491–2502, 2018.

[39] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[40] Taco S Cohen and Max Welling. Steerable CNNs. *arXiv preprint arXiv:1612.08498*, 2016.

[41] Marco Cotogni and Claudio Cusano. Explaining image enhancement black-box methods through a path planning based algorithm. *Multimedia Tools and Applications*, pages 1–20, 2023.

[42] Marco Cotogni and Claudio Cusano. Treenhance: A tree search method for low-light image enhancement. *Pattern Recognition*, 136:109249, 2023.

[43] Claudio Cusano. Notes on machine learning. Technical report, University of Pavia, 2023.

[44] Pim De Haan, Maurice Weiler, Taco Cohen, and Max Welling. Gauge equivariant mesh CNNs: Anisotropic convolutions on geometric graphs. *arXiv preprint arXiv:2003.05425*, 2020.

[45] José de Jesús Rubio. Stability analysis of the modified levenberg–marquardt algorithm for the artificial neural network training. *IEEE transactions on neural networks and learning systems*, 32(8):3510–3524, 2020.

[46] José de Jesús Rubio, Marco Antonio Islas, Genaro Ochoa, David Ricardo Cruz, Enrique Garcia, and Jaime Pacheco. Convergent newton method and neural network for the electric energy usage prediction. *Information Sciences*, 585:89–112, 2022.

[47] José de Jesús Rubio, Edwin Lughofer, Jeff Pieper, Panuncio Cruz, Dany Ivan Martinez, Genaro Ochoa, Marco Antonio Islas, and Enrique Garcia. Adapting h-infinity controller for the desired reference tracking of the sphere position in the maglev process. *Information Sciences*, 569:669–686, 2021.

[48] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[49] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[50] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei Efros. What makes Paris look like Paris? *ACM Transactions on Graphics*, 31(4), 2012.

[51] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[52] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[53] Graham D. Finlayson, Steven D. Hordley, and Paul M. Hubel. Color by correlation: A simple, unifying framework for color constancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1209–1221, 2001.

[54] Graham D Finlayson and Elisabetta Trezzi. Shades of gray and colour constancy. In *Color and Imaging Conference*, volume 2004, pages 37–41. Society for Imaging Science and Technology, 2004.

[55] Graham D Finlayson, Roshanak Zakizadeh, and Arjan Gijsenij. The reproduction angular error for evaluating the performance of illuminant estimation algorithms. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1482–1488, 2016.

[56] David A Forsyth. A novel algorithm for color constancy. *International Journal of Computer Vision*, 5(1):5–35, 1990.

[57] Ryosuke Furuta, Naoto Inoue, and Toshihiko Yamasaki. PixelRL: Fully convolutional network with reinforcement learning for image processing. *IEEE Transactions on Multimedia*, 22(7):1704–1719, 2019.

[58] Siddhartha Gairola, Rajvi Shah, and PJ Narayanan. Unsupervised image style embeddings for retrieval and recognition tasks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3281–3289, 2020.

[59] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[60] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[61] Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. Deep bilateral learning for real-time image enhancement. *ACM Transactions on Graphics (TOG)*, 36(4):1–12, 2017.

[62] Arjan Gijsenij, Theo Gevers, and Joost Van De Weijer. Computational color constancy: Survey and experiments. *IEEE transactions on image processing*, 20(9):2475–2489, 2011.

[63] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[64] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *International conference on machine learning*, pages 1319–1327. PMLR, 2013.

[65] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[66] T Goodman. Colour design: theories and applications. *International Standards for Colour*, pages 417–452, 2012.

[67] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[68] Christine Guillemot and Olivier Le Meur. Image inpainting: Overview and recent advances. *IEEE signal processing magazine*, 31(1):127–144, 2013.

[69] Chunle Guo, Chongyi Li, Jichang Guo, Chen Change Loy, Junhui Hou, Sam Kwong, and Runmin Cong. Zero-reference deep curve estimation for low-light image enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1780–1789, 2020.

[70] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[71] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[72] Mingming He, Jing Liao, Dongdong Chen, Lu Yuan, and Pedro V Sander. Progressive color transfer with dense semantic correspondences. *ACM Transactions on Graphics (TOG)*, 38(2):1–18, 2019.

[73] Kibeom Hong, Seogkyu Jeon, Huan Yang, Jianlong Fu, and Hyeran Byun. Domain-aware universal style transfer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14609–14617, 2021.

[74] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pages 2366–2369. IEEE, 2010.

[75] Yuanming Hu, Hao He, Chenxi Xu, Baoyuan Wang, and Stephen Lin. Exposure: A white-box photo post-processing framework. *ACM Transactions on Graphics (TOG)*, 37(2):1–17, 2018.

[76] Xingsheng Huang, Sheng-hua Zhong, and Zhijiao Xiao. Fine-art painting classification via two-channel deep residual network. In *Advances in Multimedia Information Processing–PCM 2017: 18th Pacific-Rim Conference on Multimedia, Harbin, China, September 28-29, 2017, Revised Selected Papers, Part II 18*, pages 79–88. Springer, 2018.

[77] Andrey Ignatov, Nikolay Kobyshev, Radu Timofte, Kenneth Vanhoey, and Luc Van Gool. Dslr-quality photos on mobile devices with deep convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3277–3285, 2017.

[78] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):1–14, 2017.

[79] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[80] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[81] Yifan Jiang, Xinyu Gong, Ding Liu, Yu Cheng, Chen Fang, Xiaohui Shen, Jianchao Yang, Pan Zhou, and Zhangyang Wang. Enlightengan: Deep light enhancement without paired supervision. *IEEE Transactions on Image Processing*, 30:2340–2349, 2021.

[82] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711, 2016.

[83] Sergey Karayev, Matthew Trentacoste, Helen Han, Aseem Agarwala, Trevor Darrell, Aaron Hertzmann, and Holger Winnemoeller. Recognizing image style. *arXiv preprint arXiv:1311.3715*, 2013.

[84] Onur Keleş, M Akın Yılmaz, A Murat Tekalp, Cansu Korkmaz, and Zafer Doğan. On the computation of psnr for a set of images or video. In *2021 Picture Coding Symposium (PCS)*, pages 1–5, 2021.

[85] Fahad Shahbaz Khan, Shida Beigpour, Joost Van de Weijer, and Michael Felsberg. Painting-91: a large scale database for computational painting categorization. *Machine vision and applications*, 25:1385–1397, 2014.

[86] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2426–2435, 2022.

[87] Hanul Kim, Su-Min Choi, Chang-Su Kim, and Yeong Jun Koh. Representative color transform for image enhancement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4459–4468, 2021.

[88] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning*, pages 2747–2755. PMLR, 2018.

[89] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[90] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[91] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[92] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8183–8192, 2018.

[93] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[94] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[95] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan

Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[96] Jingyuan Li, Ning Wang, Lefei Zhang, Bo Du, and Dacheng Tao. Recurrent feature reasoning for image inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7760–7768, 2020.

[97] Ming Li, Chunyang Ye, and Wei Li. High-resolution network for photorealistic style transfer. *arXiv preprint arXiv:1904.11617*, 2019.

[98] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast image and video style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3809–3817, 2019.

[99] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. A closed-form solution to photorealistic image stylization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 453–468, 2018.

[100] Yuelong Li, Mohammad Tofighi, Junyi Geng, Vishal Monga, and Yonina C Eldar. Efficient and interpretable deep blind image deblurring via algorithm unrolling. *IEEE Transactions on Computational Imaging*, 6:666–681, 2020.

[101] Seokjae Lim and Wonjun Kim. Dslr: Deep stacked laplacian restorer for low-light image enhancement. *IEEE Transactions on Multimedia*, 2020.

[102] Jing Lin, Yuanhao Cai, Xiaowan Hu, Haoqian Wang, Xin Yuan, Yulun Zhang, Radu Timofte, and Luc Van Gool. Coarse-to-fine sparse transformer for hyper-spectral image reconstruction. *arXiv preprint arXiv:2203.04845*, 2022.

[103] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[104] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European conference on computer vision (ECCV)*, pages 85–100, 2018.

[105] Hongyu Liu, Ziyu Wan, Wei Huang, Yibing Song, Xintong Han, and Jing Liao. Pd-gan: Probabilistic diverse gan for image inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9371–9381, 2021.

[106] Risheng Liu, Long Ma, Jiaao Zhang, Xin Fan, and Zhongxuan Luo. Retinex-inspired unrolling with cooperative prior architecture search for low-light image enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10561–10570, 2021.

[107] Xiaokai Liu, Weihao Ma, Xiaorui Ma, and Jie Wang. Lae-net: A locally-adaptive embedding network for low-light image enhancement. *Pattern Recognition*, page 109039, 2022.

[108] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[109] Zhi-Song Liu, Vicky Kalogeiton, and Marie-Paule Cani. Multiple style transfer via variational autoencoder. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2413–2417. IEEE, 2021.

[110] ImageMagick Studio LLC. Imagemagick.

[111] Alexandro López-González, JA Meda Campaña, EG Hernández Martínez, and P Paniagua Contro. Multi robot distance based formation using parallel genetic algorithm. *Applied Soft Computing*, 86:105929, 2020.

[112] Kin Gwn Lore, Adedotun Akintayo, and Soumik Sarkar. Llnet: A deep autoencoder approach to natural low-light image enhancement. *Pattern Recognition*, 61:650–662, 2017.

[113] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[114] Zhongyu Lou, Theo Gevers, Ninghang Hu, Marcel P Lucassen, et al. Color constancy by deep learning. In *BMVC*, pages 76–1, 2015.

[115] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4990–4998, 2017.

[116] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471, 2022.

[117] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[118] Feifan Lv, Feng Lu, Jianhua Wu, and Chongsoon Lim. MBLLEN: Low-light image/video enhancement using CNNs. In *BMVC*, page 220, 2018.

[119] Xiaoqian Lv, Yujing Sun, Jun Zhang, Feng Jiang, and Shengping Zhang. Low-light image enhancement via deep retinex decomposition and bilateral learning. *Signal Processing: Image Communication*, 99:116466, 2021.

[120] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, pages 1–6, 2013.

[121] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using convolutional auto-encoders with symmetric skip connections. *arXiv preprint arXiv:1606.08921*, 2016.

[122] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, pages 1–14, 2018.

[123] Tom M Mitchell. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, 1980.

[124] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[125] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[126] Dante Mújica-Vargas. Superpixels extraction by an intuitionistic fuzzy clustering algorithm. *Journal of applied research and technology*, 19(2):140–152, 2021.

[127] Naila Murray, Luca Marchesotti, and Florent Perronnin. Ava: A large-scale database for aesthetic visual analysis. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2408–2415. IEEE, 2012.

[128] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *Advances in neural information processing systems*, 29, 2016.

[129] Seoung Wug Oh and Seon Joo Kim. Approaching the computational color constancy as a classification problem through deep learning. *Pattern Recognition*, 61:405–416, 2017.

[130] Jongchan Park, Joon-Young Lee, Donggeun Yoo, and In So Kweon. Distort-and-recover: Color enhancement using deep reinforcement learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5928–5936, 2018.

[131] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[132] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[133] Stephen M Pizer, E Philip Amburn, John D Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer vision, graphics, and image processing*, 39(3):355–368, 1987.

[134] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Equivariance through parameter-sharing. In *International Conference on Machine Learning*, pages 2892–2901. PMLR, 2017.

[135] Praveen Ravirathinam, Divyam Goel, and J Jennifer Ranjani. C-lienet: A multi-context low-light image enhancement network. *IEEE Access*, 9:31053–31064, 2021.

[136] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[137] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241, 2015.

[138] Charles Rosenberg, Martial Hebert, and Sebastian Thrun. Color constancy using kl-divergence. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 239–246. IEEE, 2001.

[139] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[140] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[141] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022.

[142] Babak Saleh and Ahmed Elgammal. Large-scale classification of fine-art paintings: Learning the right metric on the right feature. *arXiv preprint arXiv:1505.00855*, 2015.

[143] Hiroshi Sasaki, Chris G Willcocks, and Toby P Breckon. Unit-ddpm: Unpaired image translation with denoising diffusion probabilistic models. *arXiv preprint arXiv:2104.05358*, 2021.

[144] A Savchik, E Ershov, and S Karpenko. Color cerberus. In *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 355–359, 2019.

[145] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[146] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[147] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4570–4580, 2019.

[148] Wu Shi, Chen Change Loy, and Xiaoou Tang. Deep specialized network for illuminant estimation. In *European Conference on Computer Vision*, pages 371–387. Springer, 2016.

[149] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153, 2017.

[150] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[151] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters Chess, Shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[152] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[153] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[154] Trisha Singhal, Junhua Liu, Lucienne Blessing, and Kwan Hui Lim. Photozilla: A large-scale photography dataset and visual embedding for 20 photography styles. *arXiv preprint arXiv:2106.11359*, 2021.

[155] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[156] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[157] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[158] Wei Ren Tan, Chee Seng Chan, Hernán E Aguirre, and Kiyoshi Tanaka. Ceci n'est pas une pipe: A deep convolutional network for fine-art paintings classification. In *2016 IEEE international conference on image processing (ICIP)*, pages 3703–3707. IEEE, 2016.

[159] Christopher Thomas and Adriana Kovashka. Seeing behind the camera: Identifying the authorship of a photograph. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3494–3502, 2016.

[160] Yi-Hsuan Tsai, Xiaohui Shen, Zhe Lin, Kalyan Sunkavalli, Xin Lu, and Ming-Hsuan Yang. Deep image harmonization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3789–3797, 2017.

[161] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. *arXiv preprint arXiv:2204.01697*, 2022.

[162] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[163] Joost Van De Weijer, Theo Gevers, and Arjan Gijsenij. Edge-based color constancy. *IEEE Transactions on image processing*, 16(9):2207–2214, 2007.

[164] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[165] J von Kries. Chromatic adaptation, festschrift der albercht-ludwig-universität, 1902.

[166] Ziyu Wan, Bo Zhang, Dong Chen, Pan Zhang, Fang Wen, and Jing Liao. Old photo restoration via deep latent space translation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2071–2087, 2022.

[167] Pei Wang, Yijun Li, and Nuno Vasconcelos. Rethinking and improving the robustness of image style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 124–133, 2021.

[168] Ruixing Wang, Qing Zhang, Chi-Wing Fu, Xiaoyong Shen, Wei-Shi Zheng, and Jiaya Jia. Underexposed photo enhancement using deep illumination estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6849–6857, 2019.

[169] Xiaohui Wang, Yiran Lyu, Junfeng Huang, Ziying Wang, and Jingyan Qin. Interactive artistic multi-style transfer. *International Journal of Computational Intelligence Systems*, 14:1–13, 2021.

[170] Xin Wang, Geoffrey Oxholm, Da Zhang, and Yuan-Fang Wang. Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5239–5247, 2017.

[171] Zhaowen Wang, Ding Liu, Jianchao Yang, Wei Han, and Thomas Huang. Deep networks for image super-resolution with sparse prior. In *Proceedings of the IEEE international conference on computer vision*, pages 370–378, 2015.

[172] Michael J Wilber, Chen Fang, Hailin Jin, Aaron Hertzmann, John Collomosse, and Serge Belongie. Bam! the behance artistic media dataset for recognition beyond photography. In *Proceedings of the IEEE international conference on computer vision*, pages 1202–1211, 2017.

[173] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[174] Runsheng Xu, Zhengzhong Tu, Yuanqi Du, Xiaoyu Dong, Jinlong Li, Zibo Meng, Jiaqi Ma, Alan Bovik, and Hongkai Yu. Pik-fix: Restoring and colorizing old photos. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1724–1734, 2023.

[175] Chenggang Yan, Biao Gong, Yuxuan Wei, and Yue Gao. Deep multi-view enhancement hashing for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(4):1445–1451, 2020.

[176] Chenggang Yan, Yiming Hao, Liang Li, Jian Yin, Anan Liu, Zhendong Mao, Zhenyu Chen, and Xingyu Gao. Task-adaptive attention for image captioning. *IEEE Transactions on Circuits and Systems for Video technology*, 32(1):43–51, 2021.

[177] Chenggang Yan, Zhisheng Li, Yongbing Zhang, Yutao Liu, Xiangyang Ji, and Yongdong Zhang. Depth image denoising using nuclear norm and learning graph model. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 16(4):1–17, 2020.

[178] Chenggang Yan, Lixuan Meng, Liang Li, Jiehua Zhang, Zhan Wang, Jian Yin, Jiyong Zhang, Yaoqi Sun, and Bolun Zheng. Age-invariant face recognition by multi-feature fusionand decomposition with self-attention. *ACM Transactions*

on Multimedia Computing, Communications, and Applications, 18(1s):1–18, 2022.

[179] Chenggang Yan, Tong Teng, Yutao Liu, Yongbing Zhang, Haoqian Wang, and Xiangyang Ji. Precise no-reference image quality evaluation based on distortion identification. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 17(3s):1–21, 2021.

[180] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6721–6729, 2017.

[181] Huan Yang, Baoyuan Wang, Noranart Vesdapunt, Minyi Guo, and Sing Bing Kang. Personalized exposure control using adaptive metering and reinforcement learning. *IEEE transactions on visualization and computer graphics*, 25(10):2953–2968, 2018.

[182] Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *Constructive Approximation*, pages 1–68, 2021.

[183] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5505–5514, 2018.

[184] Ke Yu, Chao Dong, Liang Lin, and Chen Change Loy. Crafting a toolchain for image restoration by deep reinforcement learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2443–2452, 2018.

[185] Runsheng Yu, Wenyu Liu, Yasen Zhang, Zhi Qu, Deli Zhao, and Bo Zhang. Deepexposure: Learning to expose photos with asynchronously reinforced adversarial learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2153–2163, 2018.

[186] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[187] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5728–5739, 2022.

[188] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Learning enriched features for real image restoration and enhancement. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*, pages 492–511. Springer, 2020.

[189] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14821–14831, 2021.

[190] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

[191] Hang Zhang and Kristin Dana. Multi-style generative network for real-time transfer. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[192] Hongming Zhang and Tianyang Yu. Taxonomy of reinforcement learning algorithms. *Deep Reinforcement Learning: Fundamentals, Research and Applications*, pages 125–133, 2020.

[193] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017.

[194] Lin Zhang, Lijun Zhang, Xiao Liu, Ying Shen, Shaoming Zhang, and Shengjie Zhao. Zero-shot restoration of back-lit images using deep internal learning. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 1623–1631, 2019.

[195] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[196] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[197] Yonghua Zhang, Jiawan Zhang, and Xiaojie Guo. Kindling the darkness: A practical low-light image enhancer. In *Proceedings of the 27th ACM international conference on multimedia*, pages 1632–1640, 2019.

[198] Yulun Zhang, Chen Fang, Yilin Wang, Zhaowen Wang, Zhe Lin, Yun Fu, and Jimei Yang. Multimodal style transfer via graph cuts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5943–5951, 2019.

[199] Zhaoyang Zhang, Yitong Jiang, Jun Jiang, Xiaogang Wang, Ping Luo, and Jinwei Gu. Star: A structure-aware lightweight transformer for real-time image enhancement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4106–4115, 2021.

[200] Lei Zhao, Qihang Mo, Sihuan Lin, Zhizhong Wang, Zhiwen Zuo, Haibo Chen, Wei Xing, and Dongming Lu. Uctgan: Diverse image inpainting based on unsupervised cross-space translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5741–5750, 2020.

[201] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. Pluralistic image completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1438–1447, 2019.

[202] Anqi Zhu, Lin Zhang, Ying Shen, Yong Ma, Shengjie Zhao, and Yicong Zhou. Zero-shot restoration of underexposed images via robust retinex decomposition. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2020.

[203] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560, 1997.

[204] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[205] Luisa M. Zintgraf, Taco S. Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.