

Using Neural Networks as a Metric Towards Optimal Automated Image Enhancement

Master's Thesis in Computer Science

submitted by

Michael Fischer

31.01.2020



Julius-Maximilians-Universität Würzburg

Chair of Computer Science X

Author: Michael Fischer

Examiner: Prof. Dr. Andreas Hotho
Prof. Dr. Frank Puppe

Supervisor: M. Sc. Konstantin Kobs

This thesis was created during the time from 01.10.2019 to 31.01.2020 at the Chair of Computer Science X (Data Science) at the University of Würzburg to obtain the educational degree Master of Science. It was developed under the supervision of M. Sc. Konstantin Kobs and examined by Prof. Dr. Andreas Hotho and Prof. Dr. Frank Puppe.

Abstract

With the rising popularity of smartphones and visual-based social media, the number of pictures we take increases steadily. While photography can significantly benefit from image enhancement, casual users generally do not possess the necessary knowledge and patience to manually edit their images. To automate this process, we investigate the use of a neural network as a perceptually motivated loss function to guide an image enhancement network towards visually appealing results. Our presented approach combines a Context Aggregation Network to manipulate the source image with a set of learned filters and a CNN to predict the aesthetic score of the manipulation. By adjusting the filter intensities according to the predicted aesthetic score, our method automatically maximizes image beauty. We thoroughly investigate design decisions and exhibit weaknesses of the CNN that serves as our metric. In a qualitative experiment, we compare our enhancement results to two other, strong baselines and show that our results significantly outperform the original, unedited images. In consequence, our approach can be used to automatically enhance images and thus reduce workload and increase quality of experience for users. Unlike other tools for automated image enhancement, our method is transparent and publicly available as free open-source software.

Abstract

Mit der zunehmenden Popularität von Smartphones und sozialen, bildbasierten Netzwerken steigt auch die Zahl der aufgenommenen Fotos. Photographie und Bildästhetik profitieren im Allgemeinen stark von digitaler Nachbearbeitung, jedoch verfügen nur wenige Nutzer über das nötige Wissen, diese — oft sehr zeitaufwändigen — Bearbeitungsschritte effizient und zielgerichtet durchzuführen. Die vorliegende Arbeit untersucht daher, inwiefern sich ein neuronales Netz als wahrnehmungsbasierte Metrik einsetzen lässt, um die Nachbearbeitung von Bildern zu automatisieren und zu einem möglichst ästhetischen Endergebnis zu bringen. Zu diesem Zweck wird ein Context Aggregation Network entworfen, welches das Originalbild mit verschiedenen Bearbeitungsfiltern verändert. Ein nachgeschaltetes CNN bewertet daraufhin die Ästhetik der Bearbeitung. Diese Bewertung ist Grundlage der anschließenden Bildoptimierung, in deren Laufe die Filterintensitäten angepasst werden, um so die Ästhetik der Bearbeitung zu maximieren. Im Zuge der Arbeit entsteht so eine automatisierte Bildbearbeitungspipeline, deren Ergebnisse in einem qualitativen Experiment mit zwei starken Baselines verglichen werden. Wir zeigen, dass unsere Ergebnisse signifikant besser bewertet werden als die unbearbeiteten Ausgangsbilder. Im Gegensatz zu anderen automatischen Bildbearbeitungstools ist unsere Methode transparent und als kostenlose Open-Source-Software öffentlich verfügbar.

Contents

1	Introduction	1
2	Preliminaries	3
2.1	Deep Convolutional Neural Networks	3
2.1.1	Relevant Classifiers	5
2.2	Fully Convolutional Networks	7
2.2.1	Context Aggregation Networks	7
2.3	Metrics and Loss Functions	10
2.3.1	Earth Mover’s Distance	10
2.3.2	Peak Signal-to-Noise Ratio	11
2.3.3	Structural Similarity	12
3	Related Work	15
3.1	On Image Assessment	15
3.1.1	AVA: Visual Aesthetic Assessment	16
3.1.2	Neural Image Assessment	17
3.2	On Image Enhancement	19
3.2.1	MIT-Adobe FiveK	19
3.2.2	Learning Image Enhancement	19
3.2.3	Learning Perceptual Image Enhancement	22
4	Method	25
4.1	Problem Definition	25
4.2	Approach	26
4.2.1	Image Manipulation Model	26
4.2.2	Quality Assessor	35
4.2.3	Loss and Backpropagation	38
4.3	Design and Implementation	40
4.3.1	Visual Verification	42
4.3.2	Application	51

5 Evaluation & Experiments	53
5.1 Qualitative Evaluation	53
5.1.1 Baselines	54
5.1.2 Realization	56
5.1.3 Results	57
5.2 Experiments	61
5.2.1 Local Gradient Application	62
5.2.2 Timing	63
5.2.3 Extension: Video Editing	64
6 Discussion	67
7 Conclusion	71
Appendix	79
A Network Architectures	79
B Metrics and Measurements	80
C Supplement Material	82

List of Abbreviations

ABN	Adaptive Brightness Normalization
AVA	A Large-Scale Database for Aesthetic Visual Analysis (Dataset)
BN	Batch Normalization
BRI	Brightness (Image Filter)
CAN	Context Aggregation Network
CDF	Cumulative Distribution Function
CNN	Convolutional Neural Network
CON	Contrast (Image Filter)
CPU	Central Processing Unit
DNG	Digital Negative
DNN	Deep Neural Network
DSSIM	Structural Dissimilarity
EMD	Earth Mover's Distance
FCN	Fully Convolutional Networks
FLOPS	Floating Point Operations Per Second
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
GUI	Graphical User Interface

HD	High Definition
HIG	Highlights (Image Filter)
HSP	Hue Saturation Perceived-Brightness (Color Space)
HSV	Hue Saturation Value (Color Space)
HVS	Human Visual System
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ISO	International Organization for Standardization
LLF	Local Laplacian Filter
MAE	Mean Absolute Error
MIT	Massachusetts Institute of Technology
ML	Machine Learning
MSE	Mean Squared Error
MSSIM	Mean Structural Similarity
MS-SSIM	Multi-Scale Structural Similarity
NIMA	Neural Image Assessment
NLD	Non-Local Dehazing
NLP	Natural Language Processing
PIL	Python Imaging Library
PSNR	Peak Signal-to-Noise Ratio
ReLU	Rectified Linear Unit
RGB	Red Green Blue (Color Space)
RL	Reinforcement Learning
ROF	Rudin-Osher-Fatemi
SAT	Saturation (Image Filter)
SGD	Stochastic Gradient Descent

SHA	Shadows (Image Filter)
SLR	Single-Lens Reflex (Camera)
SNR	Signal-to-Noise Ratio
SSIM	Structural Similariy
SVM	Support Vector Machine
VGG	Visual Geometry Group
WGAN	Wasserstein-GAN

List of Figures

2.1	Comparison between dilated convolutions	9
2.2	MSE compared to SSIM	13
3.1	Exemplary AVA images	17
3.2	Mean and standard deviation for selected images	18
3.3	MIT-Adobe FiveK expert retouching	20
3.4	Selected CAN operator results	22
3.5	Guided local Laplacian filtering	23
4.1	Structure of the enhancement pipeline	26
4.2	Filter performance for the employed filters	28
4.3	Exemplary construction of filter layers	30
4.4	CAN training loss	30
4.5	CAN performance comparison	33
4.6	Fine-tuning operators to alleviate artefacts	34
4.7	NIMA training loss	36
4.8	NIMA prediction comparison	37
4.9	Desired distribution for optimization	39
4.10	Comparison between loss functions	40
4.11	Comparison between NIMA classifiers	41
4.12	Comparison between optimizers	42
4.13	NIMA score change for varying filter intensities	43
4.14	NIMA brightness insensitivity	44
4.15	Enhancement performance with ABN and retrained NIMA I .	47
4.16	Enhancement performance with ABN and retrained NIMA II .	48
4.17	Enhancement performance with ABN and retrained NIMA III .	49
4.18	Final enhancement pipeline	50
4.19	Optimization results for initial user presets	51
4.20	Application GUI	52
5.1	Image editing comparison	55

5.2	Ranking results	57
5.3	Absolute ranking differences	59
5.4	User subjectivity in image ratings	60
5.5	User ratings with respect to filter intensities	61
5.6	Adversarial examples and gradient smoothing	62
5.7	Filter convergence	64

List of Tables

2.1	VGG architectures	5
2.2	VGG parameter count	6
4.1	CAN24 architecture overview	27
4.2	Parametrizations for LLF and NLD	29
4.3	CAN24 performance on test set	31
4.4	Comparison of different NIMA base models	37
5.1	SSIM and NIMA score for the evaluation baselines	56
5.2	Relative performance and rating difference	58
5.3	Timing on CPU and GPU	63

1

Introduction

According to a recent study by the opinion research center Statista, the number of pictures taken in 2016 exceeded one trillion images and is predicted to grow even further, with smartphone cameras accounting for an estimated 85% of the total number [32]. The mainstream popularity of visual-content-based social media networks like Instagram, Snapchat and Facebook further promotes this trend and infers an interesting relation: Considering that smartphone- and social media usage correlate with young age [9], it is highly probable that the majority of the aforementioned images is taken by relatively young people, with little to no knowledge in professional photography or image enhancement. Yet, the photographers aspire to have their pictures look particularly attractive and appealing — especially so in the context of social media. The necessary enhancements typically are conducted manually: Among smartphone users, it currently is common practice to cycle the picture through a set of pre-defined filter masks and then manually select the “best-looking” version. Photography experts, on the other hand, determine the hyperparameters for the employed editing techniques in extensive, time-consuming analysis.

Albeit the term “best-looking” refers to the ill-posed problem of maximum image beauty, recent advances in computer vision research made it possible to accurately assess both technical and perceptual image quality: The Neural Image Assessment (NIMA) architecture rates the aesthetic of a picture in accordance with human perception, while considering semantic content and photographic conditions [8]. Relating to the previously men-

Introduction

tioned process of enhancing image quality by manual post-processing, we pose the question whether these post-processing steps can be automated in a way that generates perceptually superior results without user interference by using NIMA as a metric. To this end, we first train a Context Aggregation Network (CAN) to apply a set of photographic and artistic filters which are similar to those used by professional photographers. We then use NIMA to rate the filter combination that currently is employed onto the source image, and — using the NIMA score as a metric — guide the intensity values of the aforementioned filters towards maximum image beauty and high perceptual quality. We show that, in spite of the subjective nature of the problem, our approach yields images of high aesthetic in its own, characteristic editing style, and compare our results to other enhancement algorithms and the manually edited pictures of a professional photographer. Unlike other automatic image enhancement frameworks, our approach yields different results for varying intensity starting configurations and thus embeds user preferences into the enhancement outcome. We share our code and make the approach publicly available as open-source software.

The remainder of this thesis is structured as follows: In chapter two, we expound the fundamentals of CNNs and introduce metrics and concepts that will be used throughout the course of this thesis. Chapter three assesses related work and points out the differences and commonalities between other, comparable work and our approach. In chapter four, we present our devised method for automatic image enhancement and show first results. Chapter five qualitatively evaluates the approach’s performance and gives a brief overview over experiments and extensions, while chapter six presents an in-depth discussion of the obtained results. Lastly, chapter seven draws a conclusion and highlights potential starting points for future work.

2

Preliminaries

For an accurate comprehension of the following chapters, it is essential to establish a common understanding of the basic terminology that will be used throughout this work. This section introduces relevant concepts and terms.

2.1 Deep Convolutional Neural Networks

The concept of Convolutional Neural Networks (CNNs) was inspired by the Human Visual System (HVS) and is based on the Neocognitron, which models the basic cells in the human cortex [1]. The first successful use of gradient-based learning in image recognition with a CNN dates back to Yann LeCun’s LeNet in 1998 [11]. Since then, research has presented a variety of different CNN architectures, although progress was initially stalled by the intense resource consumption of the training procedure. The increasing availability of publicly available large-scale data sources (e.g. ImageNet [14]) and GPU compute power mitigated the initial resource problems: With AlexNet winning the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [12], ongoing research has made CNNs the de facto standard for a large number of machine-learning tasks in computer vision, Natural Language Processing (NLP) and other disciplines. Today, CNN-based state-of-the-art architectures in image classification, text recognition and audio or video analysis outperform traditional algorithms by an order of magnitude.

Convolutions

The characteristic operation of a CNN is the convolution of a given input with one or more filter kernel(s), commonly followed by a non-linear activation function and a downsampling operation. It is worth noting that the term convolution is misleading, since the mathematical operation that actually is performed when using the term convolution is the cross-correlation. Given a matrix M with indices a and b , the convolution of M with a kernel C is defined as

$$M'_{a,b} = (M * C)_{a,b} = \sum_j \sum_k M_{a-j,b-k} \cdot C_{j,k} , \quad (2.1)$$

where j and k indicate the relative dimensions of the kernel C with respect to its center [13]. Equation 2.1 can be visualized as sliding the filter kernel across the input matrix and multiplying each kernel-position with the underlying input to calculate the values of the convoluted matrix M' . As the actual cross-correlation operator simply uses a convolution with rotated kernel, it holds that convolution equals cross-correlation if $C = \text{rot}_{180^\circ}\{C\}$. Following the conventions used in machine-learning frameworks and literature [13], the term convolution will henceforth denote the following operator:

$$M'_{a,b} = (M * C)_{a,b} = \sum_j \sum_k M_{a+j,b+k} \cdot C_{j,k} . \quad (2.2)$$

By using convolutions to process the input, CNNs make use of sparse interactions between neighbouring grid cells, which in computer vision often share semantic meaning. Convolutions enable a CNN to extract features of different granularity as the layer depth increases and reduce the number of parameters in the network, thus saving computation time and memory consumption while increasing statistical efficiency. Furthermore, the convolution with multiple filter kernels yields different feature-maps that each show a distinct representation of the input [13].

Downsampling

Downsampling operations are usually applied after convolution and non-linearity to reduce dimension within the network. Amongst the most common downsampling techniques is the pooling operation, which reduces the size of the feature-maps by substituting the convolution's output at a certain point with a statistical summary of a region around that point, usually by averaging or choosing the maximum. Pooling increases the network's robustness to changes in the input and helps to prevent overfitting, which both are desirable properties for many applications of neural networks [13].

2.1.1 Relevant Classifiers

Due to the variety of available CNN architectures, the following sections focus solely on the approaches and models that will be used throughout this work. More specifically, the two classifiers that will be used by the NIMA architecture in the following sections will be introduced.

VGG-16

The VGG-network was created by the Visual Geometry Group (VGG) of the University of Oxford as an entry to 2014’s ILSVRC and was awarded the first and second price in localization and classification tasks, respectively [3].

The characteristic parameters of the VGG-architecture are its depth, often given as part of the model name (e.g. VGG-16), and its kernel size, which is set to the very small receptive field of only 3×3 — the smallest size possible to still allow for distinction between center, up, down, left and right. The network uses a resolution-preserving padding while convolving and applies spatial pooling by appending max-pooling layers to the convolutional layers. The max-pooling operations halve the size of the feature-maps as they apply a 2×2 kernel with stride two [3]. After the convolutional layers, the architecture features three fully-connected layers with 2×4096 and 1×1000 neurons, respectively, and a soft-max activation layer. For the convolutional layers, the Rectified Linear Unit (ReLU) activation is used. Table 2.1 shows three different architectures proposed in [3] and their individual layering.

VGG 16 A	conv3-64 conv3-64	mp	conv3-128 conv3-128	conv3-256 conv3-256 conv1-256	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv1-512	mp	FC 4096	FC 4096	FC 1000
VGG 16 B	conv3-64 conv3-64		conv3-128 conv3-128								
VGG 19	conv3-64 conv3-64		conv3-128 conv3-128								

Table 2.1: Three of the different VGG configurations proposed in [3]. The versions with 11 and 13 weights layers and the activation functions were omitted for brevity. Adapted from [3].

The network was trained with random-cropped RGB input images of size 224×224 and mini-batch gradient descent with momentum. To regularize the training, the authors employed an L_2 weight decay and a dropout of 50% for the first two fully connected layers. Simonyan *et al.* [3] showed

that classification accuracy could be improved significantly by increasing the network's depth to 16 - 19 layers, while still being able to generalize well to other datasets. The VGG-architecture outperformed 2014 competition by a large margin ($\Delta \approx 4.2\%$ top-1 val. error, [3]) and thus showed that deepening convolutional networks is a potent means of increasing model potential.

However, the high number of layers leads to an increase in parameter count. As a result, the VGG-models are relatively large (cf. Tab. 2.2), and feature several million trainable parameters for each configuration. While this may not be a problem when using large-scale GPU clusters or environments with constant specification like desktop workstations, it might complicate VGG usage in mobile or high-performance applications.

Architecture	VGG-16A	VGG-16B	VGG-19
Parameter Count	134	138	144

Table 2.2: The VGG-models' parameter counts, given in million parameters. For brevity, only the models described in Table 2.1 are mentioned.

MobileNet

On the contrary, the MobileNet architecture is efficient and lightweight. It is specifically designed to be adaptable to hardware limitations or timing constraints and therefore is well-suited for embedded system applications or mobile devices, where low latency is to be achieved with a relatively small amount of compute power [17].

The main feature of MobileNets are depthwise separable convolutions, which originate from the idea of refactoring regular convolutions. In depthwise separable convolutions, a single filter is applied to each channel of the layer input before then cumulating the filtered results with a 1×1 convolution along the channel axis. Given a square input feature-map F of size D_F with M channels, i.e. with dimensions $D_F \times D_F \times M$, and a square kernel \hat{K} with dimensions $D_k \times D_k \times M$, Howard *et al.* [17] express the depthwise convolution as

$$\hat{\mathbf{G}}_{k,l,m} = \sum_{i,j} \hat{\mathbf{K}}_{i,j,m} \cdot \mathbf{F}_{k+i-1,l+j-1,m} . \quad (2.3)$$

Equation 2.3 can be visualized as applying the m_{th} channel of \hat{K} to the m_{th} channel of F to produce the m_{th} channel of \hat{G} . This is extremely effective in terms of compute demand: A standard convolution on a square-sized feature-map costs $D_F^2 \cdot M \cdot N \cdot D_k^2$ operations, with N being the number

of output channels after the convolution. When using depthwise convolutions, i.e. when applying one filter per input channel, the relation between the number of output channels and the kernel size is broken. The cost of 1×1 -convoluting along the channel axis obviously is $D_F^2 \cdot M \cdot N$. Summing depthwise and channelwise convolutional costs and comparing them to the standard convolution results in a total cost reduction of $\frac{1}{N} + \frac{1}{D_k^2}$ [17]. The MobileNet architecture furthermore employs two hyperparameters, α and ρ , which can be used to scale the network to the deployment context at hand. By multiplying the number of in- and output channels of each layer by a factor $\alpha \in (0, 1]$, the network can be made more compact at the cost of removing channels, and therefore expressiveness. The parameter $\rho \in (0, 1]$ controls the network's input resolution and subsequently that of all consecutive network layers [17]. The full MobileNet body architecture as proposed in [17] can be found in appendix A.

The employed techniques make MobileNets very efficient and enable them to be trained with smaller datasets and less regularization, as less-parametrized networks are not as prone to overfitting. Although MobileNets are compact and only have a low number of parameters, they perform remarkably well when compared to much larger models: On the ImageNet dataset, MobileNet outperformed GoogLeNet by 0.8% while only using roughly one third of Mult-Adds. MobileNet performs only 0.9% worse than VGG-16, although it uses 27 times less computations and roughly 3% of VGG-16 parameters [17]. However, the limited parameter count might cause MobileNet architectures to struggle with datasets of large diversity and rich features.

2.2 Fully Convolutional Networks

The second class of neural networks used by the method proposed in this work is that of Fully Convolutional Networks (FCNs). During recent years, FCNs have achieved state-of-the-art results in scene understanding [25], object detection [24], semantic segmentation [18] and a variety of other tasks. This section will focus on the particular case of Context Aggregation Networks (CANs).

2.2.1 Context Aggregation Networks

As described in section 2.1.1, image classifier architectures often are trained to predict a single class label per input image. However, in the context of

semantic segmentation, the employed networks oftentimes are required to predict a label for each pixel in the input image. This is also known as dense-prediction [19], i.e. the in- and output of the network have the same dimension. A common way to achieve dense prediction (used by e.g. UNet [23]) is to downsample the image, process the downsampled version, and then upsample the results to retrieve the input's original resolution.

The specific architecture presented here is called the “multi-scale context aggregation network” [22] and will henceforth be termed CAN. CANs use a different approach on dense prediction: By inserting dilations into the convolutional kernels, the network can aggregate long-range contextual information without the need to compress the image. Thus, CANs can save on down- and upsampling operations by applying adequate zero-padding, and the internal representations of the input have the same resolution as the original image. Yu *et al.* [22] define the convolutional operation as

$$(F * k)(p) = \sum_{s+t=p} F(s)k(t) , \quad (2.4)$$

which equals equation 2.1 when $p = (a, b)$. Moreover, they define the l -dilated convolution, denoted with the operator $*_l$, as

$$(F *_l k)(p) = \sum_{s+l\cdot t=p} F(s)k(t) , \quad (2.5)$$

where l is the dilation factor between adjacent kernel items. Summation over the index $s + l \cdot t$ causes the filter to skip certain items. A filter kernel dilated by the factor l can be visualized as the regular filter kernel with $l - 1$ zeros inserted horizontally and vertically between filter items. However, [22] stresses that no new filters are constructed when using dilated convolutions — instead, the original kernel's entries are used at different locations of the image. When $l = 1$, the dilated convolution is equivalent to the standard convolution.

The main advantage of using dilated convolutions is the exponential increase of the receptive field. The receptive field of an element p in the feature-map F_{i+1} at the convolutional layer $i + 1$ is defined as the set of all elements in the input feature-map F_0 that modify the value of $F_{i+1}(p)$. The size of the receptive field with respect to layer depth i is $(2i + 1) \times (2i + 1)$ for regular, non-dilated convolutions, as each convolution simply applies the same filter kernel to process the previous layer. When the dilation factor is exponentially increasing with layer depth, i.e. the convolution at layer i becomes the operation $F_{i+1} = F_i *_{2^i} k_i$, the receptive field for each element in F_{i+1} becomes $(2^{i+2} - 1) \times (2^{i+2} - 1)$. Figure 2.1 shows the comparison between receptive fields when applying regular and dilated convolutions.

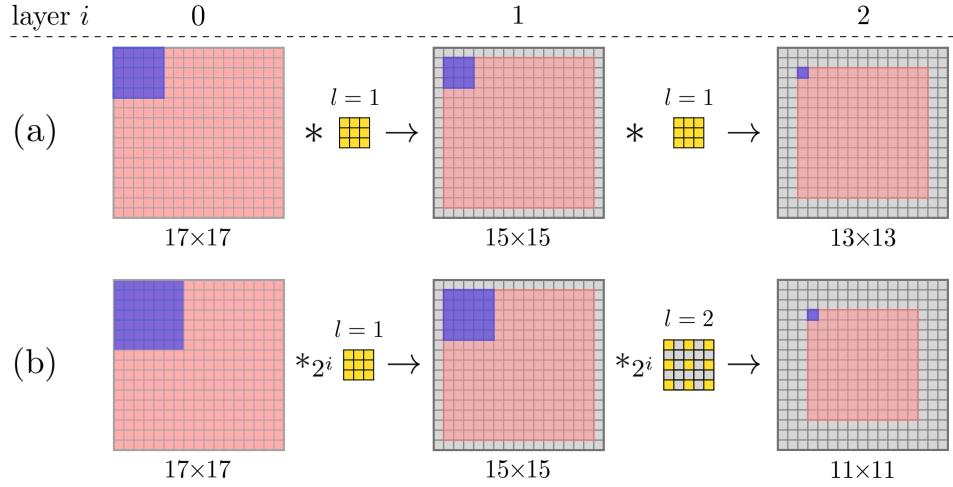


Figure 2.1: Comparison between (a) regular and (b) dilated convolution. During the forward pass, the resulting values (red, shapes annotated below) are padded with zeros (grey) to preserve resolution. The blue squares indicate the receptive fields and are best understood when examining the figure backwards: The receptive field of the first element at layer 2 consists of the elements in layer 0 that created said element. The dilation in the filter kernels causes the receptive field to grow exponentially w.r.t layer depth i .

The exponential growth of the receptive field allows the CAN to aggregate multi-scale views of contextual information in the input image. A particularly intriguing observation is, that albeit the receptive field grows exponentially, the number of layer parameters grows only linearly, as the kernel parameters themselves are left untouched and only their direct application changes. In practical applications, it is useful to increase the dilation parameter l in such a way that the receptive field grows until it encompasses the entire input image. In the network's last layer, [22] use a linear transformation — a 1×1 convolution along the channel axis, with no non-linearity — to accumulate the contextual information that the network encoded in the different feature maps. A detailed description of the architecture proposed in [22] can be found in appendix A.

2.3 Metrics and Loss Functions

This section aims to introduce the metrics and loss functions that will be used throughout the course of this work and elaborate on their usage, advantages and drawbacks. Typical loss functions for image-based problems are the Mean Absolute Error (MAE) and the Mean Squared Error (MSE), which, in a pixel-wise setting with reference \hat{y} and manipulated image y , are defined as

$$\mathcal{L}_{\text{MAE}}(\hat{y}, y) = \frac{1}{N} \sum_i |\hat{y}_i - y_i| \quad \text{and} \quad (2.6)$$

$$\mathcal{L}_{\text{MSE}}(\hat{y}, y) = \frac{1}{N} \sum_i (\hat{y}_i - y_i)^2 . \quad (2.7)$$

According to [27], MAE outperforms MSE in dense prediction tasks like image enhancement, as MSE, due to its tolerance to small errors, is prone to yielding blurry images. Although both of these losses are de-facto standard and produce good results in many applications, they neither do consider the visual aspect of the enhanced image nor correlate well with human perception of image quality [33]. Therefore, the following sections will introduce perceptually-motivated metrics.

2.3.1 Earth Mover’s Distance

The Earth Mover’s Distance (EMD) (also known as the Wasserstein metric) is a method to quantize the distance between two probability distributions, i.e. the minimal cost it takes to transform one distribution into the other. One can visualize the EMD by imagining two piles of dirt with the same integral (the same amount of dirt). The EMD then is defined as the minimal cost it takes to transform one pile of dirt into the other, i.e. to transfer the shape of pile A into that of pile B [26] with respect to the amount of dirt to be shifted and the distance by which it is moved. In the context of this work, it will be used to measure the distance between class distributions in a multi-class categorization problem. The EMD addresses the drawback that comes with the cross-entropy loss, another widely used loss function in classification:

$$\sum_{i=1}^N -p_{s_i} \log(\hat{p}_{s_i}) . \quad (2.8)$$

In the above equation, p_{s_i} is 1 if, and only if, sample s_i belongs to category p , while \hat{p}_{s_i} is the model’s class prediction for the sample s_i . As such, this loss is useful for maximizing network prediction accuracy. However, the soft-max cross-entropy loss does not consider inter-class relations and therefore cannot distinguish between weak and hard negatives — informally speaking, it lacks the ability to express “how wrong” a prediction is.

In the context of predicting image beauty, the classes (commonly referred to as score buckets [8], [10]) are inherently ordered: It is only logical to assume an image that received the rating 5 to be more beautiful than an image that was rated 4, or even 3. As previously mentioned, the soft-max cross-entropy loss cannot distinguish whether the network’s prediction just slightly missed the correct score bucket or whether it showed severe deviations from the ground truth. Hou *et al.* [26] show that EMD-based loss functions encode this ordering by penalizing mis-classifications with respect to prediction distances and thus outperform other methods. When given N intrinsically ordered image quality classes $s_1 < \dots < s_N$, the EMD between two probability mass functions \mathbf{p} and $\hat{\mathbf{p}}$ is defined as

$$\text{EMD}(\mathbf{p}, \hat{\mathbf{p}}) = \left(\frac{1}{N} \sum_{k=1}^N |\text{CDF}_{\mathbf{p}}(k) - \text{CDF}_{\hat{\mathbf{p}}}(k)|^r \right)^{1/r}, \quad (2.9)$$

with $\text{CDF}_{\mathbf{p}}(k)$ being the cumulative distribution function $\sum_{i=1}^k \mathbf{p}_{s_i}$ [31]. The aforementioned requirement for both distributions to have the same integral usually is guaranteed by feeding the probability distributions through a soft-max layer and thus ensure that $\sum_i \mathbf{d}_i = 1$ for all distributions \mathbf{d} . According to [8] and [10], $r = 2$ facilitates easier optimization, as it penalizes the Euclidean distance between distributions.

2.3.2 Peak Signal-to-Noise Ratio

Peak Signal-to-Noise Ratio (PSNR) is a measure for the proportion between the maximum power of a signal and the power of corrupting noise that negatively affects the signal’s fidelity. As signals tend to have a widespread amplitude, PSNR commonly is expressed on a logarithmic scale and given in decibel. The PSNR of a monochrome image I is defined as

$$\text{PSNR} = 10 \log_{10} \left(\frac{\text{MAX}_I^2}{\text{MSE}} \right), \quad (2.10)$$

where MSE is the Mean Squared Error introduced in equation 2.7 and MAX_I is the highest possible pixel value, i.e. $2^{bits} - 1$ [28]. For multi-channel (e.g. RGB-) images, PSNR uses the averaged per-channel MSE. Note that when the original and the noisy image are identical, i.e. in the total absence of noise, the PSNR is undefined, as it holds that $\text{MSE}(\hat{y}, y) = 0$ if $\hat{y} = y$.

Huynh-Thu and Ghanbari [28] showed that PSNR is a valid metric for evaluating the technical quality of an image after manipulation (e.g. compression), as it is sensitive to quality changes. Yet, [28] also shows that this sensitivity decreases drastically when the image’s codec or content is changed. Therefore, PSNR is well-suited for comparing quality changes within the same image domain, but inadequate for comparisons between images with differing codec or content. A figure depicting the resulting PSNR values for different compression strengths can be found in appendix B.

2.3.3 Structural Similarity

The Structural Similarity (SSIM) is a full-reference metric to rate the perceived quality of digital images. Established after Weber's law [30], according to which the human visual system's (HVS) reactivity to change increases in regions with consistent lighting and coloring, the structural similarity index [6] rates image perception according to image luminance, contrast and structure.

SSIM is computed over various windows at different locations of the image. As mentioned above, it considers window luminance, contrast and structure. If given windows x and y both have the size $N \times N$, the luminance function $l(x, y)$ is defined as a function of the mean intensity of the respective window's pixel values and hence can be expressed as

$$\mu_k = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N k_{ij} \quad \text{and} \quad (2.11)$$

$$l(x, y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}, \quad (2.12)$$

where c_i is used to denote a small constant that ensures the stability of the division [6]. Further, the difference in window contrast is measured by the contrast function $c(x, y)$, which uses the standard deviation σ to estimate contrast deviation:

$$\sigma_k = \left(\frac{1}{N-1} \sum_{i=1}^N (k_i - \mu_k)^2 \right)^{1/2} \quad \text{and} \quad (2.13)$$

$$c(x, y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}. \quad (2.14)$$

Lastly, the structure comparison function $s(x, y)$ is defined as a function of the covariance σ_{xy} between the windows:

$$s(x, y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}. \quad (2.15)$$

Combining the previous equations into one term yields the overall similarity measure $\text{SSIM}(x, y) = [l(x, y)]^\alpha \cdot [c(x, y)]^\beta \cdot [s(x, y)]^\gamma$ with $\alpha, \beta, \gamma > 0$ being used to weight the respective components. Setting $\alpha = \beta = \gamma = 1$ and $c_3 = c_2/2$, Wang *et al.* [6] express the SSIM index as

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}. \quad (2.16)$$

The values produced by SSIM reside in the interval $[-1, 1]$, where the maximum value 1 can only be achieved in case of truly identical pictures. For practical image evaluation, the Mean Structural Similarity (MSSIM) is computed as the average

of the image contents at the j -th local windows \mathbf{x}_j , \mathbf{y}_j of the original and distorted image \mathbf{X} and \mathbf{Y} :

$$\text{MSSIM}(\mathbf{X}, \mathbf{Y}) = \frac{1}{M} \sum_{j=1}^M \text{SSIM}(\mathbf{x}_j, \mathbf{y}_j) \quad . \quad (2.17)$$

Following the conventions in literature, all further mentions of SSIM refer to the previously defined MSSIM index. Since the SSIM is designed to imitate the HVS rather than naïvely evaluating differences in pixel values, SSIM scores correlate well with human judgement. To illustrate this, Figure 2.2 shows the comparison between a reference image and several distorted versions that all share the same MSE.



Figure 2.2: Comparison of a reference image (a) with several distorted versions (b) to (f), all with $\text{MSE} = 210$. (b) $\text{SSIM} = 0.9168$ (c) $\text{SSIM} = 0.9900$ (d) $\text{SSIM} = 0.6949$ (e) $\text{SSIM} = 0.7052$ (f) $\text{SSIM} = 0.7748$. Figure adapted from [6].

3

Related Work

With recent advances in computer vision research, image enhancement methods using neural networks have proven to be capable of outperforming traditional algorithms in terms of speed and efficiency, often so by orders of magnitude. While the previous chapter concentrated on familiarizing different architectures, metrics and loss functions, this chapter aims to give an overview over the work associated with this thesis. As described previously, the goal of this work is to create an automated image enhancement pipeline that produces the perceptually most appealing version of the image. To this end, two main components are used: an enhancement model, to apply filters onto the original source image, and a quality assessor, to rate the enhancement’s outcome. This chapter presents related work and highlights the commonalities and differences between the presented approach and other concepts.

3.1 On Image Assessment

Automatic assessment of image quality is a profitable task, as it has a variety of applications in business and other contexts. The fact that human judgement is highly interwoven with visual perception can informally be expressed as “we believe what we see” — the popular tourism website Tripadvisor reports a direct correlation between customer engagement, booking inquiries and the images that were uploaded by the property host [7]. The website Idealo further exploits this by using automatic quality assessment to rank hotel images in order to propose superior images to the users and therefore increase the likelihood of a booking [5].

When referring to image assessment, it is reasonable to distinguish between measuring the technical and assessing the aesthetic quality of an image. Technical image assessment, e.g. the evaluation of noise, blur or compression artefacts, is typically achieved by using full-reference approaches like SSIM or PSNR (cf. section 2.3) with the original gold-standard image, to achieve a pixel-wise comparison ([6, 8, 28]). Assessing the aesthetic quality of a picture is a more subtle task, since the human judgement takes into account image semantics, content, emotions, style of photography and many other factors when judging the perceptual image appeal. Aesthetic quality assessment thus usually is conducted in a no-reference setting, since there is no reference image to compare to.

With the increasing availability of human-rated image datasets ([34, 36–38]), several approaches to aesthetic quality assessment based on neural networks have been proposed in literature and research. The main advantage of using neural networks (CNNs in particular) trained on rich image datasets for quality prediction is the circumvention of traditional, hand-crafted image features like color harmonies, edge distribution or foreground-background composition [44]. Manually crafting these features is labour intensive and inefficient even with modern algorithms, as each image needs to be processed and annotated before it can be used for quality assessment.

3.1.1 AVA: Visual Aesthetic Assessment

One of the most popular datasets for aesthetic image assessment is the AVA dataset (AVA: A Large-Scale Database for Aesthetic Visual Analysis, [34]), which was introduced by Murray *et al.* and consists of 255.509 JPEG images. The dataset was gathered from DPChallenge¹, a website that hosts pictures of professional and amateur photographers and whose members vote for the artistry and aesthetics of the uploaded images [67]. Participants can cast image votes in the range [1, 10], with one denoting the least and ten denoting the most beautiful score category. Instead of categorizing the pictures into classes, as in many other ML datasets ([14, 64, 66]), AVA uses the community votes as aesthetic annotations, with an average of around 210 votes per image. The image aesthetic scores can then be computed as the mean across the user votes. Additionally, AVA provides 963 so-called “challenge-tags” which correspond to the photography contest where the respective image was uploaded, and 66 textual tags, describing the semantic content of the image. Every sample in the dataset is associated with exactly one challenge tag and up to two semantic tags. AVA’s images range from blurry, low-quality snapshots over artistic imagery and advertisements to high-quality photography. Figure 3.1 illustrates this diversity with selected examples from the AVA dataset.

¹DPChallenge - A digital photography contest, www.dpchallenge.com, retrieved on 30.01.2020



Figure 3.1: Selected images from the AVA dataset, with the respective average rating annotated below. Challenge and semantic tags, respectively, are (a) Advertisement, Humorous/Animals (b) Travel, Cityscape/Architecture (c) Night-Shot, Astrophotography (d) On the road, None. Images from [34].

3.1.2 Neural Image Assessment

As previously mentioned, CNNs bypass manual feature engineering by learning the salient image features that are indicative of favourable aesthetic ratings from training on large image corpora. Kang *et al.* [39] combine feature-learning and regression to prove that high-level CNN features extracted from 32×32 image crops can be used to reliably predict image quality. Although this approach is refined by Bosse *et al.* [40] with a deeper network and higher accuracy, the small input sizes require the predicted score to be aggregated across the image, which is particularly problematic in aesthetic assessment, since semantic information is usually not distributed evenly throughout the image. Lu *et al.* [41] further facilitate the use of deep networks and avoid the aforementioned problem by resizing the input image to 256×256 and randomly cropping 224×224 windows, which additionally serves as data augmentation. Since the spatial structure of the image content is altered by this transformation, their proposed approach uses local and global image representations in two parallel networks and gathers the information encoded in the resulting feature-maps by using fully-connected layers. This approach reliably rates aesthetic quality on the AVA dataset [41]. However, Lu *et al.* formulate the problem as a binary classification task and therefore only categorize to low and high aesthetics. In [42], the authors build upon ideas described in [43] and introduce a regression framework to predict a histogram of AVA ratings. Kong *et al.* [45] formulate aesthetic assessment as a ranking problem and propose a content-adaptive model trained on AVA for rank correlation. A more recent approach by Fu *et al.* [46] uses a combination of pre-trained CNNs and Support Vector Machines (SVMs) to assess image beauty and thus completely eliminates the need for specific model training. While this approach seems promising, the employed architecture, similar to the approach by Lu *et al.* [41], only categorizes to low and high aesthetics. If the quality prediction used in this thesis was binary, the enhancement pipeline would stop working upon reaching the higher class. Evidently, for the approach presented in this thesis, continuous score prediction is indispensable.

The image quality assessor NIMA (Neural Image Assessment) by Talebi and Milanfar [8] meets this requirement and combines the advantages of the aforementioned approaches to achieve state-of-the-art accuracy in predicting technical and aesthetic image quality. The approach combines a CNN with a dense layer, as in [41], and likewise adapts rescaling to 256px, followed by random-cropping to patches of size 224×224. Additionally, the authors introduce a random horizontal flip of the cropped images. Contrary to [41] and [46] categorizing into low and high, the classifier's last layer is replaced by a fully-connected layer with ten nodes and a soft-max activation function, in order to enable the network to predict a discrete probability distribution for score buckets one to ten, which is then interpreted as aesthetic score distribution. NIMA thus predicts the distribution of AVA ratings instead of regressing to the mean score. The image's aesthetic score is then calculated as the distribution mean $\mu = \sum_i x_i p(x_i)$ of all samples x_i and their probabilities $p(x_i)$, leading to a high correlation between NIMA predictions and humanly perceived image appeal [8]. Furthermore, NIMA allows to assess so-called “image conventionality” by calculating the standard deviation of the predicted distribution. As stated in [8], predictions with high standard deviations indicate high unconventionality in image content, as the image either attracts or repels the viewers. Figure 3.2 illustrates this with selected pictures and their respective mean scores and standard deviations.

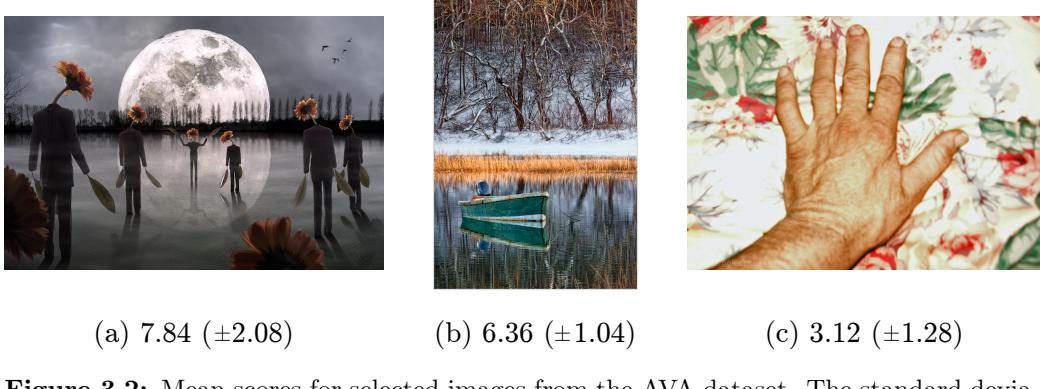


Figure 3.2: Mean scores for selected images from the AVA dataset. The standard deviation is given in parenthesis. Image (a) receives a high rating, while its content is considered relatively unconventional. Images (b) and (c) receive high and low ratings, respectively, and the image content is assessed as comparatively conventional. Figure adapted from [8].

Using the Earth Mover’s Distance (cf. section 2.3.1) as a loss function, the deviation between the predicted distribution and the AVA ground truth is back-propagated through the network by Stochastic Gradient Descent (SGD). Halebi and Milanfar use CNNs that were pretrained on the ImageNet dataset [14] and both (re-)train the classifier weights and the dense layer. According to [8], this is the only method besides [45] that achieves correlation with AVA ratings while predicting true distributions instead of classifying into high and low.

3.2 On Image Enhancement

Image enhancement has been an active field of research for the past decades, leading to the development of sophisticated processing operators and to an increase in their efficiency and speed. Most enhancement operators aim to increase the practicality and fidelity of the image, e.g. for displaying or further processing. While it should be noted that image enhancement does not increase the amount of information contained in the image data [29], many applications can greatly benefit from enhanced images. Examples of this are low-resolution imaging [35] or noise-removal [61], where enhanced images are more interpretable than their original counterparts. While these examples improve images in terms of technical quality, this chapter will focus on perceptual image enhancement — the kind of post-processing operations that, for instance, a photographer would apply to unedited image data in order to increase the perceptual image appeal.

3.2.1 MIT-Adobe FiveK

A commonly used dataset in perceptual image enhancement is the MIT-Adobe FiveK dataset, which was introduced in the 2011 publication “Learning Photographic Global Tonal Adjustment with a Database of Input / Output Image Pairs” [65]. The dataset consists of 5.000 high-resolution DNG images that were captured with single-lens reflex cameras and manually annotated with camera meta-data (e.g. manufacturer, model, focal width, etc.) and image meta-data (e.g. lightning conditions, daytime, etc.). Contrary to AVA (cf. section 3.1.1), where the pictures stem from a photography-dedicated website and therefore naturally lean towards photographic artistry, the MIT-Adobe FiveK dataset is not focussed on aesthetically pleasing images and thus covers a broader range of photographic styles, scenes and lightning conditions. Moreover, the dataset contains five additional versions of each picture that were enhanced by professional photographers, resulting in a total of 30.000 images, or five sets of 5.000 enhanced images that extend the original dataset. The differences between the experts’ enhancements range from very subtle to clearly visible, depending on the original image and the retoucher’s preferences. Figure 3.3 shows an example where the enhancement style differs significantly: While experts A and B manipulated the image towards a sunset mood, experts C and D enhanced the brightness to achieve a daylight setting, and expert E enhanced contrast, exposure and colors to make the image more vivid.

3.2.2 Learning Image Enhancement

In recent work, it has been shown that deep neural networks perform notably well when tested on the task of perceptual image enhancement ([4, 10, 54, 57]). Xu *et al.* use a deep convolutional network to learn edge-aware filters by creating a



Figure 3.3: The original image and its enhanced versions by the respective experts [65].

gradient field and then use gradient integration to construct the enhanced image [55]. While the approach performs well for the family of edge-aware filters, it is not easily portable to other enhancement operators that do not use gradient-based processing. In [56], the authors use a combination of feature extraction and image segmentation to achieve automatic photo enhancement with content-adaptive behaviour. Albeit their approach produces good results, it operates on a per-pixel-basis, incorporating semantic information solely by manual feature crafting. This makes the technique invariant to image composition and heavyweight, and therefore inapt for this work. Building upon [60], Santurkar *et al.* recently proposed a novel approach relying on adversarial training of a classifier architecture to perform a number of image manipulation tasks, such as in-painting, super-resolution and interactive image manipulation [58] and show the connection between salient image features and robustness of adversarially trained models. As adversarial examples typically tend to show blurry lines and inhomogeneous lightning and contrast [59], the human eye perceives them as not visually pleasing, rendering the approach inadequate for the goal of this thesis.

In other related work, Hu *et al.* [68] combine Generative Adversarial Networks (GANs) and Reinforcement Learning (RL) to create an enhancement pipeline that replicates the photo retouching style of a photographer. In their work, the GAN-generator tries to enhance images in order to look like the work of a professional retoucher, while the discriminating part of the network has to detect whether the presented image is fraudulent or real. The setting of filter parameters and their succession is controlled by a RL agent. Although GANs and RL algorithms are known to be unstable and hard to train ([69–71]), the approach bears the intriguing property of enhancing the image in a white-box fashion: The RL agent, which determines the parameters that lead to the enhanced image, is driven by a policy whose steps can be traced and visualized, rendering the enhancement outcome transparent. However, the GAN is trained on image pairs

from the MIT-Adobe FiveK dataset [65], consisting of the original image and a version that has been manipulated by one of the five experts. Therefore, the image enhancement process is strongly biased by this expert’s personal preferences (cf. Fig. 3.3). While this may not necessarily be a drawback, it implies a significant bias for the enhancement’s outcome which may not necessarily correlate with the user’s preferences.

Recent work by Chen *et al.* [4] breaks the paradigm of creating a filter or network dedicated to, and excelling at, a single task. In the publication “Fast Image Processing With Fully-Convolutional Networks” [4], they propose a context-aware image manipulation architecture that relies on the principle of Context Aggregation Networks (cf. section 2.2.1) and outperforms all previously mentioned approaches in terms of accuracy, compactness and speed. The work was inspired by the downsample-evaluate-upsample method, which is commonly used in classic image processing: The processing operator is evaluated on a downsampled version of the image before upsampling the processed image to its original resolution, and thus approximating the operator’s effect on the full-sized image in fractional compute time. This otherwise very versatile approach suffers from a severe limitation, as it is not able to recover spatial transformations of the processed, downsampled image (e.g. perspective correction) when upsampling. In [4], the idea of approximating the operator’s effect on the full-sized image is adapted, albeit without changing image resolution, and hence eliminating this issue. The compact representation of the network (roughly 37.000 parameters) enables model inference at interactive rates and in constant time (190 ms per forward pass on an NVIDIA Titan GPU). The approach beats classical algorithms by an order of magnitude: Particularly demanding operators like Rudin-Osher-Fatemi [61] or image dehazing [62, 63] exhibit run-times in the range of up to 20 seconds on modern computers [4]. An exhaustive listing of operator performance can be found in appendix B.

By feeding pairs of original and manipulated images \mathbf{I} and $\hat{\mathbf{I}}$ to the CAN, Chen *et al.* accomplish an approximation \hat{f} of the original operator f on the full-scale image. Interestingly, this is achieved by using a rather simple loss function, the so-called image-space regression loss

$$\ell(K, B) = \sum_i \frac{1}{N_i} \| \hat{f}(\mathbf{I}_i; K, B) - f(\mathbf{I}_i) \|^2 , \quad (3.1)$$

where K and B are the network weights and biases, respectively, and N_i is the number of pixels in the image i [4]. With $\| \cdot \|^2$ being the Euclidean norm, $\ell(K, B)$ can be interpreted as the mean-squared error in the three-dimensional RGB color space across the dataset. Surprisingly, [4] report this loss to outperform MAE and other, more sophisticated loss functions, and attribute this to the fact that most of the image processing operators approximated by the network are not semantic in nature.

The approach is tested and evaluated on ten different image processing operators that can be categorized into four groups of filter families: edge-aware filtering (local Laplacian filtering, multi-scale tone manipulation), image restoration (Rudin-Osher-Fatemi, TL-V, dark channel dehazing, non-local dehazing), detail suppression (relative total variation, L_0 -smoothing) and artistic filters (style transfer, pencil-drawing). Figure 3.4 shows the network's approximation of selected operators. Chen *et al.* [4] stress that all of the images in Figure 3.4 were created with the same architecture and hence show that their approach generalizes well to a variety of image manipulation tasks. The above characteristics suggest that Chen's approach is well-suited for the enhancement pipeline used in this thesis.



Figure 3.4: Selected operators approximated by [4]. The input images in the top row were converted to the manipulated images below using the CAN trained on the respective operator. Figure adapted from [4].

3.2.3 Learning Perceptual Image Enhancement

The field of research most closely associated with this thesis is that of learned perceptual image enhancement, which has received ample attention in recent publications, particularly so after the emergence of neural image assessors with well-defined derivatives. In learned perceptual image enhancement, a neural network is used to enhance an image before judging the enhancement's outcome with a quality assessor, e.g. NIMA (cf. section 3.1). The fact that neural networks themselves are differentiable implies that, when adding a perceptually motivated loss

function to the network’s output, the loss can be backpropagated through the network. This information can then be used to either manipulate the image directly or adjust the network’s parameters in a way that minimizes the loss and therefore increases image fidelity.

In [10], Halebi and Milanfar use their NIMA model as a loss function for an enhancement pipeline in order to guide the parameters of a local Laplacian filter. The filter’s result is rated by NIMA and the network optimizes the NIMA score by fine-tuning the filter parameters, yielding results of superior visual quality. Figure 3.5 suggests that the NIMA score is very sensitive to transformations introduced by detail boosting and color manipulation and therefore is well-suited for guiding the filter parameters towards the ideal setting.



Figure 3.5: Original and enhanced images with their respective NIMA scores. The filter used to enhance the images is a form of the local Laplacian filter (LLF) proposed in [21]. Figure adapted from [8].

Similarly to [10], [20] and [27] use perceptually motivated loss functions (e.g. SSIM, cf. section 2.3.3) to guide networks towards learning filter representations that achieve high aesthetic scores. Although this might initially sound similar to the goal of this thesis, it in fact is a fundamentally different approach: [8], [10], [20] and [27] all use the loss to adjust filter parameters or fine-tune operator hyperparameters, i.e. change the filter’s representation during the optimization process to reach the best combination *for a single image* and with a single, given filter. Evidently, constricting oneself to one operator and re-learning the operator’s parameters for each image is not very flexible. In the approach presented in this work, however, the filters are fixed, as they will be learned by the CAN architecture proposed in [4]. After the initial CAN training phase, the weights remain untouched — once the network learned to represent the operators, the filter representations do not change any further. This bears the advantage of not having to re-learn the filter parameters for each separate image, but instead enables the approach to quickly infer the best possible filter setting with only few passes through the architecture. The proposed enhancement pipeline and its realization shall be discussed in the following chapter.

4

Method

As shown in the previous chapter, automated image enhancement is a hot topic in current computer vision research. Albeit many of the approaches presented in chapter 3 achieve promising results, only few combine compactness, speed and generality while considering user preferences. This thesis aims to introduce a pipeline that automates the time-consuming task of professional image enhancement in tolerable compute time, while giving users the freedom to apply their personal preferences in terms of optimization strength and filter presets.

4.1 Problem Definition

Let $F = \{f_1, f_2, \dots, f_n\}$, $n \in \mathbb{N}$ be a set of image filters f . Each filter f_i realizes a different post-processing effect that can be applied with the intensity $k_i \in \mathbb{R}$. Let therefore K be the set of all filter intensities $\{k_1, k_2, \dots, k_n\}$. Let further Ω_I be a set of unedited images, and $\Omega_{\hat{I}}$ a set of enhanced images. We define $\psi_F : K, \Omega_I \mapsto \Omega_{\hat{I}}$ to be a function that maps each filter $f_i \in F$ and its respective intensity $k_i \in K$ onto an unedited image $I \in \Omega_I$ to create a manipulated image $\hat{I} \in \Omega_{\hat{I}}$. The goal of the image optimization is to find an intensity combination $K' = \{k'_1, k'_2, \dots, k'_n\}$ that maximizes the metric M . This is formally expressed as

$$K' = \underset{k_1, k_2, \dots, k_n}{\operatorname{argmax}} M(\psi_F(K, I)) . \quad (4.1)$$

In the concrete case of this thesis, M is a metric measuring the visual aesthetics of the manipulated image. Therefore, K' is the combination of filter settings that produces the perceptually most appealing version of the original image I .

4.2 Approach

The presented approach aims to solve the above problem of maximizing image aesthetics with an automatic image enhancement pipeline consisting of two main components: an image manipulation network (corresponding to ψ_F) and a quality prediction network (corresponding to M). To optimize an unedited, original picture, the image is first passed through the manipulation network, where it is enhanced with an initial combination of filter intensities. Subsequently, the enhancement's outcome is rated by the quality prediction network, assessing the aesthetic of the resulting image and assigning a score to the result. Using this score, a loss is calculated and then backpropagated through the architecture, towards the filter intensities used by the enhancement network. Via gradient descent, a new intensity combination is computed and the optimization process starts anew. This is repeated until convergence and thus guarantees to maximize the enhancement's aesthetics (as considered by the quality assessor) with an optimal filter intensity combination K' . Figure 4.1 gives an exemplary overview over the proposed architecture, the subsequent sections will elaborate on the separate components.

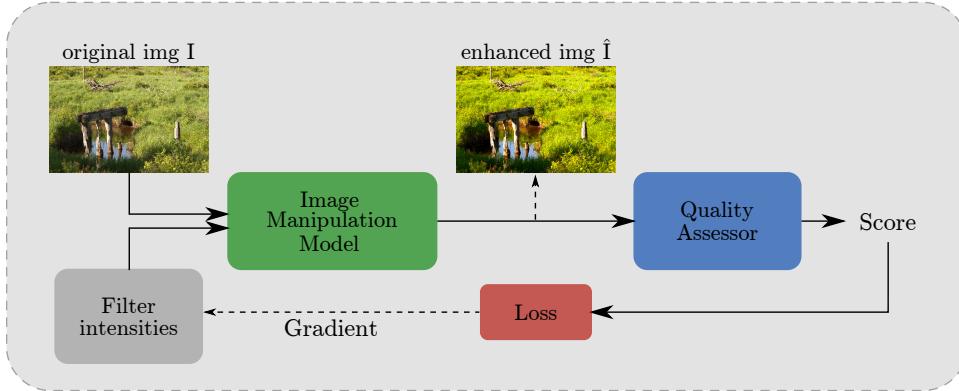


Figure 4.1: Structure of the enhancement pipeline. The unedited input image is passed through the enhancement network and the colors are enhanced. The dashed line between loss and filters indicates that the loss does not directly affect the filter intensities but instead is backpropagated through the network and changes the intensity values via gradient descent. Original image from the MIT-Adobe FiveK dataset [65].

4.2.1 Image Manipulation Model

The image manipulation model presented in Figure 4.1 is used to apply the filters in F with their respective intensities onto the current image and thus is also termed the “enhancement model”. Referring to section 4.1, it can be interpreted as the function $\psi_F : K, \Omega_I \mapsto \Omega_{\hat{I}}$. Although image manipulation is possible with a variety of algorithms and frameworks (cf. section 3.2, [72–74, 76]), the presented approach requires the enhancement function ψ_F to be fully differentiable in order

to allow for backpropagation of the loss. This severely impairs the viability of classic computer vision libraries like OpenCV or PIL. While there are differentiable frameworks for image enhancement (e.g. Kornia [73]), the use of a predefined library limits the action space of the enhancement model and reduces the generality of the approach, since it restricts the number of filters the enhancement model can replicate and imposes premises on the filter parametrization and implementation. However, most image enhancement operators are based on a mathematical model. Neural networks are well suited to approximate mathematical models, with one of their key aspects being differentiability. It therefore is logical to approximate the enhancement process with a neural network.

The approach presented in this thesis uses the CAN architecture (cf. section 3.2.2) introduced by Yu *et al.* [22] and adapted by Chen *et al.* [4] to model the image enhancement process. Aside from the previously mentioned, architecture-inherent advantages (compactness, speed, dense prediction, context aggregation via large receptive field), choosing the CAN as enhancement model is beneficial for several other reasons: First – and most importantly, the CAN architecture is able to approximate a large variety of image filters [4], and consequently makes a very flexible and general enhancement model. Secondly, the CAN enables the optimization pipeline to emulate the image enhancement workflow of a photographer: It allows to apply several filters simultaneously instead of successively optimizing one filter after the other. Lastly, the CAN can learn to approximate additional filters at the negligible extra cost of one multiply-add, leaving space for future extensions.

In their work, Chen *et al.* introduce several versions of the CAN architecture that differ significantly in terms of accuracy and compactness. In general, the CAN models are characterized by the two parameters layer depth d and channel count – also called layer width – w . The CAN32 is the most accurate model, using $d = 10$ layers with $w = 32$ channels per layer [4]. This thesis adapts the CAN24 model ($d = 9$, $w = 24$), as it provides a good trade-off between accuracy and speed (for a comparison on performance and accuracy of CAN24 and CAN32 cf. appendix B). Table 4.1 gives a detailed summary of the architecture. Between layers one to eight, a leaky ReLU non-linearity is applied, with a negative slope of -0.2 . The last layer has no activation function, as it needs to project the previously gathered information on the pixel intensities into the RGB color space.

Layer	1	2	3	4	5	6	7	8	9
Convolution	3×3	3×3	3×3	3×3	3×3	3×3	3×3	3×3	1×1
Dilation	1	2	4	8	16	32	64	1	1
Padding	1,1	2,2	4,4	8,8	16,16	32,32	64,64	1,1	-
Receptive Field	3×3	7×7	15×15	31×31	63×63	127×127	255×255	257×257	257×257
Width	24	24	24	24	24	24	24	24	3

Table 4.1: CAN24 architecture overview.

There are some technical differences between the CAN24 presented in [4] and the model used in this approach: While the basic network structure presented in Table 4.1 is identical, the image filters that the network is trained to approximate are different. As mentioned, the goal of this thesis is to automate image enhancement and to emulate the way a professional photographer would retouch a picture. Therefore, many of the filters used in [4] are irrelevant, as they do not increase visual image appeal (Rudin-Osher-Fatemi (ROF), L_0 -Smoothing) or are of artistic nature (pencil drawing, style transfer). Instead, we propose six filters that are considered well-suited for the task of photography enhancement and used by most professionals (1.–6.), and adapt two filters (7., 8.) from [4]:

1. **Saturation:** changes the intensity of colors in the image
2. **Contrast:** increases the variation between image regions of low and high intensity
3. **Brightness:** changes image light- or darkness by adjusting the pixel intensities
4. **Shadows:** alleviates or amplifies the intensity of dark image regions
5. **Highlights:** alleviates or amplifies the intensity of bright image regions
6. **Exposure:** changes image light- or darkness by simulating varying shutter speeds and thus changing the amount of light that reaches the camera sensor.
7. **Local Laplacian Filtering (LLF):** accentuates details by edge-aware filtering [77]
8. **Non-local Dehazing (NLD):** remove haze for increased fidelity [62]

Figure 4.2 demonstrates the effect of each filter and shows a major difference between our approach and [4]: Chen *et al.* trained their model with image pairs of the original and the filtered image. This was a logical choice in [4], as the filters that were used are of unilateral nature, i.e. usually are applied only in the positive direction. Consider, for example, the case of ROF-Denoising [61]: The higher the filter intensity is set, the more noise is reduced from the image. Negative values introduce additional noise, which is usually undesired. This applies to many filters used in [4]. Our approach, however, aims to emulate the behaviour of image editing software, where the filter intensity can be decreased and increased (cf. Fig. 4.2).

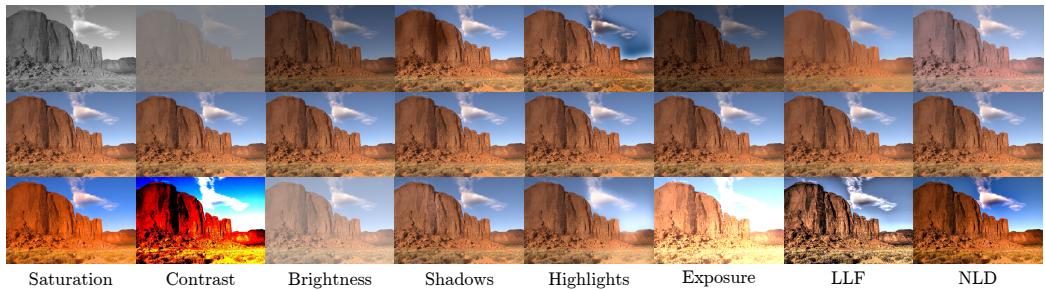


Figure 4.2: Filter performance on a sample image from MIT-Adobe FiveK [65]. The respective filter was applied to the original image (middle row) with full negative (top row) and full positive (bottom row) intensity.

Another difference is that Chen *et al.* state that Batch Normalization (BN) improves accuracy on certain image filters and thus apply an adaptive batch normalization of the form

$$\hat{I} = \lambda_s I + \mu_s BN(I) \quad (4.2)$$

during training [4]. In the above equation, λ_s is a scalar factor that adjusts the strength of the identity mapping of the original image I onto the manipulated image \hat{I} , and μ_s weights the strength of the BN branch. We did not adapt this adaptive batch normalization scheme, as it is primarily useful for artistic image filters (style transfer, pencil drawing) and did not improve the performance of our network on photographic filters. This is confirmed by the findings of Table 5 in [4].

Training

To train the CAN, we used the MIT-Adobe FiveK dataset [65] with a 50/50 train/test split, resulting in 2.500 images per set. For the six photographic filters (Saturation (Sat), Contrast (Con), Brightness (Bri), Shadows (Sha), Highlights (Hig), Exposure (Exp)), we employed the image manipulation program GIMP¹ to create two manipulated versions of each original image, with the filter intensity set to full on and full off. For the LLF and NLD filter, we used the implementation of Aubry *et al.* [77] and Berman *et al.* [62], respectively, with the parametrization given in Table 4.2. NLD is a unilateral filter, as negative values would introduce additional haze (cf. Fig. 4.2), and thus is only used in one direction.

	LLF	NLD
	$\sigma = 0.2$	
low	$N = 5$	–
	$k = -1$	
		$A_1 = 0.72$
high	$N = 10$	$A_2 = 0.785$
	$k = 5$	$A_3 = 0.81$

Table 4.2: LLF and NLD filter parametrization. Note that we used the Local Laplacian Filter in version “standard”, as described in [77]. In standard LLF, σ expresses the standard deviation of the Gaussian convolution kernel used to process the image, N is the number of intensity discretization values and k is a factor for remapping [75]. In NLD, A is the airlight estimation for the current image, upon which the algorithm builds its results [62]. We used a sample airlight matrix provided by Berman [62] for all images.

This results in 2.500 image triplets for the filters Sat, Con, Bri, Sha, Hig, Exp and LLF, and 2.500 image pairs for NLD, summing up to 50.000 training images in the processed training set. To inform the network about which filter has been used to manipulate the current image, the image is concatenated with so-called filter

¹The GIMP team, GIMP 2.10.1, www.gimp.org, 1997-2019, retrieved on 30.01.2020.

layers. Filter layers share the image’s dimensions and represent the image filters the CAN is trained to approximate. When a certain filter has been used to create the manipulated training image, the corresponding layer is filled with the filter’s intensity (± 1.00 , corresponding to $\pm 100\%$). All other layers are set to zero.

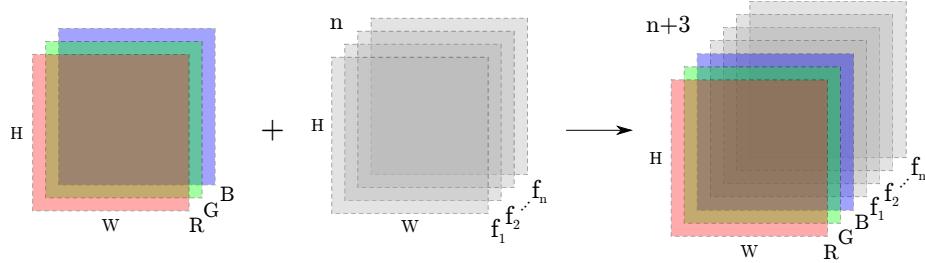


Figure 4.3: Exemplary construction of filter layers. Our approach uses $n = 8$.

The training procedure followed [4]: The network was trained using the image-space regression loss presented in equation 3.1, for a total of 500.000 iterations, with one randomly sampled image per iteration. To illustrate the filter’s effects on varying dimensions, images from the processed training set were randomly resized to a resolution between 320p and 1.440p with fixed aspect ratio. The network was trained using Adam [78] with a learning rate of 1×10^{-5} . Figure 4.4 shows the decline of the training loss curve, as the network learns to approximate the effects of each filter.

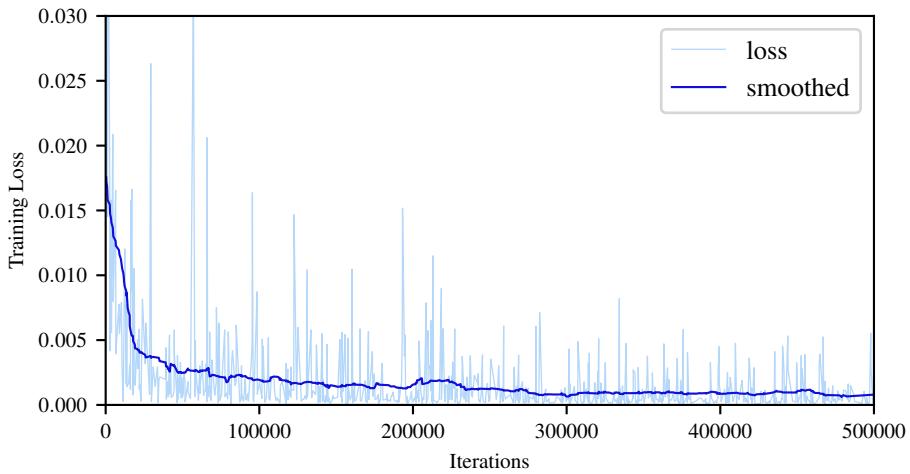


Figure 4.4: The CAN training loss (transparent), with smoothed course (opaque) for better interpretability. After ten epochs with 50.000 iterations, the training loss roughly is 1×10^{-4} . The performance of the trained model on the test set is shown in Table 4.3.

During early training stages, the training loss shows random spikes every few iterations. We attribute this to the fact that some operators are highly dependent on the spatial distribution of image content (LLF) and illumination (Shadows, Highlights, Exposure). Reconsidering Figure 4.2, it is evident that the scale of variation depends greatly on the filters: While Saturation, Contrast and LLF induce significant changes in the manipulated images, Shadows and Highlights only change the image marginally, even when applied with full intensity. Considering these facts, it seems that the network initially struggles to understand that different filters induce changes of different granularity. Towards the end of the training, the effect is mitigated and the loss converges.

Performance

The network performance on the test set is shown in Table 4.3. We applied the preprocessing from [4] and resized all images in the test set to 1080p while keeping their aspect ratio. For reference, the table also shows the results for the trivial baseline ‘‘Plain’’, which measures the distance between the original image without any processing or modification and the edited picture. Note that we renounce from comparing our results to Chen *et al.* [4] for a variety of reasons: Firstly, only two filters of our setup are used in [4], and even then the intensities values in [4] cover only half the range we employ (0-100% in [4] vs. $\pm 100\%$ in our approach). Secondly, Chen *et al.* evaluate their results on five different CAN architectures which all differ from the model we employ, making an expedient comparison infeasible. Our model has a slightly richer parametrization (Chen *et al.* CAN24-AN: 37.000 parameters, ours: 38.715 parameters) and faster runtime, which we attribute to the use of newer, dedicated hardware (Chen *et al.*: NVIDIA TitanX, ours: NVIDIA RTX2080 Ti).

Saturation			Contrast			Brightness			Shadows			
	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM
Plain	493.94	35.65	0.916	3678.0	30.48	0.500	3387.8	30.90	0.791	1405.0	34.87	0.815
Ours	40.19	40.52	0.976	50.29	41.75	0.937	21.28	42.01	0.991	136.43	38.23	0.952
Highlights			Exposure			Local Laplacian Filter			Non-local Dehazing			
	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM
Plain	848.41	36.30	0.968	3879.9	30.45	0.751	443.21	35.71	0.836	1133.4	33.75	0.823
Ours	78.36	39.16	0.976	31.37	41.60	0.984	132.25	37.94	0.943	210.03	37.18	0.935

Table 4.3: CAN24 performance across the test set, given for the respective image filters. The Mean Structural Similarity across the entire image (MSSIM) is given as SSIM for brevity (cf. section 2.3.3). Values are reported as the average across the test set.

Table 4.3 illustrates that the CAN24 approximates the image operators remarkably accurate. On average, the CAN prediction of the image reduces MSE between the edited ground truth and the “Plain” baseline by 90.22%. Additionally, the mean PSNR is increased by 6.16 dB. The approximation achieves SSIM values above 0.950 on five of the eight image filters, and SSIM values over 0.930 on the remaining operators.

Visual inspection confirms these findings and shows that the trained network yields results of decent quality. Figure 4.5 demonstrates the network’s effects with the ground truth enhancement from GIMP and the algorithms of Aubrey *et al.* [77] and Berman *et al* [62]. Although the network was only trained with one filter per image and iteration, it manages to correctly apply several filters simultaneously (cf. Fig. 4.5 (f)). Furthermore, it interpolates correctly for continuous filter values (cf. Fig. 4.5 (f)) and even generalizes to filter values outside of the $\pm 100\%$ range used during training (cf. Fig. 4.5 (e)). Although the more demanding experiments (e) - (h) show higher error, the results still approximate the ground truth accurately, differing only slightly in coloring.

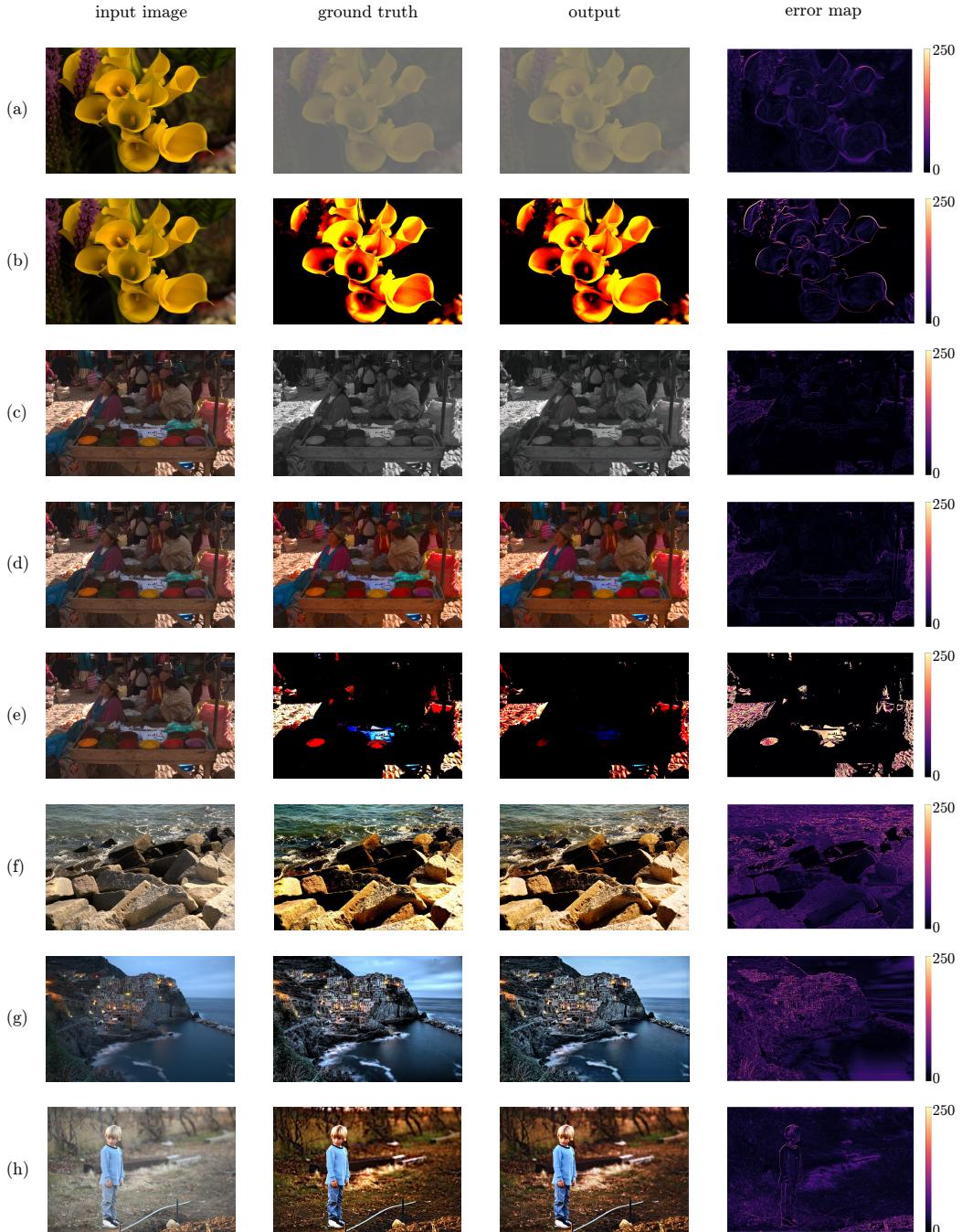


Figure 4.5: The network performance is illustrated on selected filters. The filter settings for the respective row are (a) Con -100%, (b) Con +100%, (c) Sat -100%, (d) Sat +100%, (e) Con +200%, (f) Sat +50% & Con +50%, (g) LLF +100%, (h) NLD +100%. Sample images from the MIT-Adobe FiveK dataset [65].

When inspecting the network’s effects on low-resolution images, it became evident that certain filters introduce artefacts on the image borders. After reaching out to the authors of [4] and discussing the matter, we came to the conclusion that this effect can be attributed to the use of zero-padding, as each feature-map within the CAN is heavily padded as dilation rate increases (cf. Tab. 4.1). Lower resolution leads to a decrease in the number of convolutions per image axis, as the convolution kernels are of fixed size. Therefore, the information encoded in the network’s feature-maps decreases with distance from the center, as there are not as many convolutional operations. This leads to artefacts that show as black stripes parallel to the image borders. Contrast- and detail-enhancing filters like LLF further aggravate this effect, especially if the image borders do not contain other details or structures. Chen suggested the use of different padding styles to alleviate the issue, but results were not compelling. We resolved the issue by fine-tuning the affected operators (Con, LLF, NLD) with concentrated training data. The results are shown in Figure 4.6. Although improvements are only marginal, the effect does not impair the use of the CAN architecture, as image enhancement rarely is conducted on low-resolution images.



Figure 4.6: Fine-tuning operators helps to reduce padding artefacts. The above images (b) - (d) were created from input (a), using the trained CAN architecture with $LLF = +100\%$ and all other filters set to zero. Note how the use of reflection padding does not improve the results in (c), whereas retraining (d) slightly reduces the artefacts on the right image border in the output of the original CAN24 network (b). Sample image from [34].

The presented image enhancement architecture performs well for a variety of filters and thus makes the enhancement pipeline flexible and versatile. The following section will discuss the quality assessor that is used to judge the aesthetics of the enhancement model’s output.

4.2.2 Quality Assessor

Clearly, image aesthetic is perceived differently across individuals, rendering the optimization for maximum image beauty an “ill-posed” problem. Subsequently, the quality assessor used in this approach must be of a general nature and show good generalization over a variety of photographic motives and styles. As the aesthetic classifier will implement the metric M (cf. section 4.1), it furthermore is indispensable that the aesthetic assessment does not solely consider the semantic image content, but also is sensitive to image enhancement.

As previously discussed, NIMA by Halebi and Milanfar (cf. section 3.1, [8]) meets the above desiderata and achieves state-of-the-art performance on aesthetic image assessment, which is a crucial property when optimizing for maximum image beauty. Moreover, the NIMA approach is agnostic to the underlying classifier network, which leaves room for experiments and allows the final application to be tailored to specific real-world and hardware constraints. In [8], NIMA is shown to be sensitive to enhancement operations such as contrast enhancement, image denoising or detail boosting (cf. section 3.2.2, [10, 27]). Lastly, NIMA generalizes well across datasets and photographic conditions, as Halebi and Milanfar illustrate when assessing NIMA performance on high quality images (AVA) and on mobile-captured snapshots of lower quality from the LIVE dataset ([8, 37]).

For these reasons, NIMA is used as quality assessment model in this approach and consequently implements the metric $M(\psi_F)$ that is used to judge the aesthetic of the enhancement’s outcome with the current set of filter intensities. The aesthetic prediction is a score distribution in the range [1, 10] that equals a probability distribution, as it is generated by a soft-max function. The mean of this distribution will serve as quality score and thus determines whether the aesthetics of the enhancement model’s output are high or low.

Implementation Details

NIMA consists of two main components: a feature extractor, implemented as a CNN, and a score regressor, commonly implemented as fully connected layer between the feature extractor and the ten score buckets. In [8], Halebi and Milanfar test their approach with three different classifiers (VGG16, MobileNet, Inception-v2), that each have distinctive properties (cf. section 2.1.1). In this thesis, we chose to adapt VGG16 and MobileNet, as the VGG architecture achieves best cross-dataset performance [8] and the MobileNet approach is lightweight and compact [17]. To perform score prediction with these classifier architectures, the last CNN layer is replaced by a fully connected layer with ten nodes that correspond to the ten score categories. The layer’s output is then fed through a soft-max activation function to create a probability distribution.

Both networks are trained with the EMD loss proposed in [8]

$$\text{EMD}(\mathbf{p}, \hat{\mathbf{p}}) = \left(\frac{1}{N} \sum_{k=1}^N |\text{CDF}_{\mathbf{p}}(k) - \text{CDF}_{\hat{\mathbf{p}}}(k)|^2 \right)^{1/2}, \quad (4.3)$$

where $N = 10$ is the number of ordered score buckets (one being the lowest, ten being the highest score category) and CDF is the cumulative distribution function of the true and estimated distributions \mathbf{p} and $\hat{\mathbf{p}}$, respectively. For training, we use 80% of the AVA dataset and split the remaining 20% equally into validation- and test-set. The training procedure largely follows [8]: For both VGG16 and MobileNet, we use models that were pre-trained on the ImageNet dataset [12] and re-train the convolutional and fully connected layers per SGD with momentum 0.9. As both VGG16 and MobileNet expect the input to have dimensions 224×224, all images are rescaled to 256px while keeping their aspect ratio, and then random-cropped to patches of size 224×224. For additional data augmentation, a horizontal flip is applied randomly during training. Halebi and Milanfar state that small learning rates stabilize and accelerate the training [8], which is why we train the convolutional and dense weights with a learning rate of 3×10^{-7} and 3×10^{-6} , respectively. During training, a 0.75 dropout rate is applied to the fully connected layer to avoid overfitting. We furthermore employ a learning rate decay by a factor of 5% every ten epochs. Figure 4.7 shows the training loss for both architectures.

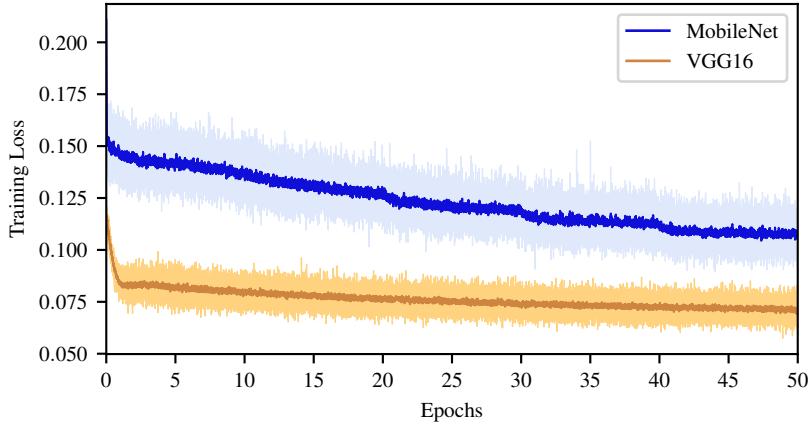


Figure 4.7: Training loss for the different NIMA base models, with smoothed course for better interpretability. The training proceeded for 50 epochs. The performance of the trained models on the test set is shown in Table 4.4.

NIMA with the VGG16 base classifier outperforms the MobileNet-based NIMA version, which corresponds to the findings in [8]. We attribute this to the fact that VGG has a richer parametrization than MobileNet and can therefore encode more information within its feature maps that can then be used for quality assessment.

MobileNet, however, is faster in CPU inference and more compact, which makes it more applicable in mobile or performance-sensitive applications. We manually tested both architectures on sample images from the AVA test set. Figure 4.8 directly compares NIMA performance while Table 4.4 summarizes the architectural details.

	Million Parameters	Model Size (MB)	GPU Timing (ms)	CPU Timing (ms)	EMD
NIMA(MobileNet)	2.85	164.8	7.5	29.3	0.117
NIMA(VGG16)	14.97	276.3	1.8	126.7	0.073

Table 4.4: Comparison between the two different NIMA base classifiers. Timing is reported for a single forward pass of a $224 \times 224 \times 3$ RGB image, measured on an Intel i7-7700k CPU @ 3.60GHz and an NVIDIA RTX2080 Ti GPU. Note that MobileNet is implemented as MobileNet_v2 from the PyTorch repository, which uses depthwise separable convolutions that are not optimized on GPU firmware [79] and therefore exhibit a higher runtime than VGG16 when using a GPU. EMD is reported as the average across the AVA test set.

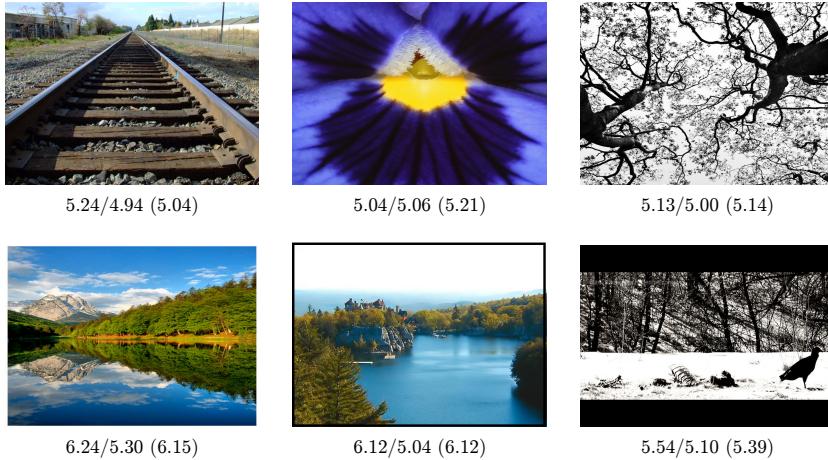


Figure 4.8: Comparison between NIMA predictions for selected images from the AVA test set. Predictions are given for the VGG / MobileNet NIMA base model, respectively, with the AVA ground truth in parenthesis. The upper row shows that both VGG and MobileNet predict the rating well, albeit VGG predictions are generally more accurate and closer to the ground truth (cf. lower row).

Using NIMA as an evaluator to rate the enhancement model’s output allows our approach to perform quality assessment that generalizes across a wide range of semantic image content and photographic styles. NIMA is well suited for this task, as it has been shown to be sensitive to image enhancement [8, 10]. The following section elaborates on the backpropagation details and explains the loss function that builds upon NIMA’s quality prediction.

4.2.3 Loss and Backpropagation

As Figure 4.1 shows, the NIMA quality score prediction is used to compute a loss that is then backpropagated through the enhancement pipeline, towards the filter intensities used by the CAN to create the enhanced image. The intensity values are adjusted via gradient descent, and the process starts anew. By default, all filter intensities are initialized as zero. In the final application, the initial filter intensities will be adjustable according to user preferences. For the optimization routine, we employ a loss function of the form

$$\mathcal{L}(K, \hat{I}) = \text{EMD}(p_t, p_d) + \gamma L_2(K) , \quad (4.4)$$

where K denotes the combination of filter intensities that created the enhanced image \hat{I} . The loss consists of two terms: Firstly, we take into account the EMD between the true, predicted score distribution p_t that the enhancement \hat{I} received, and a desired distribution p_d . When minimizing the overall loss, this term is responsible for adjusting the filter combination in a way that the NIMA predictions become close to the desired distribution. This distribution is modelled to correspond to a very beautiful image that received high AVA ratings (cf. Fig. 4.9). Note that we also experimented with score distributions of higher mean, including a Dirac delta distribution [49] with mean $\mu = 10.0$ and $\sigma = 0.0$. As the differences between the optimization results were minuscule, we chose to employ a distribution that could be found in the actual AVA dataset.

The second term in equation 4.4 can be interpreted as L_2 regularization [80] between the employed filter intensities k_i and the initial intensity values k_{i_0} :

$$L_2(K) = \sum_{i=1}^n (k_{i_0} - k_i)^2 . \quad (4.5)$$

This term is responsible for keeping the amplitudes of the filter intensities within acceptable boundaries. Informally speaking, it “punishes” high deviations from the initial filter intensities k_{i_0} and therefore forces the optimization process to balance the gain in EMD decrease that comes from optimizing towards a certain set of filter intensities, and the penalty that these intensity values entail. This ensures that the enhancements imposed on the resulting image do not alter the image style too drastically, as this is usually undesired. The L_2 regularization term is weighted by a factor γ , which hence can be interpreted as an indicator of how strong the optimization will alter the image: lower values of γ allow for higher filter intensities, while a higher γ keeps the intensity values close to zero. We empirically determined $\gamma = 0.1$. Note that we do not impose any additional borders on the filter intensities, which means that the optimization could possibly drive the values beyond the range of $\pm 100\%$.

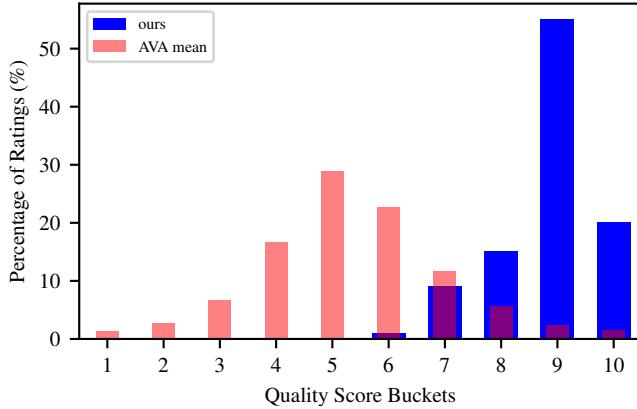


Figure 4.9: The desired distribution for optimization, with mean $\mu = 8.84$ and standard deviation $\sigma = 0.88$. To create this distribution, we randomly sampled values for high-ranking AVA image ratings to achieve a realistic rating and normalized the result.

The loss that is calculated using equation 4.4 is backpropagated through the enhancement pipeline via gradient descent [81, 82]. The gradients are applied to the filter intensities, and a new set of filter layers with adjusted intensity values is created (cf. Fig. 4.3). Each intensity change is applied onto the respective layer globally, i.e. all intensities in a filter layer are changed by the same value. For an evaluation of local gradient application, see section 5.2.1.

We also experimented with the loss proposed by Halebi and Milanfar [10]

$$\mathcal{L}(\hat{I}) = 10.0 - \text{NIMA}(\hat{I}) , \quad (4.6)$$

which penalizes the difference between the best possible NIMA score 10.0 and the current NIMA prediction of the enhanced image \hat{I} . However, results were not compelling: While this loss function produced good results in [10], it led to the creation of highly edited, high-scoring images in our approach. Oftentimes, these images had no visual appeal and were not aesthetically pleasing. We attribute this to the fact that Halebi and Milanfar optimize image operator hyperparameters for a single filter instead of setting intensity values for multiple filters. The approach in [10] hence changes the image with a single filter, whereas our approach has $n = 8$ filter intensities to adjust. We therefore conclude that the above loss function is not dynamic enough for our approach, as it does not consider the filter intensity values. Figure 4.10 shows selected results of the optimization procedure for both loss functions. Interestingly, the radical enhancement receives the highest ratings, which, considering that NIMA emulates human judgement, is counter-intuitive. The strong filter intensities delude NIMA into predicting a high score, although the image is not aesthetically pleasing. This phenomenon shall be further discussed in section 5.2.1. For the remainder of this work, the loss presented in equation 4.4 will be used for the optimization routine.

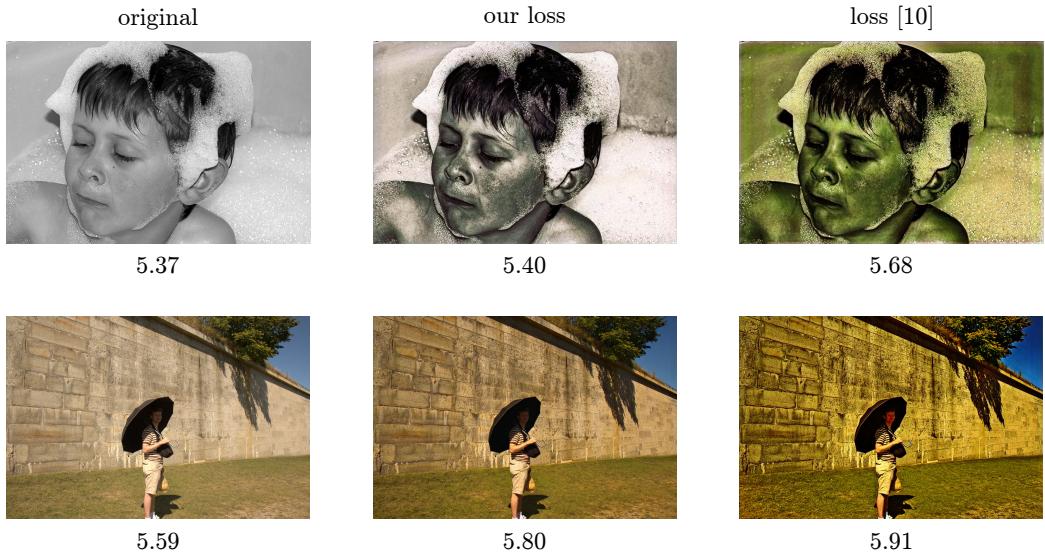


Figure 4.10: Optimization results for the different loss functions. The first column shows the original image, while the second and third columns show the optimization results with our loss (cf. eq. 4.4) and the loss used in [10] (cf. eq. 4.6), respectively. All other parameters are identical, the NIMA-VGG16 predictions are given below each image. Note how the enhancement in the last column differs drastically from the previous images. Original images from MIT-Adobe FiveK [65] and AVA [34].

4.3 Design and Implementation

The enhancement pipeline that is used by the proposed approach has been introduced and described in the previous chapters. This section will specify the employed hyperparameters and elaborate on implementation and further details.

NIMA architecture

As described, we trained NIMA with two different feature-extraction networks, VGG16 and MobileNet. While MobileNet is more compact and efficient, VGG16 is more expressive and accurate. To decide on the model for the final approach, we visually evaluated the enhancement results for each classifier. Interestingly, although the models’ EMD performance on the AVA test set differs only slightly, the employed NIMA baseline classifiers alter the enhancement outcome drastically. This holds true especially for MobileNet: While it outperforms VGG16 in terms of speed, it is less parametrized and therefore captures less details of the original, unedited image. MobileNet thus has less information available to assess the enhancement’s quality, which leads to an inferior final result when compared to VGG16. This phenomenon was observed with our own implementations as well as with publicly available reference implementations. Occasionally, MobileNet enhancements differ severely from what is perceptually pleasing (cf. section 5.2.1).

This so-called “failure-case” occurs on 5–10% of all images (randomly sampled from MIT-Adobe FiveK and AVA test sets) for MobileNet and presumably happens as NIMA increases the saliency of features that its feature-extraction network learned to analyse. Since VGG16 has more parameters, the number of learned features is higher and the enhancement is more balanced. Figure 4.11 shows the enhancement’s outcome for the two NIMA base classifiers.



Figure 4.11: Selected enhancement results for the different NIMA base classifiers. MobileNet is more likely to produce a so-called “failure-case”, where the colors are shifted or boosted unnaturally, resulting in an inferior enhancement. Original images from MIT-Adobe FiveK [65] and AVA [34].

We therefore choose VGG16 (configuration D in [3], cf. middle row Tab. 2.1, center column Tab. 2.2) as the NIMA base classifier for our final approach. We use the VGG-implementation provided by the PyTorch repository, with weights that were pre-trained on the ImageNet dataset [12]. We train the NIMA network as described in section 4.2.2.

Optimization

For training and inference, VGG uses $224 \times 224 \times 3$ image crops. Consequently, this is the scale at which the enhancement pipeline and the optimization process must operate: an unedited $N \times N \times 3$ image first is resized to $224 \times 224 \times 3$ and then concatenated with the eight filter layers (cf. Fig. 4.3), resulting in an input of dimension $224 \times 224 \times 11$. The resized, concatenated image is then fed through the CAN and NIMA networks multiple times, while the optimization process changes the filter intensities. Once the loss converges, the filters are applied to the full-scale image to receive an enhanced image of original size.

We experimented with different optimizers for the enhancement pipeline. Overall, Stochastic Gradient Descent (SGD) with Nesterov Momentum [84, 85] and Adam [78] provided the best results. Adam slightly outperformed SGD in terms of average NIMA score on the test set (SGD: 5.593, Adam: 5.602). Generally, Adam tends to find a filter setting that receives a relatively high-ranking NIMA score, but often is not visually pleasing. As Figure 4.12 shows, SGD-optimized images often look more natural and are aesthetically superior, while receiving similar NIMA scores. In our final approach, we therefore employ an SGD optimizer, with Nesterov Momentum of 0.9 and a learning rate of 0.05. Note that in a minority of cases, the aesthetics of Adam-driven optimization outperform SGD’s enhancement. As of yet, it is unclear why and when this happens, and further research needs to be facilitated on this matter.

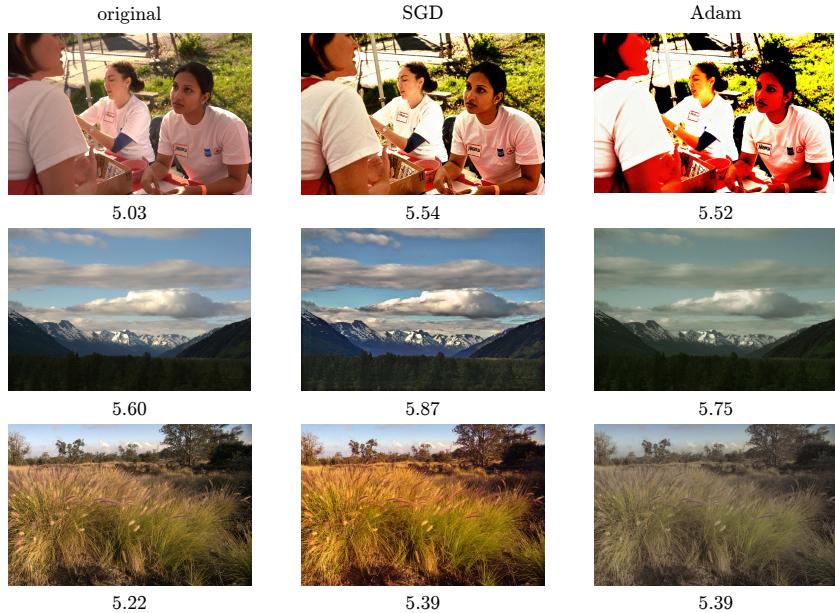


Figure 4.12: Comparison between enhancement results when using SGD or Adam optimizers. Note how Adam boosts or de-saturates the colors, while still receiving a relatively high score. Original images from the MIT-Adobe FiveK test set [65].

4.3.1 Visual Verification

A full evaluation of the approach’s performance will be conducted in chapter 5. However, to find the best-suited architecture components and fine-tune the hyper-parameters that were described in the previous subsections, we visually evaluated the enhancement outcome with images from the MIT-Adobe FiveK and AVA test sets. First tests and empirical verification yielded results of superior aesthetic

quality (cf. Fig. 4.10 - Fig. 4.12) for some images, but also showed that the approach did not succeed in optimizing all images. When further investigating the issue, it became clear that the optimization procedure struggled to find suitable intensities for the filters that change image lightning, especially Brightness and Exposure. This observation is confirmed by illustrating the NIMA scores for different filter intensities: Figure 4.13 shows the average NIMA score deviation from the unedited reference image when the respective filter is applied. Naturally, the effects of each filter are not equally effective for each image, but rather depend on image content, lightning and photographic style. However, for well-lit images (which make up the vast majority in the AVA dataset), one would intuitively expect rising NIMA scores for higher values of Contrast, LLF, Saturation, etc., and slightly falling scores for lower values. If the intensities become too high (e.g. close to 100%), one would expect the score to decline again, as extreme filter intensities tend to produce aesthetically displeasing results (cf. Fig. 4.5).

Figure 4.13 shows that this expectation is met only for the Highlights filter, and even then only so with a small amplitude of approx. 0.06 for NIMA score changes across the entire filter range. Instead, NIMA is very sensitive to the filters Saturation, Contrast, LLF and NLD. This is confirmed by Figure 13 in [8]. Filters that change image illumination and lightning (e.g. Brightness, Shadows, Exposure) do not comply with our expectation at all. Instead, the NIMA score decreases for both positive and negative intensities. This can best be observed for the filters that change the image more drastically, e.g. Brightness and Exposure (reconsider Fig. 4.2). One should further note that the NIMA-sensitive filters generally induce higher score changes (e.g. Con ≈ 0.6 , LLF ≈ 0.5) than the filters that change image lightning (e.g. Hig ≈ 0.06 , Bri ≈ 0.25).

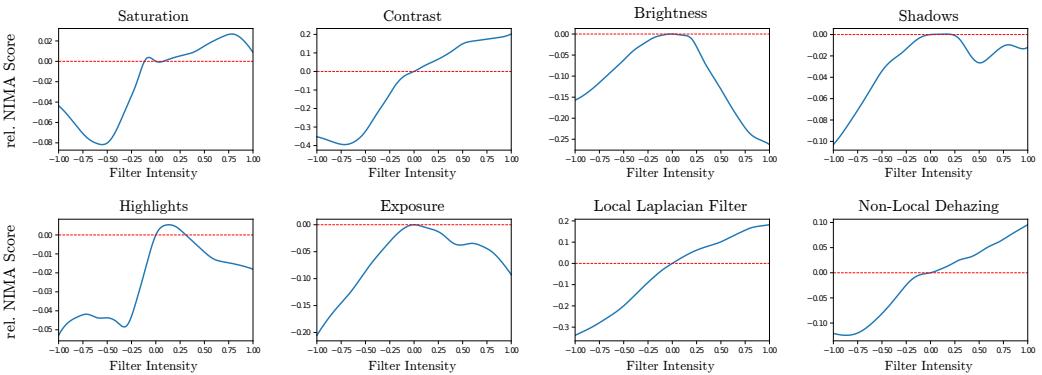


Figure 4.13: NIMA score changes (relative to the NIMA ground score) for varying filter intensities, averaged over 100 randomly sampled images from the MIT-Adobe FiveK and AVA test sets. Only the respective filter was changed in intensity, all other values were set to zero. The red line marks the zero level.

This effect further becomes apparent when feeding manually edited images to the NIMA component of the enhancement pipeline and inspecting the resulting scores: Figure 4.14 presents an image from the MIT-FiveK test set and three versions that were manually edited with GIMP. As the resulting NIMA scores show, the image with high contrast receives a higher quality rating, although its lightning conditions are inferior to other edited versions.

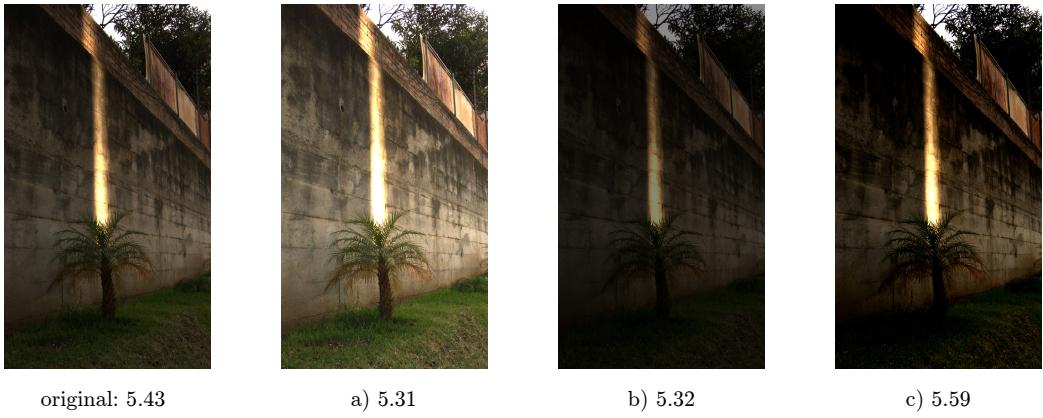


Figure 4.14: NIMA quality scores for three manually edited images. Images a) and b) have been edited with +50% and -50% Exposure, respectively, while image c) was created with +25% Contrast. Note how the change in contrast highlights the lightning bar above the palm, resulting in a higher NIMA score, although a) is visually more appealing. Sample image from the MIT-Adobe FiveK dataset [65].

The cause of this phenomenon becomes apparent when inspecting the data that was used to train the NIMA network: The AVA dataset consists of images that were shot by photographers and thus predominantly exhibit good photographic conditions and illumination. Moreover, quality score distributions in the AVA dataset can be modelled as a Gaussian distribution [34], which indicates that very high- or low-ranking images are highly under-represented. NIMA therefore does not learn to predict extreme image ratings, as they are not a significant part of the dataset. Also, NIMA is not able to predict the change in image quality that comes with changing illumination, as it has not learned to do so during the training phase.

NIMA Retraining

To mitigate this issue, we re-train the dense layer of the NIMA network with a small learning rate of 3×10^{-7} and 3.000 images from the AVA train set. These images have been edited for brightness and exposure with a randomly selected filter intensity in $[0.5, 1]$ and $[-0.5, -1]$. Moreover, the image ground truth annotations have been randomly shifted towards a lower mean score to indicate the decrease in image aesthetic that usually comes with over- or underexposure. We train the dense layer for 40 epochs, intending to make it learn the relation between im-

age rating and illumination changes. Note that this is only a very crude training method and a field for future extensions. Yet, as Figures 4.15 - 4.17 show, the re-trained NIMA network leads to improved optimization results.

Adaptive Brightness Normalization

Additionally, we introduce a preprocessing step which we name Adaptive Brightness Normalization (ABN) to ensure that all images have similar brightness before the start of the optimization cycle. For ABN, we first compute the perceived image brightness in the RGB color space as

$$P = (0.241R^2 + 0.691G^2 + 0.068B^2)^{0.5} , \quad (4.7)$$

where R, G and B are the mean pixel intensities per channel². The perceived brightness ranges in the interval [0, 255] and considers that the human perception of brightness is influenced differently by the intensity of the respective color channels. We define the desired, acceptable brightness, for which no normalization is needed, as $P_d = 128 \pm 30$. If $P > P_d$, the ABN algorithm performs histogram equalization, a linear transformation that can be used across color spaces to correct image lightning. It uses the parameters α and β to shift the color range and the intensity value of each pixel (x, y) in the original image I, respectively [72]:

$$I'(x, y) = \alpha I(x, y) + \beta . \quad (4.8)$$

The values for α and β are calculated by clipping the original histogram's left and right side by 5.0%. Hence, scarce intensity values are omitted, which results in a brighter and more defined image, as the now normalized histogram is spread out across the perceived brightness range more evenly. If $P < P_d$, the ABN algorithm transforms the image into the Hue-Saturation-Value color space, where the Value-channel denotes color brightness, and then increases the V-value by a factor of 20. As HSV-shift results vary greatly with image illumination and often introduce unwanted noise, ABN additionally checks if the PSNR between the corrected image I' and the original version I is above the threshold value $\tau = 30$ and reduces the shift factor, if necessary. Additionally, ABN randomly samples image pixels and checks if the pixel color is either white or black. If this holds true in more than 60% of all cases, ABN concludes that the image is mostly white or black, and that no brightness correction is necessary. This is especially helpful in the case of artistic imagery, where the images are intended to be extremely bright or dark and do not need further preprocessing. Classic examples are adverts, e.g. a pair of glasses displayed on white ground, or single-object photography, where a single item is displayed against a black background. For the ABN pseudo-code and examples of artistic imagery, see appendix C.

²The HSP (Hue, Saturation, Perceived Brightness) color space was defined by Darel Rex Finley, <http://alienryderflex.com/hsp.html>, retrieved on 30.01.2020.

The effects of ABN and the retrained NIMA classifier are shown in the Figures 4.15 – 4.17. Note that ABN supports the enhancement pipeline with a brightness-correcting preprocessing step, but is not indicative or mandatory for a successful enhancement outcome, as Fig. 4.17 shows. The image produced by the final approach, using ABN and the retrained NIMA network, is shown in the lower right corner. All following images are from the MIT-Adobe FiveK dataset.

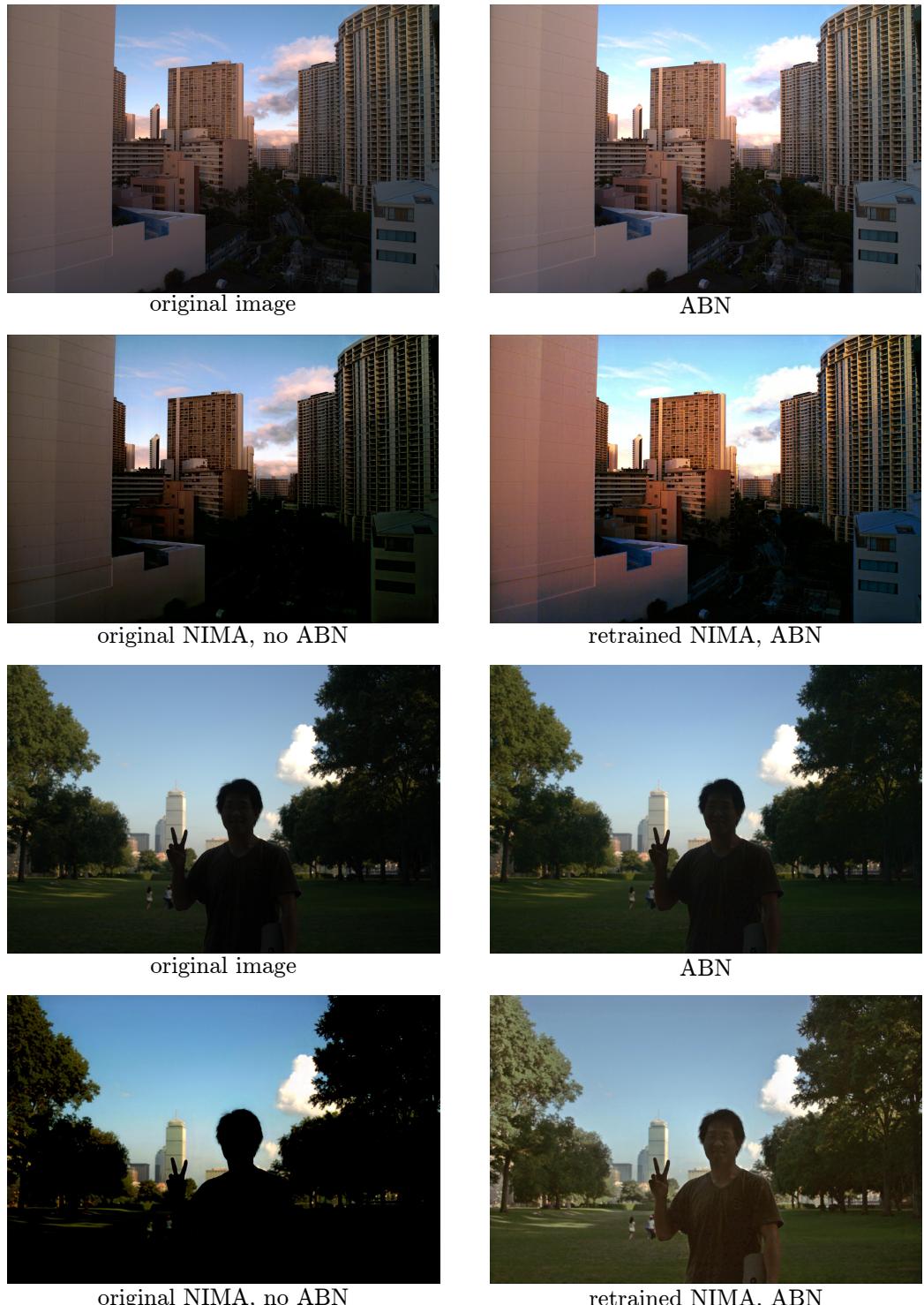


Figure 4.15: Enhancement results for different pipeline structures. Perceived brightness $P = 81.8$ (105.1 with ABN) for the upper original image, $P = 69.9$ (76.7 with ABN) for the lower original image. Note how the retrained NIMA classifier accentuates details and adjusts the illumination in both picture sets, whereas the original NIMA network simply enhances towards high-contrast images.



Figure 4.16: Enhancement results for different pipeline structures. Perceived brightness $P = 94.5$ (119.9 with ABN) for the upper original image, $P = 65.6$ (73.4 with ABN) for the lower original image. The retrained NIMA network is less prone to over-adjust contrast, even when the original image already is of high contrast (cf. lower image set).

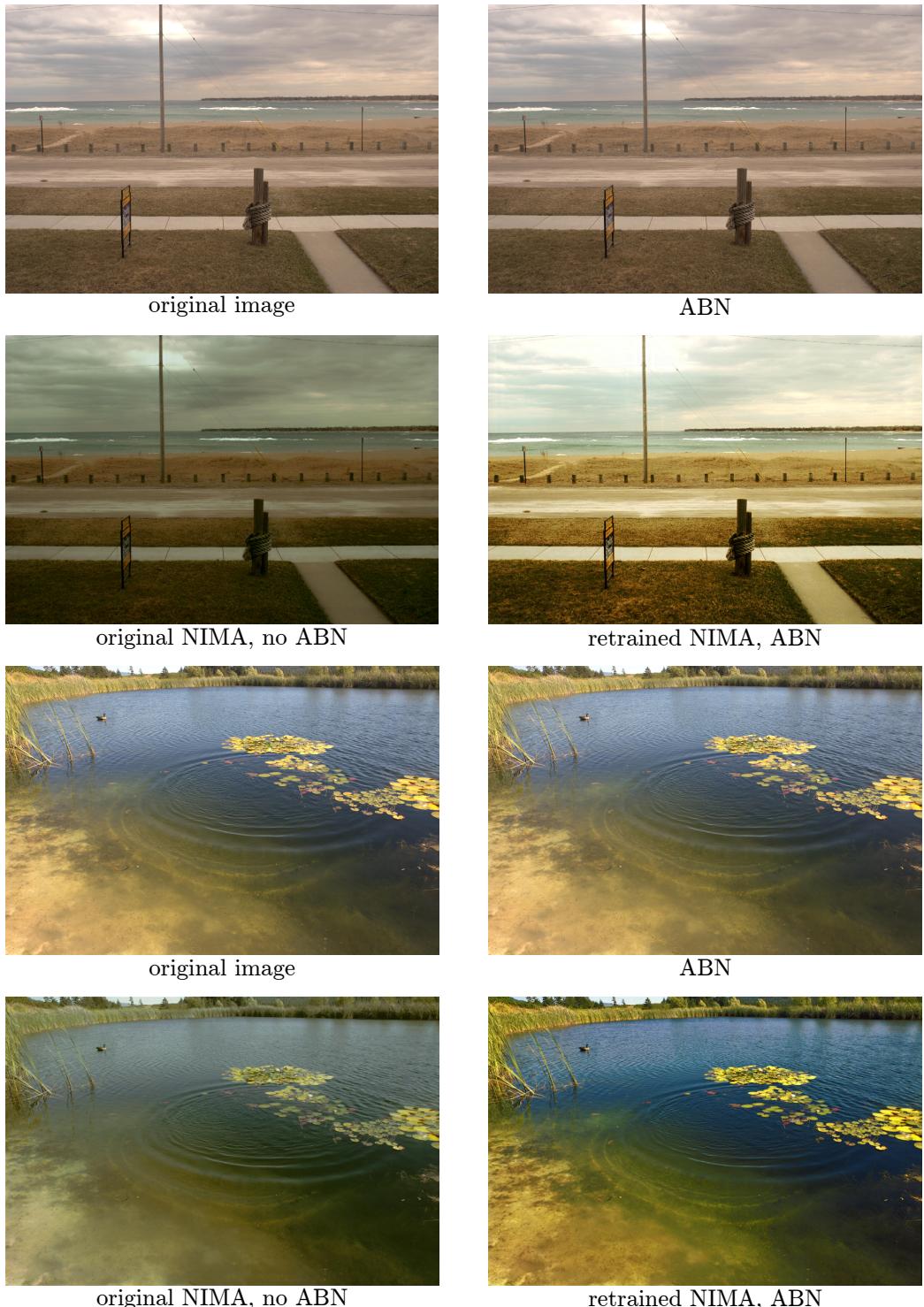


Figure 4.17: Enhancement results for different pipeline structures. Perceived brightness $P = 135.5$ for the upper original image, $P = 117.9$ for the lower original image. Note that ABN was not used, as P is within the acceptable brightness interval (128 ± 30). As the lower right image in each set shows, the retrained NIMA classifier itself — without ABN — is able to enhance the image towards a perceptually superior version if the initial brightness is within the acceptable range.

As Figures 4.15 – 4.17 show, the combination of the retrained NIMA network and the Adaptive Brightness Normalization preprocessor yields the best results. The final approach therefore incorporates both of these extensions into the pipeline that has been described through the course of this chapter. Figure 4.18 shows the final pipeline we use for our enhancement approach.

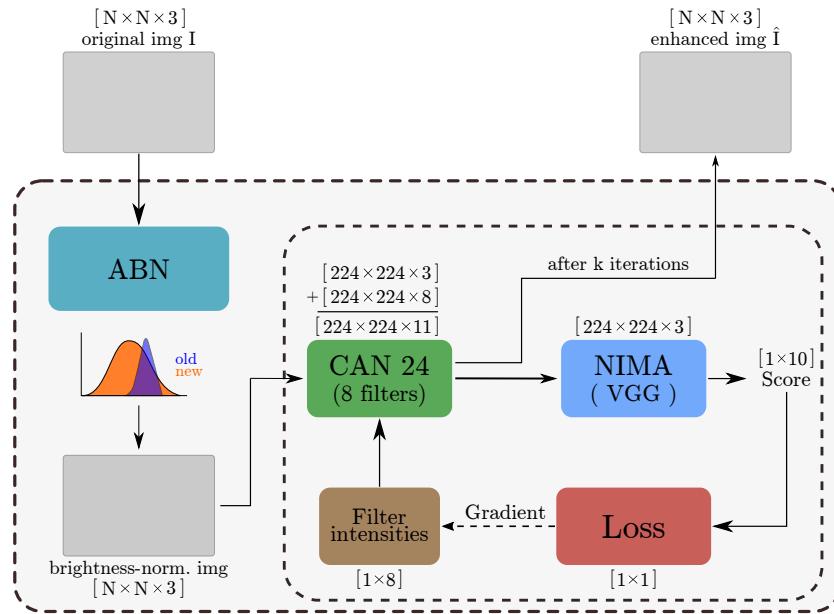


Figure 4.18: The final enhancement pipeline for our presented approach. The annotations in square brackets denote the shape of the image at the respective component. The parameter k can be set manually for every image and defaults to 50 iterations.

4.3.2 Application

The modular design of our method allows the final pipeline to be cast into a standalone application (cf. Fig. 4.20), which is designed after the motivation of our approach: Enabling users to automatically enhance their images towards aesthetically pleasing results, while optionally being able to use personal preferences to guide the optimization results. With our chosen optimization loss (cf. eq. 4.4), the L_2 regularization term penalizes filter intensities that deviate from the initial values. The optimization therefore takes into account the personal editing preferences of the user when enhancing the image and yields different results for varying starting configurations. Figure 4.19 illustrates this effect on a sample image³.

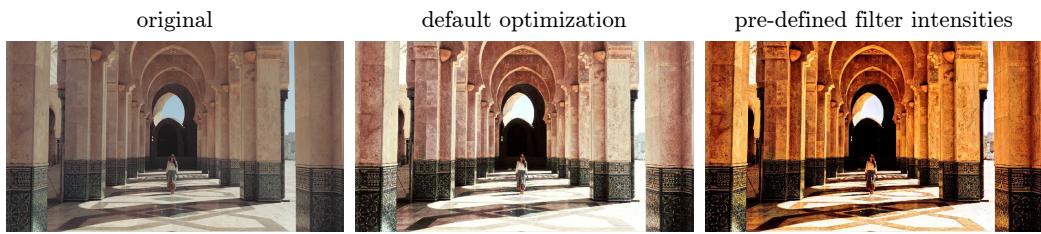


Figure 4.19: Optimization results for varying initial intensity presets. For the default preset (all filter intensities are set to zero), the enhancement pipeline transforms the original image (left) into the middle image and yields the following filter combination (given in %): Sat: -23, Con: 26, Bri: 8, Sha: -3, Hig: 20, Exp: 21, LLF: 13, NLD: 5. The rightmost image is created by re-running the optimization cycle with pre-defined filter intensities. The optimization yields (in %): Sat: 24, Con: 44, Bri: -1, Sha: -1, Hig: 15, Exp: -8, LLF: 54, NLD: 62, which results in a completely different enhancement.

However, the filter intensities do not necessarily have to be set *before* the optimization cycle: Unlike other approaches, our enhancement pipeline allows the user to conduct post-optimization adjustments of the filter intensities if the obtained results are not to the user’s liking. This is a major difference to other, automatic enhancement algorithms, which often enhance the image in a black-box fashion and then present the final result. Since our approach computes the new filter intensities from the backpropagated gradients, the enhancement steps are transparent and can be followed through the optimization procedure. This is a deterministic and reproducible process, as the trained networks do not change their parametrization during the enhancement procedure. Figure 4.20 shows our application’s interface, which displays the filter intensities as adjustable slider values.

³Sample image source: Pexels. <https://www.pexels.com/de-de/video/3015535/>, retrieved on 30.01.2020

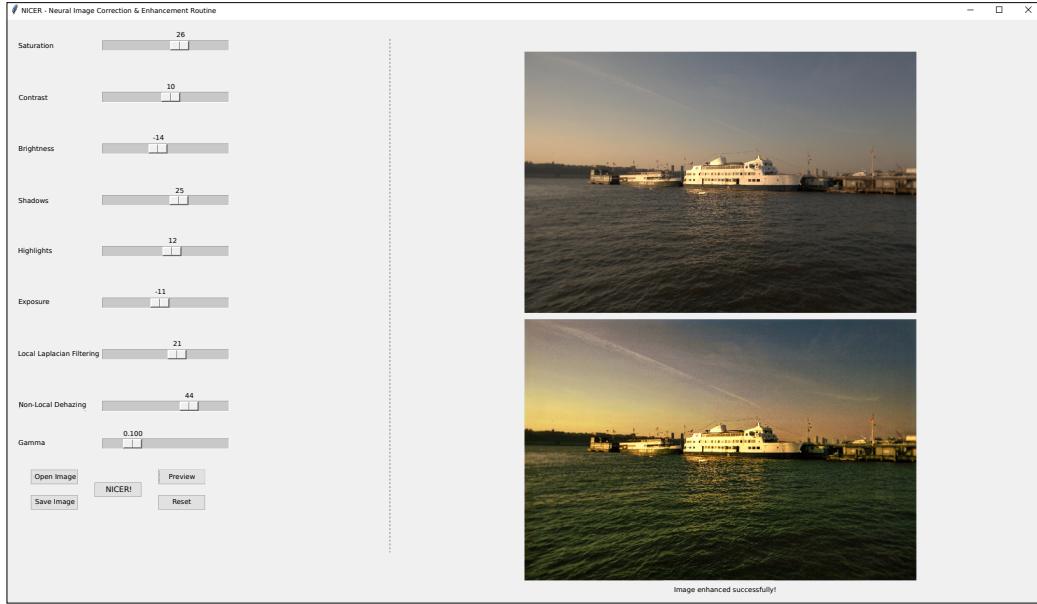


Figure 4.20: The GUI of the final application. The filter intensities can individually be set via the sliders, and the parameter γ controls the enhancement strength. The lower, enhanced image originates from the upper, original image. Sample image from MIT-Adobe FiveK [65].

We provide the enhancement results for the previously mentioned architecture structures and optimizers online⁴. The employed hyperparameters and selected enhancement examples can be found in appendix C. The source code for the enhancement pipeline and the interactive GUI is publicly available on GitHub⁵.

As all parts of the enhancement pipeline are now introduced and accurately described, the following chapter will evaluate the approach’s performance. Furthermore, we conduct experiments to understand and describe phenomena encountered during the implementation of the enhancement pipeline.

⁴Enhancement results for varying architectures and optimizers.

<https://drive.google.com/open?id=1o0SGY-1qhLPI8EDEpiHGllMX9OoJGsjE>, retrieved on 30.01.2020

⁵NICER - Neural Image Correction & Enhancement Routine, Michael Fischer, 2020. <https://github.com/mr-Mojo/NICER>, retrieved on 30.01.2020

5

Evaluation & Experiments

As Figures 4.15 - 4.17 show, the presented approach produces enhanced images of good quality and high aesthetics. However, as image aesthetics and beauty are of very subject nature, it is difficult to measure the approach's performance. This chapter aims to evaluate the resulting images and elaborate on some of the phenomena encountered in chapter four.

5.1 Qualitative Evaluation

As previously mentioned, the goal of our presented approach is to automatically edit images towards perceptually pleasing versions that show high aesthetics. Ideally, the enhancement of the resulting images is as good as the retouching of a professional photographer. As image beauty is hard to quantify, we conduct a qualitative experiment, where the human perception of aesthetic is used to rate our approach's performance. In the evaluation study we design, our enhanced image is intermixed with the results of several other enhancement baselines. The users are then asked to rate the displayed images according to the quality of the respective enhancement.

5.1.1 Baselines

As image enhancement quality is hard to evaluate without further references, we show the users three baselines to compare our edited images to:

Apple Photos. As a first baseline, we used the automatic image retouching feature “Auto Enhance” in Apple’s proprietary “Photos” App¹. The application uses histogram equalization to correct image lightning and face detection for the preservation of skin tone and complexion. Furthermore, the algorithm automatically sets several filter intensities (Saturation, Contrast, Brightness, Shadows, Highlights, Exposure, Brilliance, Black Point, Vibrance, Warmth, Tint, Sharpness, Definition, Vignette, Noise Reduction). Note that all of our photographic filters (1.–6.) are contained in the application’s retouching repertoire. A significant difference between Apple’s application and our approach is that Photos, for some filters, uses spatial image editing. To this end, 31×31 convolutions are computed across the image. The application then performs histogram and tonal curve equalization on each crop, before combining the convolutions’ results for a per-pixel enhancement. As Photos is a commercial application, there is no in-depth information about the inner working of the enhancement algorithm.

Professional Photographer. The second baseline consists of the professionally enhanced images in the MIT-Adobe FiveK dataset. The creators of the dataset hired five professional photographers to manually retouch each image (cf. section 3.2.1, [65]). We compare our enhancement results against the retouched images of Expert A.

Random. To create the final baseline, we enhanced every image in the test set with a random combination of filter intensities and did not process the image any further. Occasionally, the random enhancement achieves effects that could be considered “artistic”, and in some cases the random filter intensities actually enhance the image towards a perceptually pleasing version. However, the majority of the resulting images generally looks distorted in color, tone and illumination.

Figure 5.1 demonstrates the different editing styles for the respective baselines. The photographer often deliberately changed the image mood by altering illumination and color warmth (cf. columns C, D). Also, the professionally retouched images generally exhibit brighter whites (cf. the blanket in column B), which can consistently be observed for Expert A and therefore be attributed to the personal preferences of the photographer. “Photos” often changes the original image only marginally. Generally, its histogram equalization lifts dark shadows from the im-

¹Apple Photos: Private, on-device technologies to browse and edit photos and videos on iOS and iPadOS. https://www.apple.com/ios/photos/pdf/Photos_Tech-Brief_Sept_2019.pdf, retrieved on 30.01.2020

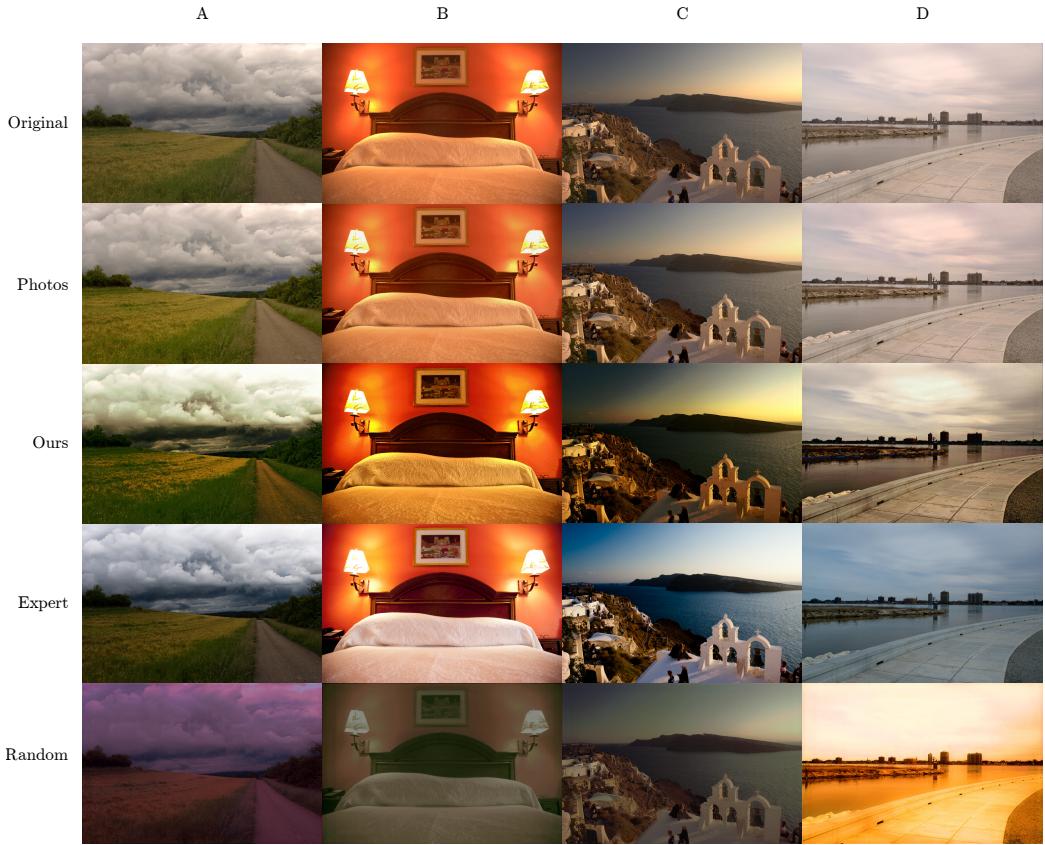


Figure 5.1: Comparison of the different baselines, original images and our approach. Original and expert images from the MIT-Adobe FiveK dataset [65].

age and makes the resulting images appear brighter and more vibrant (cf. column A, D). Our enhancement approach accentuates saturation, contrast and details. This confirms the findings of section 4.3.1 (cf. Fig. 4.13) and shows especially on the blanket in column B and the walkway in column D. Moreover, our method tends to optimize towards filter combinations that promote image yellowness, as can be seen in the rapeseed field in column A and the sunset in column C.

Generally, with the chosen L_2 -weight $\gamma = 0.1$ (cf. section 4.2.3), our method enhances the images stronger than Photos. The photographer’s images often show large differences to the original image, as the lighting and image mood is changed. The random baseline often either has dull or boosted and overexposed coloring, incorrect brightness and no visual appeal (cf. columns A-C). However, in some cases, the random enhancement by chance creates images that subjectively might be considered beautiful (cf. column D). Table 5.1 quantifies the similarity between the baselines and the original images and further shows the average NIMA score across our test set. Our method achieves the highest NIMA score, which shows that the optimization procedure we devised in equation 4.4 guides the filter

intensities towards an enhanced image which, according to NIMA, is of superior quality.

	Original	Photos	Ours	Expert A	Random
Original	1.000	0.933	0.759	0.610	0.561
Photos	0.933	1.000	0.764	0.597	0.533
Ours	0.759	0.764	1.000	0.567	0.534
Expert A	0.610	0.597	0.567	1.000	0.458
Random	0.561	0.533	0.534	0.458	1.000
Avg. NIMA Score	5.229	5.199	5.360	5.198	4.985

Table 5.1: SSIM and average NIMA score for the baselines.

5.1.2 Realization

We conducted the evaluation study with 51 voluntary participants, of whom 37 were male and 14 were female. All of the participants were under the age of 35 and had an academic background. The test group therefore is not representative, and further testing with larger groups of more diversity is required to confirm the obtained results.

In our evaluation study, participants were shown the three previously described baselines and our enhancement results, as well as the original, unedited image. We did not disclose which image was created by which method and randomly re-ordered the images in each iteration. The four edited images were displayed simultaneously at the top of the screen, while the original image was fixed at the bottom center. We asked the participants to evaluate the enhancement quality of each image. To this end, we provided an unnumbered low-to-high scale, onto which the users had to drag the images. The scale represented the difference in editing quality with respect to the unedited image. Each user was asked to rate 30 image quadruples. A screenshot of the evaluation GUI that was used in the study can be found in appendix C.

For reproducibility and to prevent score distortions induced by the use of different PC monitors and the subsequent variation in color representation, all participants used the same workstation to display the images. For the evaluation set, we randomly sampled 500 images from the MIT-Adobe FiveK test set and the photography expert, and let the unedited images pass through Apple Photos and our enhancement method.

5.1.3 Results

As each of the 51 participants ranked 30 randomly sampled image quadruples, we recorded a total of 1530 image ratings. On average, each image in the evaluation set was ranked three times. The histograms of the recorded user ratings for the respective baselines are illustrated in Figure 5.2.

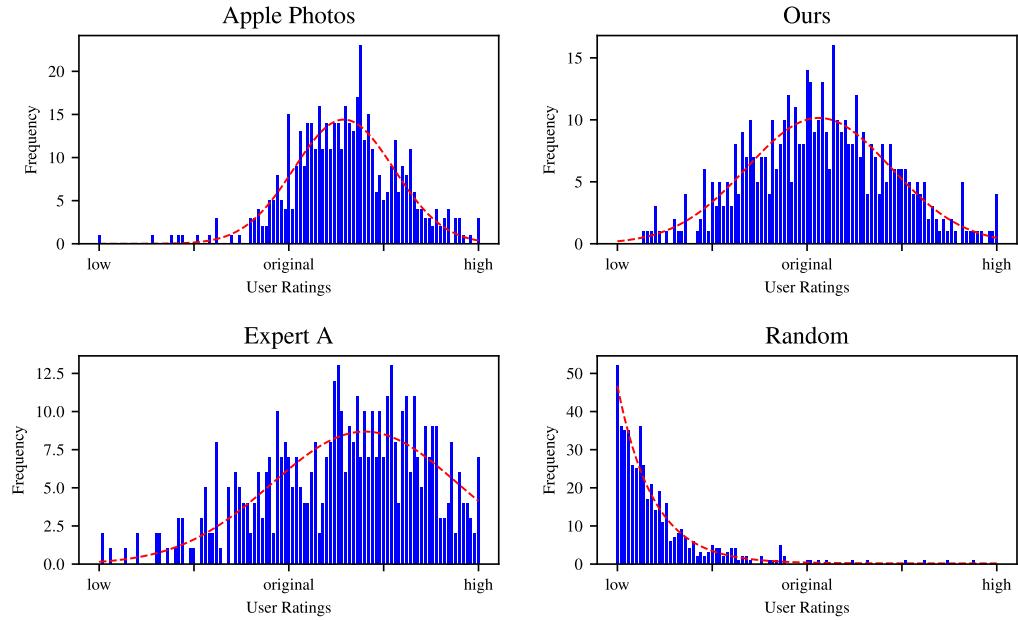


Figure 5.2: Relative results of the evaluation study. The histograms show the ratings assigned to the respective baseline on the scale provided in the GUI. For interpretability, we fitted a Gaussian distribution (red dashed line) to the data for Apple Photos, Ours and Expert A, and an exponential distribution to the data for Random.

As the resulting histograms show, the baselines Photos and Expert A outperformed our approach. The random baseline consistently received low ratings, which is what we expected. The ratings for Expert A show the largest diversity, which we attribute to the fact that the expert deliberately introduced large changes into the images. We assume that the participants subjectively either liked or disliked this enhancement style. Our enhancements received balanced scores. To quantify the relative ratings shown in Figure 5.2, we map the low-to-high scale the users were shown in the GUI to the interval [0, 10], where the worst-rated image is assigned the score 0.00. On this absolute scale, we then calculate the mean rating difference between the baselines. Table 5.2 shows these results and furthermore expresses the percentage of cases in which the baselines outperformed each other.

	Original	Photos	Ours	Expert A	Random
Original	- / 0.000	21.63 / -1.778	46.34 / -0.202	31.37 / -1.760	95.69 / 5.250
Photos	78.37 / 1.778	- / 0.000	64.31 / 1.560	45.95 / 0.018	95.75 / 7.028
Ours	53.66 / 0.202	35.69 / -1.576	- / 0.000	35.95 / -1.558	93.01 / 5.452
Expert A	68.63 / 1.760	54.05 / -0.018	64.05 / 1.558	- / 0.000	94.90 / 7.010
Random	4.31 / -5.250	2.25 / -7.028	6.99 / -5.452	5.1 / -7.010	- / 0.000

Table 5.2: Percentage of cases where one baseline outperformed the other and normed average distance between the baselines, respectively. Values are separated by a slash. The table is to be read in a row–column–way, e.g. Ours outperformed Expert A in 35.95% of all cases and the average, normed distance between Ours and Expert A is -1.558.

All following numbers are normalized and reference the absolute scale. Our method outperformed the random enhancement in 93% of all cases, which shows that NIMA is a valid means of guiding an image manipulation network towards good results. Participants preferred our enhancement result over the unedited original image in 53.7% of all cases. Interestingly, the baseline that received the highest average rating is Apple’s Photos, with an average improvement over the original image of 1.778 (the maximum possible score difference on the absolute scale is ± 10.0). Although participants preferred Expert A’s enhancement over Photos in 54% of all cases, the baseline’s averaged rating improvement over the original image is only 1.760 and thus marginally inferior to Photos. Figure 5.3 shows the normalized rating differences between the enhanced images and the original image.

To test the obtained results for statistical significance, we employ the Wilcoxon signed-rank test [88]. The null hypothesis is **H₀**: The discrepancy between related, paired samples follows a symmetric distribution with $\mu = 0.0$. In our case, this can be paraphrased as “two images stem from the same distribution of ratings, and thus are enhanced equally well”. For two given samples x_1 and x_2 from these distributions, the test score W can be calculated as

$$W = \sum_{i=1}^{N_{red}} [\text{sgn}(x_{2,i} - x_{1,i}) \cdot R_i] \quad , \quad (5.1)$$

where sgn is the sign function and N_{red} denotes the sample set, reduced by all sample pairs that satisfy the condition $|x_2 - x_1| = 0$. R_i expresses the rank index, which is computed by ranking the non-zero differences $|x_2 - x_1|$ in N_{red} in ascending order [88]. Using W, the z-score is computed as $W\sigma_W^{-1}$, where

$$\sigma_W = \sqrt{\frac{N_{red}(N_{red} + 1)(2N_{red} + 1)}{6}} \quad . \quad (5.2)$$

We reject **H₀** at the standard confidence level for statistical significance of $p = 5\%$. Reconsidering Figure 5.3, we evaluate the Wilcoxon signed-rank test on the normalized distributions for Photos and Expert A. This yields a p-value of 0.78, which

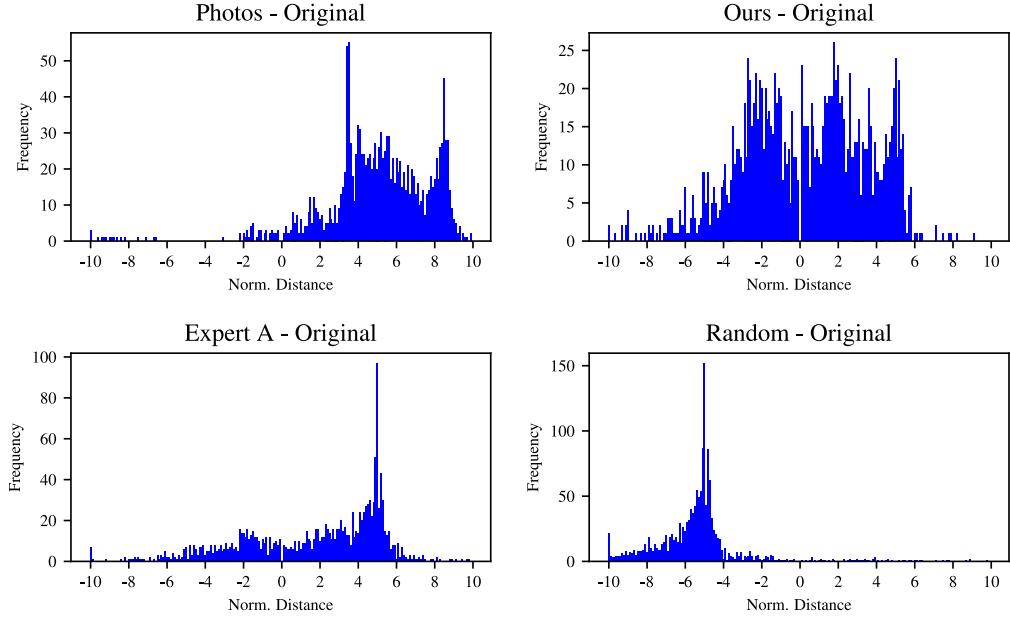


Figure 5.3: Normalized differences in image ranking with respect to the unedited original image. The peaks at ± 5.00 for Random and Expert A can be explained by the fact that these baselines often were rated close to the lowest and highest scale end on the GUI, respectively (cf. Fig. 5.2). Normalizing the relative scale in such a case yields values of approximately 0.00 for the random baseline, 10.00 for the expert’s enhancement, and 5.00 for the original image, resulting in a score difference of ± 5.00 .

allows us to conclude that the normalized rating scores for Photos and Expert A are not significantly different. Considering Figures 5.2 and 5.3, this suggests that the images are equally well enhanced. We furthermore investigate whether our approach received rankings that are significantly better than the unedited image scores. To this end, we calculate the distance in rating between our enhancement result and the worst-rated image, and the original image and the worst-rated image, respectively. Evaluating the Wilcoxon signed-rank test on these distances results in a p-value of 0.003. We therefore conclude that H_0 has to be rejected and confirm that our results significantly outperform the unedited images.

The obtained results clearly show that most users prefer the enhancements by Photos and Expert A. However, when inspecting the normalized rating scores, we found that the user ratings are extremely subjective and severely deviate from each other between participants. Roughly one third of our enhancement results (33.96%) received normalized scores that differ by more than 5.00 points between users, although the images are identical. The divergent scores prove the subjectivity of the ratings. Furthermore, a part of our results received scores that deviate highly from the original image, although our enhancement modified the source image only slightly. Figure 5.4 illustrates these discrepancies on selected examples.



Figure 5.4: Discrepancies in image ratings. The upper row shows the unedited, original image. The lower row shows our enhancement results and the normalized, averaged rating distance between our enhancement and the original image for different users. Note the large score differences, although our results do not differ much from the original.

Considering the evaluation results, we assume that the study participants, and users in general, prefer subtle over radical enhancements. This further becomes evident when looking at the distribution of user ratings with respect to the filter intensities. Figure 5.5 shows this relation for our evaluation experiment. Analysing the trend lines shows that the relative NIMA score rises with increasing filter intensities. This assumption is supported by the findings of section 4.3.1 (cf. Fig. 4.13). On the contrary, user ratings decline with higher filter intensities. Reconsidering Table 5.1, the high SSIM between Photos and the original images and the evaluation ratings for the Photos baseline reinforce this assumption. Setting $\gamma = 0.3$ and re-processing our evaluation set yields an SSIM of 0.828 between our enhancements and the original images. Following the above reasoning, this could potentially be indicative for better user ratings and thus signify that the resulting enhancement correlates better with the users' notion of aesthetic.

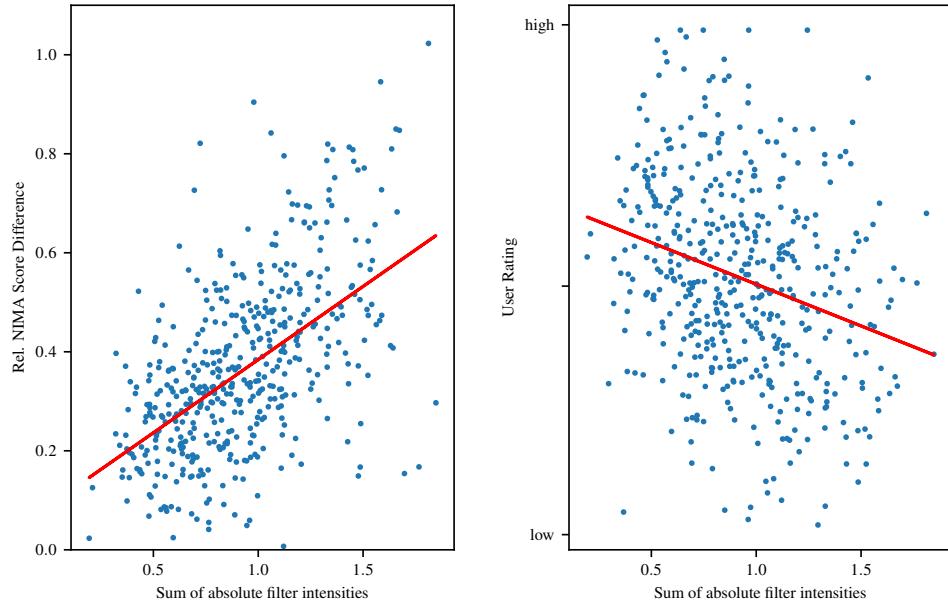


Figure 5.5: Relative NIMA scores and user ratings for our enhancement approach (blue), with respect to filter intensities. The sum of the absolute values of the filter intensities is shown on the abscissa. For interpretability, we fitted a linear function to the data using a non-linear least squares algorithm (red) provided by the SciPy package [89].

We furthermore suspect that additional biases might have subconsciously been introduced by the GUI layout and the Photos baseline. The GUI might have motivated users to place two images to the left of the original, and two images to the right, although this was never communicated during the explanation of the experiment. Auto-Enhance, the algorithm that created the Photos baseline, is also used in the automatic image enhancement of the iPhone. Users that use an iPhone therefore might have been biased towards the enhancement style of the Photos baseline. As we did not record any data on this, it is impossible to quantify the impact of this potential bias.

5.2 Experiments

The consecutive sections and the remainder of this chapter explain experiments and investigations that were carried out during the creation of this thesis and highlight potential future applications and extensions of the presented approach.

5.2.1 Local Gradient Application

As previously explained, the created method applies the backpropagated gradients globally to adjust the filter intensities. Consequently, each filter is applied uniformly and onto the entire image. To actually replicate the enhancement workflow of a photographer, the filter intensities would have to be set locally, to e.g. enhance dark corners or intensify details in the image.

In early stages of the project implementation, we experimented with local filter application. Considering that the filter layers share the input image's dimensions, it in fact is a logical step to apply the backpropagated gradients locally, as each pixel of the filter map receives a gradient associated with one pixel of the current output. The resulting gradient shape therefore is equal to the filter layers' shape of $224 \times 224 \times 8$, as the enhancement pipeline operates at the resolution 224×224 and we use eight filter layers. However, results were not compelling. The local, pixel-wise gradient application led to the creation of so-called “adversarial examples” [59] — images that delude NIMA into predicting a very high aesthetic score albeit having no visual appeal or aesthetic. This happened across images of varying semantic content and style, and across datasets. Figure 5.6 (column 3) shows a precedent of such an adversarial example: Although the image is degraded in quality and aesthetic, it receives an abnormally high NIMA score.

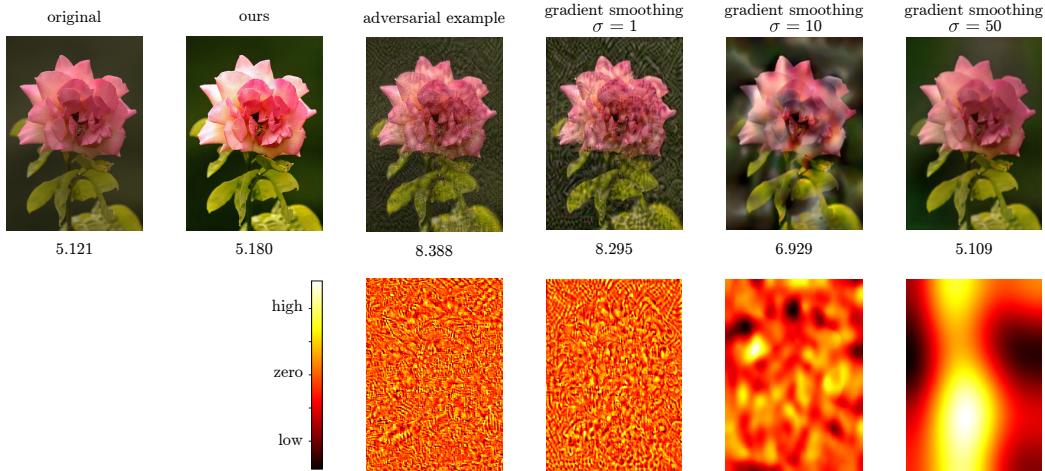


Figure 5.6: Local gradient application leads to the creation of adversarial examples. NIMA scores are given below the images. For brevity, we only display the gradient heatmap for Brightness. The other heatmaps were omitted, but show analogous results. Image from the MIT-Adobe FiveK dataset [65].

As the figure shows, the high-scoring image's gradient heatmap resembles random noise, which often is distinctive for adversarial examples [59]. The characteristic image blur and the streams and streaks that commonly are found in adversarial examples stem from extreme local differences in neighbouring gradients. We tried

to smooth the gradient field by convolving the pixel-wise gradients with a Gaussian kernel with different values for σ . Figure 5.6 shows that the results were not compelling: The gradient smoothing with low σ values did not increase image aesthetic, while higher values for σ created patchy spots where the filter was applied with high or low intensity. Even with very high smoothing values, the resulting image’s aesthetics are inferior to our enhancement.

Figure 5.6 shows that gradient smoothing reduces the adversarial effects of local gradient application. However, high smoothing values have to be applied. This is not constructive in the sense of local gradient application as a means of local image editing, as high smoothing values even out the gradients across larger image regions. For true local gradient application, we hence suggest that NIMA must be trained adversarially robust [86, 87]. As adversarial robustness is beyond the scope of this thesis, this remains an open research task for future work.

5.2.2 Timing

The runtime of the approach mainly depends on the inference time of the two networks that are used within the pipeline, and on the backpropagation. The CAN is a very compact and small network, and therefore is quicker in inference than NIMA. As CAN and NIMA inference and backpropagation both run in constant time and at the constant scale of 224×224 , the parameter that effectively determines the approach’s runtime is the number of optimization iterations per image. The resulting trade-off between high NIMA score — and thus increased image aesthetic — and required runtime is different for each image, as the optimization considers semantic content and photographic style. However, for most images, the filter intensities converge after 40-50 iterations (cf. Fig. 5.7). We therefore set the hyperparameter iterations to 50. Table 5.3 quantifies the results. Note that the initial image resolution does not affect optimization speed, as the image is rescaled to 224×224 within the pipeline. However, the final filter intensities are applied to the full-scale image at the end of the optimization cycle. Chen *et al.* state that CAN runtime scales linearly with the number of pixels [4]. Therefore, the runtime of the final CAN pass for the full-resolution image depends on the desired output resolution.

	CAN24	NIMA(VGG)	Backprop.	Combined(1)	Combined(50)
CPU	0.183	0.127	0.432	0.716	38.16
GPU	0.001	0.002	0.018	0.025	1.364

Table 5.3: Timing for the different pipeline parts, reported in seconds. Measured on an Intel i7-7700k CPU @ 3.60GHz and an NVIDIA RTX2080 Ti GPU for a 1080p Full-HD image. Combined(1) is one optimization epoch, Combined(50) is the full optimization cycle, including the final application of the filter intensities on the full-resolution image.

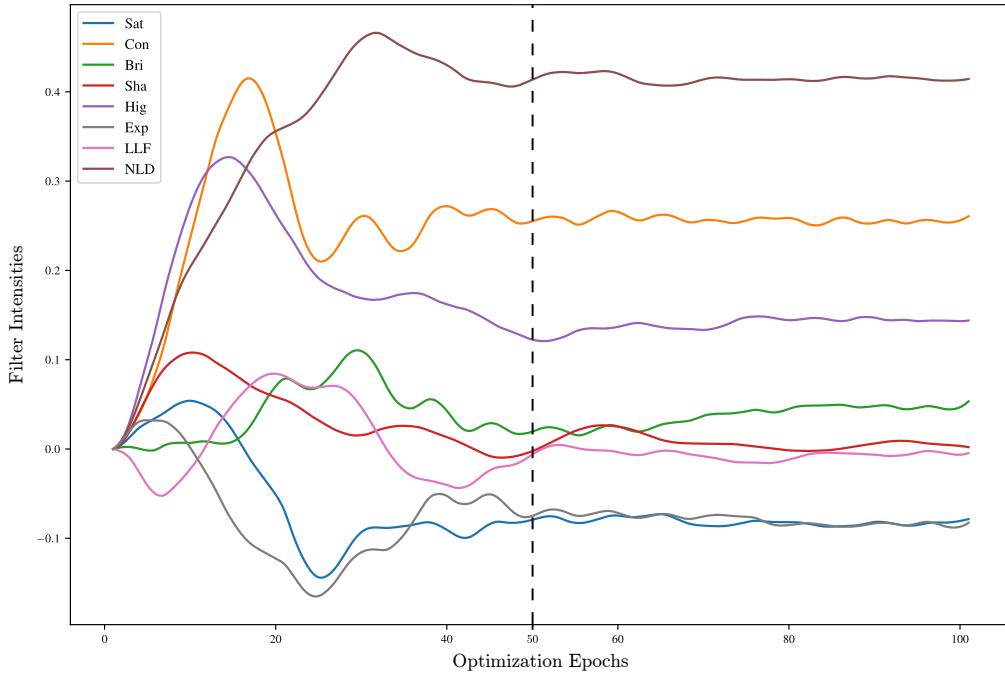


Figure 5.7: Filter intensities with respect to optimization epochs for the rightmost image in Figure 5.4. The vertical, black line marks 50 epochs, which we define as the timestamp where the filter intensities are mostly settled. This hyperparameter defaults to 50 iterations, but can individually be set for each image.

5.2.3 Extension: Video Editing

Naturally, our enhancement approach can also be applied to the task of video editing. To this end, we split stock video footage into single frames, and apply the enhancement pipeline onto these frames. This technique works best and fastest for continuous motion videos, where subsequent video frames vary only slightly in content, as the initial filter combination that has been computed for the first video frame can simply be applied to all following frames. For fast-moving video content and cut scenes, we compute the structural similarity between video frames. We use the SSIM between two subsequent frames as an indicator to detect a cut scene and re-compute the optimal filter intensities for the next scene. We tested the video-processing capabilities of our approach on online stock video footage. The results are publicly available on GitHub² and YouTube³. The enhanced videos show that our approach generalizes across datasets and is universally applicable.

²NICER - Neural Image Correction & Enhancement Routine, Michael Fischer, 2020.
<https://github.com/mr-Mojo/NICER>, retrieved on 30.01.2020

³NICER - Neural Image Correction & Enhancement Routine,
<https://youtu.be/7DkAy7NYcu0>, retrieved on 30.01.2020

As the user evaluation and the illustrated results show, our approach yields images of good enhancement quality. However, these enhancements do not always correlate with the human perception of aesthetic. The following chapter will discuss the overall quality of the approach and highlight potential starting points for future research.

6

Discussion

Chapters 4 and 5 show that the devised approach works well and yields images of good enhancement quality. The fact the enhancement is assessed by NIMA leads to results that generally show high contrast, saturated colors, increased details and a certain yellowness, which is induced by high values of the NLD filter. The resulting enhancement style and strength varies with the source image and adapts to content and lighting of the image. Our enhancement strategy produces consistent results for well-illuminated images that significantly outperform the unedited images.

However, a major drawback of our method is the fact that images of sub-optimal brightness or lighting settings are not optimized correctly (cf. Fig. 4.14). This can be attributed to NIMA’s biased score predictions which we use to score the enhancement results. NIMA was trained on the AVA dataset, which mostly contains well-lit images that were taken by experienced photographers. Therefore, the decrease in aesthetic score that intuitively is expected for bad illumination cannot be observed in NIMA’s predictions, as it is not present in the dataset. We confirm this with our experiments (cf. Fig. 4.13) and show that different filters affect NIMA’s score predictions in a non-uniform, non-linear way. We further observe that NIMA is oversensitive to contrast- and detail-enhancing filters.

These are significant problems for our approach, as NIMA is a key component of the enhancement pipeline. Although Table 5.1 shows that our enhancement results received the highest average NIMA score and therefore confirms that the optimization routine yields good intensity combinations, our enhanced images were outperformed by two other baselines and thus can not be optimally enhanced. We hence suspect that — while the predicted NIMA scores for unedited images might

Discussion

be accurate — the score predictions for edited images show only limited correlations with the human notion of aesthetic. Our experiments confirmed this (cf. Fig. 4.13).

Further investigation showed that the loss landscape of NIMA predictions is not, as we expected, uniform and even, but has steep gradients and “holes” in it. In rare cases, the optimization converges towards such an anomaly and cannot recover, creating an image of relatively high NIMA score but with no visual appeal (cf. Fig. 4.12). This is reproducible and happens across images and optimizers. Indeed, it has been observed that the absence of skip-connections in neural networks (e.g. in VGG16, which we use as NIMA base classifier) increases the probability of pointy and jagged loss landscapes [90]. In order to mitigate this problem, we hence suggest to smooth out the loss landscape. According to Li *et al.* [90], this could be achieved by using a NIMA feature extractor with skip-connections (e.g. DenseNet [91]). Another possibility to achieve a more robust NIMA classifier is adversarial training [59, 86], which previously has been shown to smooth out gradient geometry [92]. Adversarial training also might permit local gradient application — and subsequently local filter application — and therefore further improve our approach’s enhancements. Both of these suggestions are starting points for potential future work.

In our presented approach, NIMA’s non-sensitivity towards image illumination is addressed by the ABN preprocessor, which was designed during the course of this thesis. ABN automatically corrects image exposure and brightness and thus increases the probability of an enhancement outcome with high aesthetic. However, ABN is merely a preprocessing step in our enhancement pipeline and does not directly alter NIMA’s predictions or gradient geometry.

A further point of improvement lies in the distribution of score ratings within the AVA dataset. As the distribution of image ratings is largely Gaussian (cf. Fig. 4.9, [34]), very low and high ratings are highly under-represented in the dataset. As a consequence, NIMA does not learn to accurately predict image aesthetics correctly for very good or bad images. In part, we solved this problem by crudely re-training NIMA’s dense layer, as described in section 4.3.1. Using a more diverse and broadly rated dataset or a more sophisticated re-training technique could potentially mitigate this issue.

Nevertheless, the enhancement results of our approach are of good quality: When evaluating the resulting images in a qualitative user study, we scored above the unedited image in the majority of cases. We did not outperform the enhancement results of Apple Photos and the expert photographer. However, to put the results into perspective, these are two very strong baselines: The professional photographer can rely on many years of experience, human intuition and knowledge based on his qualification and the received training and education. Apple, on the other hand, is renowned for its competence in multimedia processing, and the

Photos application includes all of our photographic filters and several additional operators. Moreover, Photos uses sophisticated processing steps like spatial image editing, local filter application and face detection (cf. section 5.1.1). We do not use any of these steps and apply all filters globally. Our results could potentially be improved further by optimizing with higher values for γ , which penalizes high filter intensities more severely and thus yields images that are closer to the original.

7

Conclusion

We proposed a novel approach for automatic image enhancement. Our approach uses the quality assessment network NIMA as a metric to guide the filter intensities of an image manipulation network towards perceptually pleasing results, eliminating the need for manual, user-guided image editing. We investigated important design decisions of the enhancement pipeline and devised ABN, an automatic brightness corrector, as a way of circumventing NIMA’s insensitivity to suboptimal illumination. We evaluated our results in a user study, where we outperformed a random enhancement created by our pipeline by a large margin. Albeit our enhancement results were outperformed by two other baselines and thus cannot be considered “optimal”, our findings show that NIMA can indeed be used as a metric for automatic image enhancement. Our approach runs in reasonable compute time and works on images of arbitrary size and content, making it a versatile and useful tool for image enhancement applications in real-world scenarios.

In preparation for future extensions of our enhancement pipeline, we studied the loss landscape of our metric NIMA and outlined various starting points for potential future work. Amongst these aspects is the adversarial training of the NIMA base classifier and the smoothing of the NIMA loss landscape. We further propose to use scene understanding and sentiment analysis on images and potential accompanying texts to classify image mood and set the initial filter intensities accordingly.

Bibliography

- [1] K. Fukushima, “Neocognitron: A hierarchical neural network capable of visual pattern recognition”, *Neural networks*, vol. 1, no. 2, pp. 119–130, 1988.
- [2] J. R. Jensen and K. Lulla, “Introductory digital image processing: A remote sensing perspective”, 1987.
- [3] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, *CoRR*, vol. abs/1409.1556, 2014.
- [4] Q. Chen, J. Xu, and V. Koltun, “Fast image processing with fully-convolutional networks”, in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2497–2506.
- [5] Idealo. (2019). Convolutional neural networks to predict the aesthetic and technical quality of images, [Online]. Available: <https://idealo.github.io/image-quality-assessment/> (visited on 01/30/2020).
- [6] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, *et al.*, “Image quality assessment: From error visibility to structural similarity”, *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.
- [7] Tripadvisor. (2019). Bookings and traveler engagement driven by management actions, [Online]. Available: <https://www.tripadvisor.com/TripAdvisorInsights/w613> (visited on 01/30/2020).
- [8] H. Talebi and P. Milanfar, “Nima: Neural image assessment”, *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3998–4011, 2018.
- [9] Statista. (2019). Distribution of instagram users worldwide as of july 2019, by age and gender, [Online]. Available: <https://www.statista.com/statistics/248769/age-distribution-of-worldwide-instagram-users> (visited on 01/30/2020).
- [10] H. Talebi and P. Milanfar, “Learned perceptual image enhancement”, in *2018 IEEE International Conference on Computational Photography (ICCP)*, IEEE, 2018, pp. 1–13.
- [11] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, “Gradient-based learning applied to document recognition”, *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

Bibliography

- [13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>, (visited on 01/30/2020).
- [14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database”, in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.
- [15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions”, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.
- [17] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilennets: Efficient convolutional neural networks for mobile vision applications”, *arXiv preprint arXiv: 1704.04861*, 2017.
- [18] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.
- [19] T. Sercu and V. Goel, “Dense prediction on sequences with time-dilated convolutions for speech recognition”, *arXiv preprint arXiv:1611.09288*, 2016.
- [20] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, “Loss functions for image restoration with neural networks”, *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47–57, 2016.
- [21] H. Talebi and P. Milanfar, “Fast multilayer laplacian enhancement”, *IEEE Transactions on Computational Imaging*, vol. 2, no. 4, pp. 496–509, 2016.
- [22] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions”, *arXiv preprint arXiv:1511.07122*, 2015.
- [23] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation”, *ArXiv*, vol. abs/1505.04597, 2015.
- [24] J. Dai, Y. Li, K. He, and J. Sun, “R-fcn: Object detection via region-based fully convolutional networks”, in *Advances in neural information processing systems*, 2016, pp. 379–387.
- [25] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.
- [26] L. Hou, C.-P. Yu, and D. Samaras, “Squared earth movers distance loss for training deep neural networks on ordered-classes”, in *NIPS Workshop*, 2017.
- [27] P. D’Oro and E. Nasca. (2019). An empirical evaluation of convolutional neural networks for image enhancement, [Online]. Available: <https://github.com/proceduralia/pytorch-neural-enhance> (visited on 01/30/2020).
- [28] Q. Huynh-Thu and M. Ghanbari, “Scope of validity of psnr in image/video quality assessment”, *Electronics letters*, vol. 44, no. 13, pp. 800 –801, 2008.

Bibliography

- [29] R. Maini and H. Aggarwal, “A comprehensive review of image enhancement techniques”, *arXiv preprint arXiv:1003.4053*, 2010.
- [30] G. Ekman, “Weber’s law and related functions”, *The Journal of Psychology*, vol. 47, no. 2, pp. 343–352, 1959.
- [31] E. Levina and P. Bickel, “The earth mover’s distance is the mallows distance: Some insights from statistics”, in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, IEEE, vol. 2, 2001, pp. 251–256.
- [32] Statista. (2019). Smartphones cause photography boom, [Online]. Available: <https://www.statista.com/chart/10913/number-of-photos-taken-worldwide> (visited on 01/30/2020).
- [33] L. Zhang, L. Zhang, X. Mou, and D. Zhang, “A comprehensive evaluation of full reference image quality assessment algorithms”, in *2012 19th IEEE International Conference on Image Processing*, IEEE, 2012, pp. 1477–1480.
- [34] N. Murray, L. Marchesotti, and F. Perronnin, “Ava: A large-scale database for aesthetic visual analysis”, in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2012, pp. 2408–2415.
- [35] T. Rukkanchanunt, T. Masayuki, and M. Okutomi, “Image enhancement framework for low-resolution thermal images in visible and lwir camera systems”, Oct. 2017, p. 9. doi: 10.1117/12.2277393.
- [36] N. Ponomarenko, O. Ieremeiev, V. Lukin, K. Egiazarian, L. Jin, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti, *et al.*, “Color image database tid2013: Peculiarities and preliminary results”, in *european workshop on visual information processing (EUVIP)*, IEEE, 2013, pp. 106–111.
- [37] H. Sheikh, *Live image quality assessment database release 2*, <http://live.ece.utexas.edu/research/quality/>, (last visited 01/30/2020),, 2005.
- [38] K. Sheng, W. Dong, H. Huang, C. Ma, and B.-G. Hu, “Gourmet photography dataset for aesthetic assessment of food images”, Dec. 2018, pp. 1–4. doi: 10.1145/3283254.3283260.
- [39] L. Kang, P. Ye, Y. Li, and D. Doermann, “Convolutional neural networks for no-reference image quality assessment”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1733–1740.
- [40] S. Bosse, D. Maniry, T. Wiegand, and W. Samek, “A deep neural network for image quality assessment”, in *2016 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2016, pp. 3773–3777.
- [41] X. Lu, Z. Lin, H. Jin, J. Yang, and J. Z. Wang, “Rating image aesthetics using deep learning”, *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 2021–2034, 2015.
- [42] B. Jin, M. V. O. Segovia, and S. Süsstrunk, “Image aesthetic predictors based on weighted cnns”, in *2016 IEEE International Conference on Image Processing (ICIP)*, Ieee, 2016, pp. 2291–2295.
- [43] Y. Kao, C. Wang, and K. Huang, “Visual aesthetic quality assessment with a regression model”, in *2015 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2015, pp. 1583–1587.

Bibliography

- [44] S. Dhar, V. Ordonez, and T. L. Berg, “High level describable attributes for predicting aesthetics and interestingness”, in *CVPR 2011*, IEEE, 2011, pp. 1657–1664.
- [45] S. Kong, X. Shen, Z. Lin, R. Mech, and C. Fowlkes, “Photo aesthetics ranking network with attributes and content adaptation”, in *European Conference on Computer Vision*, Springer, 2016, pp. 662–679.
- [46] X. Fu, J. Yan, and C. Fan, “Image aesthetics assessment using composite features from off-the-shelf deep models”, in *2018 25th IEEE International Conference on Image Processing (ICIP)*, IEEE, 2018, pp. 3528–3532.
- [47] P. Milanfar, “A tour of modern image filtering: New insights and methods, both practical and theoretical”, *IEEE signal processing magazine*, vol. 30, no. 1, pp. 106–128, 2012.
- [48] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images.”, in *Iccv*, vol. 98, 1998, p. 2.
- [49] B. Hosseini, N. Nigam, and J. M. Stockie, “On regularizations of the dirac delta distribution”, *Journal of Computational Physics*, vol. 305, pp. 423–447, 2016.
- [50] D. R. Brownrigg, “The weighted median filter”, *Communications of the ACM*, vol. 27, no. 8, pp. 807–818, 1984.
- [51] S. Paris, S. W. Hasinoff, and J. Kautz, “Local laplacian filters: Edge-aware image processing with a laplacian pyramid.”, *ACM Trans. Graph.*, vol. 30, no. 4, p. 68, 2011.
- [52] Q. Yang, K.-H. Tan, and N. Ahuja, “Real-time o (1) bilateral filtering”, in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 557–564.
- [53] M. Gharbi, Y. Shih, G. Chaurasia, J. Ragan-Kelley, S. Paris, and F. Durand, “Transform recipes for efficient cloud photo enhancement”, *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 228, 2015.
- [54] M. Egmont-Petersen, D. de Ridder, and H. Handels, “Image processing with neural networks—a review”, *Pattern recognition*, vol. 35, no. 10, pp. 2279–2301, 2002.
- [55] L. Xu, J. Ren, Q. Yan, R. Liao, and J. Jia, “Deep edge-aware filters”, in *International Conference on Machine Learning*, 2015, pp. 1669–1678.
- [56] Z. Yan, H. Zhang, B. Wang, S. Paris, and Y. Yu, “Automatic photo adjustment using deep neural networks”, *ACM Transactions on Graphics (TOG)*, vol. 35, no. 2, p. 11, 2016.
- [57] S. Liu, J. Pan, and M.-H. Yang, “Learning recursive filters for low-level vision via a hybrid neural network”, in *European Conference on Computer Vision*, Springer, 2016, pp. 560–576.
- [58] S. Santurkar, D. Tsipras, B. Tran, A. Ilyas, L. Engstrom, and A. Madry, “Computer vision with a single (robust) classifier”, *arXiv preprint arXiv: 1906.09453*, 2019.
- [59] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks”, *arXiv preprint arXiv:1312.6199*, 2013.
- [60] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

Bibliography

- [61] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms”, *Physica D: nonlinear phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.
- [62] D. Berman, S. Avidan, *et al.*, “Non-local image dehazing”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1674–1682.
- [63] K. He, J. Sun, and X. Tang, “Single image haze removal using dark channel prior”, *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 12, pp. 2341–2353, 2010.
- [64] B. Wu, W. Chen, Y. Fan, Y. Zhang, J. Hou, J. Huang, W. Liu, and T. Zhang, “Tencent ml-images: A large-scale multi-label image database for visual representation learning”, *arXiv preprint arXiv:1901.01703*, 2019.
- [65] V. Bychkovsky, S. Paris, E. Chan, and F. Durand, “Learning photographic global tonal adjustment with a database of input / output image pairs”, in *The Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
- [66] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Mallochi, T. Duerig, *et al.*, “The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale”, *arXiv preprint arXiv:1811.00982*, 2018.
- [67] L. Challenging Technologies. (2019). Dpchallenge, [Online]. Available: <https://www.dpchallenge.com/> (visited on 01/30/2020).
- [68] Y. Hu, H. He, C. Xu, B. Wang, and S. Lin, “Exposure: A white-box photo post-processing framework”, *ACM Transactions on Graphics (TOG)*, vol. 37, no. 2, p. 26, 2018.
- [69] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training gans”, in *Advances in neural information processing systems*, 2016, pp. 2234–2242.
- [70] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein gan”, *arXiv preprint arXiv:1701.07875*, 2017.
- [71] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, “Deep reinforcement learning that matters”, in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [72] G. Bradski, “The OpenCV Library”, *Dr. Dobb’s Journal of Software Tools*, 2000.
- [73] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski, *Kornia: An open source differentiable computer vision library for pytorch*, 2019. [Online]. Available: <https://arxiv.org/pdf/1910.02190.pdf> (visited on 01/30/2020).
- [74] E. Riba, M. Fathollahi, W. Chaney, E. Rublee, and G. Bradski, *Torchgeometry: When pytorch meets geometry*, 2018. [Online]. Available: https://drive.google.com/file/d/1xiao1Xj9WzjJ08YY_nYwsthE-wxfyfhG/view?usp=sharing (visited on 01/30/2020).
- [75] M. Aubry, S. Paris, S. W. Hasinoff, J. Kautz, and F. Durand, “Fast and robust pyramid-based image processing”, 2011.
- [76] A. Clark and Contributors. (2019). Python imaging library, [Online]. Available: <https://pypi.org/project/Pillow> (visited on 01/30/2020).

Bibliography

- [77] M. Aubry, S. Paris, S. W. Hasinoff, J. Kautz, and F. Durand, “Fast local laplacian filters: Theory and applications”, *ACM Transactions on Graphics (TOG)*, vol. 33, no. 5, p. 167, 2014.
- [78] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization”, *arXiv preprint arXiv:1412.6980*, 2014.
- [79] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, “In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 607–12 616.
- [80] A. Y. Ng, “Feature selection, l 1 vs. l 2 regularization, and rotational invariance”, in *Proceedings of the twenty-first international conference on Machine learning*, ACM, 2004, p. 78.
- [81] R. Hecht-Nielsen, “Theory of the backpropagation neural network”, in *Neural networks for perception*, Elsevier, 1992, pp. 65–93.
- [82] J. Kiefer, J. Wolfowitz, *et al.*, “Stochastic estimation of the maximum of a regression function”, *The Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 462–466, 1952.
- [83] A. Alfonsi and B. Jourdain, “Lifted and geometric differentiability of the squared quadratic wasserstein distance”, *arXiv preprint arXiv: 1811.07787*, 2018.
- [84] N. Qian, “On the momentum term in gradient descent learning algorithms”, *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.
- [85] A. Nemirovskii and Y. Nesterov, “Interior point polynomial algorithms in convex programming”, *Society for Industrial and Applied Mathematics*, 1994.
- [86] S. Gu and L. Rigazio, “Towards deep neural network architectures robust to adversarial examples”, *arXiv preprint arXiv:1412.5068*, 2014.
- [87] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, “Certified robustness to adversarial examples with differential privacy”, in *2019 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2019, pp. 656–672.
- [88] B. Rosner, R. J. Glynn, and M.-L. T. Lee, “The wilcoxon signed rank test for paired comparisons of clustered data”, *Biometrics*, vol. 62, no. 1, pp. 185–192, 2006.
- [89] T. E. Oliphant, “Python for scientific computing”, *Computing in Science & Engineering*, vol. 9, no. 3, pp. 10–20, 2007.
- [90] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, “Visualizing the loss landscape of neural nets”, in *Advances in Neural Information Processing Systems*, 2018, pp. 6389–6399.
- [91] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.
- [92] J. Xu, D. A. Yap, and V. U. Prabhu, “Understanding adversarial robustness through loss landscape geometries”,

Appendix

A Network Architectures

MobileNet Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Table 1: MobileNet architecture. Adapted from [17].

Appendix

CAN Architecture

Layer	1	2	3	4	5	6	7	8
Convolution	3×3	3×3	3×3	3×3	3×3	3×3	3×3	1×1
Dilation	1	1	2	4	8	16	1	1
Truncation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Receptive field	3×3	5×5	9×9	17×17	33×33	65×65	67×67	67×67
Output channels								
Basic	C	C	C	C	C	C	C	C
Large	$2C$	$2C$	$4C$	$8C$	$16C$	$32C$	$32C$	C

Table 2: Context Aggregation Network architecture. Adapted from [22].

B Metrics and Measurements

PSNR and Compression

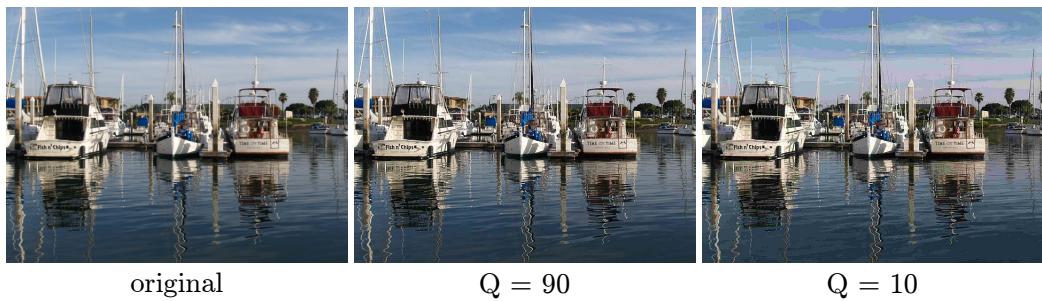


Figure 1: PSNR values for images with different compression strength: Q = 90: PSNR = 45.53, Q = 10: PSNR = 31.45. Compression artefacts can best be seen in the sky. Figure adapted from Wikipedia. https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio, retrieved on 30.01.2020.

Image Manipulation Operator Performance

Method	ROF	TV- L^1	L_0	RTV	Tone	Detail	Dehaze (NL)	Dehaze (DC)	Style	Pencil
Reference	18,598	22,181	7,053	10,411	10,268	1,190	6,114	7,983	6,271	4,947
BGU-fast	382	458	187	323	217	186	232	227	739	251
BGU-opt	2,436	2,472	2,321	2,377	2,271	2,240	2,286	2,281	2,793	2,305
Xu et al.	5,493	5,493	5,493	5,493	5,493	5,493	5,493	5,493	5,493	5,493
Johnson et al.	203	203	203	203	203	203	203	203	203	203
Liu et al.	458	458	458	458	458	458	458	458	458	458
Isola et al.	198	198	198	198	198	198	198	198	198	198
Isola et al. +BGU	2,352	2,352	2,352	2,352	2,352	2,352	2,352	2,352	2,352	2,352
Ours	190	190	190	190						

Table 3: Operator performance on the MIT-Adobe FiveK test set in milliseconds, as reported by Chen *et al.* (“Ours”), [4]. For the measurements, images were rescaled to 1080p. The employed workstation featured an Intel i7-5960X 3.0GHz CPU and an NVIDIA TitanX GPU. Table adapted from [4].

Performance Comparison for CAN Architectures

Method	Rudin-Osher-Fatemi			TV- L^1			L_0 smoothing			Relative total variation			Multiscale tone		
	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM
FCN-8s	54.6	32.39	0.916	38.8	33.42	0.958	89.1	30.05	0.918	40.2	33.65	0.972	357.9	24.30	0.714
Encoder-decoder	0.5	51.96	0.999	2.3	45.22	0.995	12.6	37.27	0.992	3.6	43.06	0.997	3.1	43.81	0.997
Plain	0.7	50.06	1.000	1.9	46.91	0.995	27.9	33.90	0.980	9.2	38.85	0.983	12.8	37.66	0.989
CAN32	0.6	51.29	0.999	3.7	43.44	0.993	12.6	37.32	0.988	3.8	42.99	0.994	3.8	42.57	0.997
CAN32+BN	69.2	31.30	0.963	86.3	30.22	0.950	136.5	27.81	0.927	95.6	29.76	0.955	128.0	28.12	0.953
CAN24+AN	0.6	51.24	0.999	4.3	42.72	0.992	14.9	36.50	0.983	4.4	42.45	0.993	6.3	40.42	0.995
CAN32+AN	0.6	51.86	0.999	3.0	44.57	0.993	12.1	37.48	0.987	3.3	43.55	0.995	4.9	41.60	0.996
CAN32+AN+Single	20.9	35.16	0.978	26.0	34.28	0.965	50.4	31.20	0.948	34.4	33.05	0.958	51.0	31.24	0.967
Method	Detail manipulation			Nonlocal dehazing			Dark-channel dehazing			Style transfer			Pencil drawing		
	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM
FCN-8s	413.7	23.00	0.689	294.3	24.53	0.788	296.9	24.51	0.806	754.3	19.64	0.556	1101.6	18.14	0.735
Encoder-decoder	83.2	29.26	0.949	101.2	30.11	0.970	142.3	28.37	0.966	593.4	20.66	0.827	836.4	19.33	0.810
Plain	313.3	23.37	0.863	135.7	28.38	0.952	186.9	27.67	0.957	1712.0	15.99	0.689	1296.9	17.68	0.788
CAN32	19.7	35.39	0.981	102.5	30.09	0.969	154.7	28.31	0.965	298.1	23.81	0.854	577.3	21.02	0.826
CAN32+BN	116.4	28.19	0.934	281.7	25.35	0.918	192.1	27.07	0.923	146.1	27.08	0.900	47.4	31.55	0.865
CAN24+AN	48.0	31.52	0.965	133.2	28.25	0.965	74.6	31.24	0.965	258.4	24.37	0.880	45.7	31.70	0.861
CAN32+AN	29.3	33.66	0.966	84.2	30.46	0.970	53.0	32.76	0.974	129.8	27.62	0.913	39.6	32.31	0.869
CAN32+AN+Single	72.7	29.67	0.938	137.2	27.99	0.951	212.7	26.06	0.932	345.7	23.27	0.850	152.3	26.72	0.825

Table 4: Performance comparison on the approximation accuracy of image manipulation operators for different versions of the CAN network. Values are reported across the MIT-Adobe FiveK test set. Table adapted from [4].

C Supplement Material

Approach Hyperparameters

- Epochs: 100
- Optimizer: SGD
- Learning Rate: 0.05
- Nesterov Momentum: 0.9
- γ : 0.1
- Final Resolution: 1080p

Artistic Imagery as considered by ABN



Figure 2: Artistic imagery, as considered by the ABN algorithm. For single-object photography on black or white ground, no enhancement is conducted, as images in this style usually are intended to be very bright or dark. Images from the AVA dataset [34].

Adaptive Brightness Normalization

```

Input: original image I
Output: brightness-normalized image  $\hat{I}$ 
initialization:  $\tau = 30$                                 // brightness threshold

P = perceivedBrightness(I)
if  $P \notin (128 \pm \tau)$  then
    if not isBlackImage(I) and not isWhiteImage(I) then
        if  $P < 128 - \tau$  then
            // image too dark
            if  $P < 33$  then
                |  $\hat{I} = \text{equalizeHist}(\alpha = 1.1, \beta = 0.0)$ 
            else if  $P < 70$  then
                | shiftFactor = 20
                |  $\hat{I} = \text{shiftHSV}(I, \text{shiftFactor})$ 
                | while PSNR(I,  $\hat{I}$ ) < 30.0 do
                    |   shiftFactor -= 1.0
                    |    $\hat{I} = \text{shiftHSV}(I, \text{shiftFactor})$ 
                end
            else
                |  $\hat{I} = \text{equalizeHist}(\alpha = 1.3, \beta = 0.0)$ 
            end
            return  $\hat{I}$ 
        else
            // image too bright
            clipPercentage = 5.0
             $\hat{I} = \text{clipHistogram}(I, \text{clipPercentage})$ 
            while SSIM(I,  $\hat{I}$ ) < 0.80 do
                | clipPercentage /= 10
                |  $\hat{I} = \text{clipHistogram}(I, \text{clipPercentage})$ 
            end
            return  $\hat{I}$ 
        end
    else
        | return I           // mostly black or white, no normalization
    end
else
    | return I           // brightness level okay, no normalization
end

```

Algorithm 1: Adaptive Brightness Normalization (ABN) routine. The threshold values PSNR < 30.0 and SSIM < 0.80 and the values for α and β were determined empirically.

Selected Enhancement Results



Figure 3: Selected enhancement results. Original images (left column) from the MIT-Adobe FiveK dataset [65].

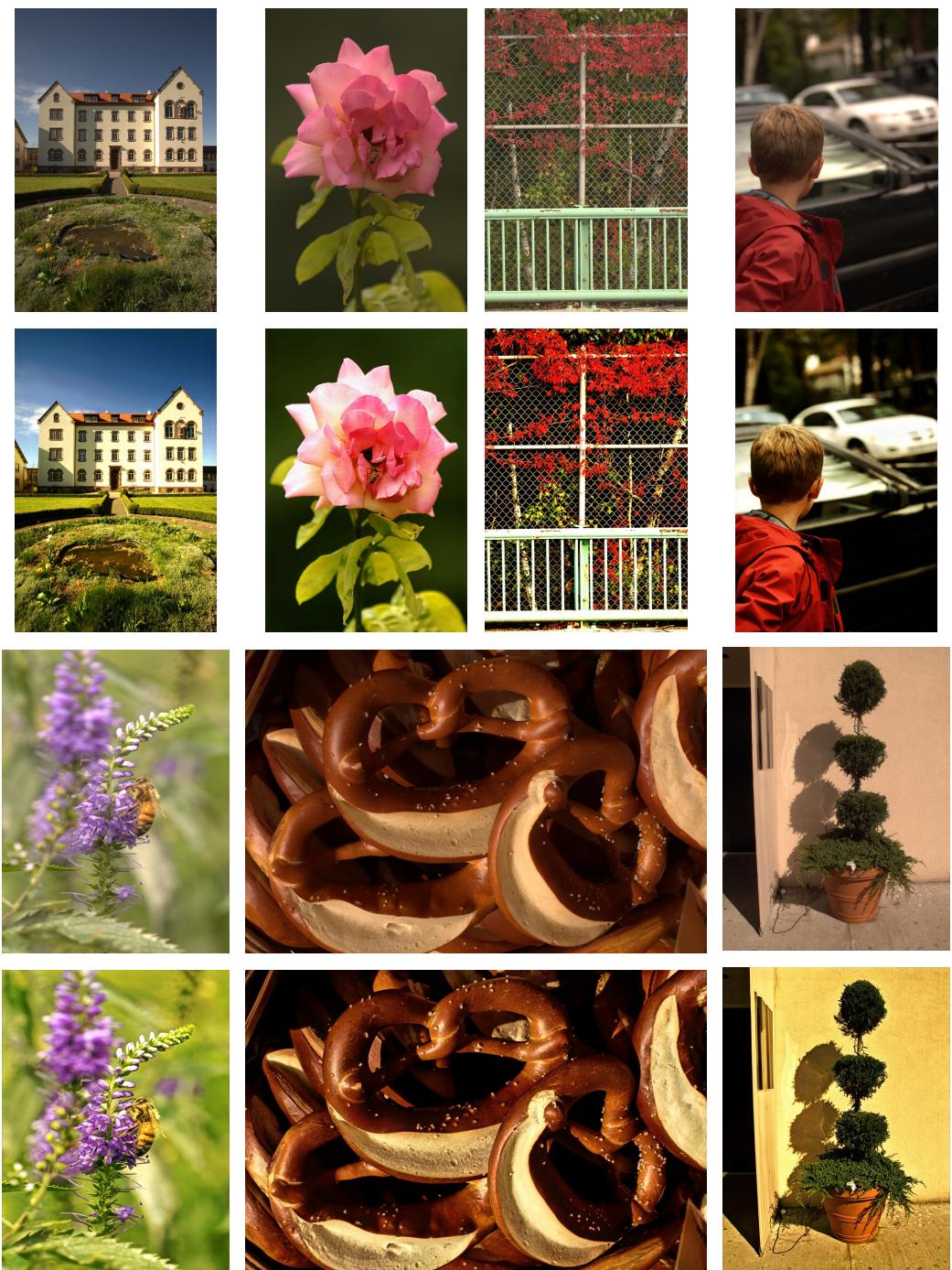


Figure 4: Selected enhancement results. Original images (upper row) from the MIT-Adobe FiveK dataset [65].

Appendix

Evaluation GUI

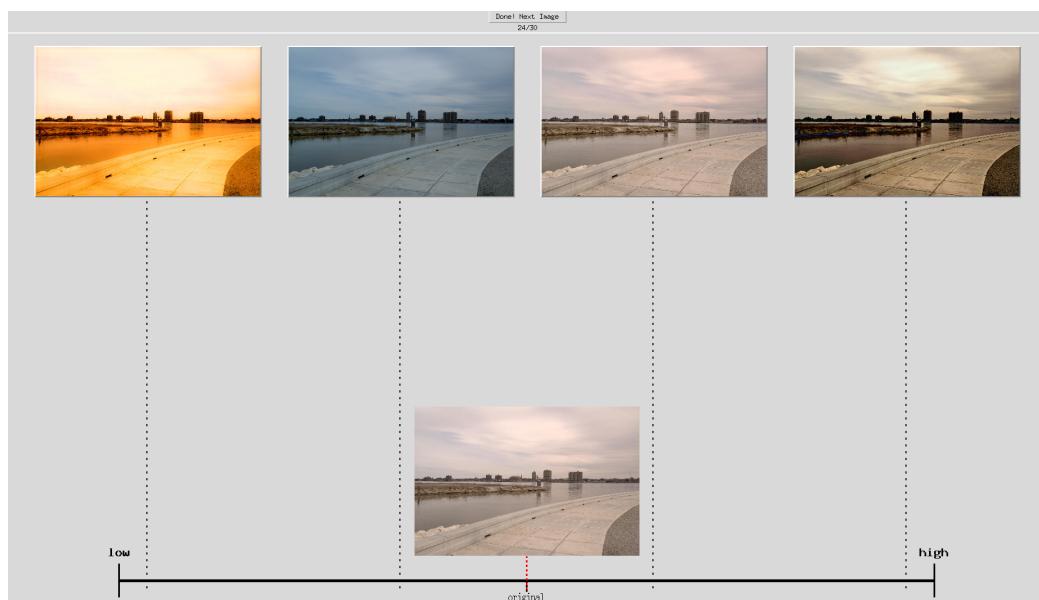


Figure 5: The GUI used for the evaluation study. The upper four images are draggable, while the lower, original image is fixed to the bottom center of the scale.

Eigenständigkeitserklärung

Hiermit versichere ich, die vorliegende Masterarbeit selbstständig angefertigt und keine anderen als die von mir kenntlich gemachten Quellen und Hilfsmittel benutzt zu haben. Ich habe diese Arbeit keiner anderen Prüfungsbehörde zur Erlangung eines akademischen Grades vorgelegt.

Würzburg,
.....
Michael Fischer