EURECOM

POLITECNICO DI TORINO

Double Master Degree
in Data Science and Data Engineering

Master Thesis

Feature aware image sampling for CNN-based image quality enhancement algorithms

Supervisors
prof. Paolo Garza
prof. Pietro Michiardi
Luca Alberto Rizzo

Candidate
Giulia Bonino

Academical year 2024-2025

Summary

In classical machine learning algorithms, for example those used for classification tasks, a common issue arises when dealing with an unbalanced training dataset. When certain classes are under-represented, it can result in poor performance not only on those classes but also on average. This issue also occurs in Image Enhancement algorithms, where some types of images may be under-represented in the training dataset. Consequently, the performance on these types of images tends to be lower.
The most common solution to deal with an unbalance dataset in classical machine learning is to resample the dataset, in order to have a balanced representation of all classes. In the case of Image Enhancement, however, we usually do not have classes that can be used to quantify how unbalanced the dataset is and to resample the training dataset.
In this work, we try to solve this problem by using the in-house unsupervised clustering algorithm, *CARDIE*, which creates classes of images based on their luminance and hue distribution. Moreover, we perform an analysis on the *relevance* of the clusters in relation to the Image Enhancement algorithm we are using.
Based on these classes, we experiment with different techniques of resampling and data augmentation and show how the performance changes depending on the composition of the training dataset.
In our experiments, we focus on two different Image Enhancement tasks, *Tone Mapping* and *Denoising*, first studying the literature and then taking one state-of-the-art algorithm for each (respectively, *HDRNet* and *NAFNet*) which we use as baseline for the resampling tests.
For what concerns the *Denoising* task, we propose an alternative to *CARDIE* based on the quantification of the texture in the images, and study how the performance changes with respect to the clusters based on this feature.
We show improvements on the average performance on both tasks, which we can confirm both with the PSNR metric and visually, by looking at some sample images.

Résumé

Dans les algorithmes d'apprentissage automatique classiques, par exemple ceux utilisés pour les tâches de classification, un problème courant survient lorsqu'il s'agit d'un jeu de données d'entraînement déséquilibré. Lorsque certaines classes sont sous-représentées, cela peut entraîner une mauvaise performance. Ce problème se produit également dans les algorithmes d'Image Enhancement, où certains types d'images peuvent être sous-représentés dans le jeu de données d'entraînement. Par conséquent, les performances sur ces types d'images ont tendance à être inférieures.
La solution la plus courante pour traiter un jeu de données déséquilibré en apprentissage automatique classique est de rééchantillonner le jeu de données, afin d'avoir une représentation équilibrée de toutes les classes. Cependant, dans le cas de l'Image Enhancement, nous n'avons généralement pas de classes qui peuvent être utilisées pour quantifier à quel point le jeu de données est déséquilibré et savoir si on doit rééchantillonner le jeu de données d'entraînement.
Dans ce travail, nous essayons de résoudre ce problème en utilisant l'algorithme de clustering non supervisé en interne, *CARDIE*, qui crée des classes d'images en fonction de leur distribution de luminance et de teinte. De plus, nous effectuons une analyse sur la *pertinence* des clusters par rapport à l'algorithme d'Image Enhancement que nous utilisons.
En nous basant sur ces classes, nous expérimentons différentes techniques de rééchantillonnage et d'augmentation de données et montrons comment les performances changent en fonction de la composition du jeu de données d'entraînement.
Dans nos expériences, nous nous concentrons sur deux tâches différentes de l'Image Enhancement, *Tone Mapping* et *Denoising*, en étudiant d'abord la littérature puis en prenant un algorithme state-of-the-art pour chacune tâche (respectivement, *HDRNet* et *NAFNet*) que nous utilisons comme référence pour les tests de rééchantillonnage.
Pour ce qui concerne la tâche de *Denoising*, nous proposons une alternative à *CARDIE* basée sur la quantification de la texture dans les images, et étudions comment les performances changent par rapport aux clusters basés sur cette caractéristique.
Nous montrerons des améliorations des performances pour *HDRNet* et *NAFNet*. Ce performances sont confirmés visuellement, en regardant un échantillon d'images et avec la métrique PSNR.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we introduce the company and the specific research center where the thesis was conducted, outline the project plan and present the general organization of the thesis.

1.1 Huawei

Huawei Technologies Co., Ltd. is a leading global provider of information and communications technology (ICT) infrastructure and smart devices. Founded in 1987 by Ren Zhengfei, a former officer in the People's Liberation Army (PLA), Huawei has grown to become a multinational conglomerate headquartered in Shenzhen, Guangdong, China.
Huawei designs, develops, manufactures, and sells a wide range of products, including mobile and fixed broadband networks, consumer electronics, smart devices, distributed operating systems, and cloud services. The company is committed to bringing digital technology to every person, home, and organization for a fully connected, intelligent world.
With over 207,000 employees and operations in more than 170 countries and regions, Huawei serves more than three billion people worldwide. The company is known for its significant investments in research and development, with a focus on innovation and quality. Huawei continues to collaborate with industry organizations and ecosystem partners to drive technological advancements and promote sustainable development.
Huawei operates 14 R&D institutes and centers globally, with a presence in countries like China, Germany, Canada, and Japan. These centers focus on a wide range of advanced technologies, including 5G networks, artificial intelligence (AI), cloud computing, and smart devices. Huawei reinvests over 10% of its annual sales revenue into R&D, demonstrating its commitment to innovation. Notable research achievements include pioneering work in 5G technology, contributing significantly to global standards, and developing cutting-edge AI algorithms for various applications.
Huawei collaborates with a wide range of partners, including leading universities, research institutions, and industry organizations. These collaborations foster innovation and drive progress in ICT. Huawei's research partnerships include joint projects, funding support, and shared resources to advance technology and create impactful solutions.

In Fig. 1.1, we can observe the position of all the most important locations of Huawei centers, showcasing the company's extensive global footprint and commitment to innovation.

Figure 1.1: Map of Huawei globalized resource deployment

1.2 The Nice Research Center

write something about the nice research center The Nice Research Center (NRC), located in Mougins near Sophia-Antipolis, France, is a crucial part of Huawei's global research and development network. Established in 2013 under Mr. Stephen Busch's leadership in 2013, this center focuses primarily on image processing and enhancement technologies. The NRC has significantly contributed to Huawei's advancements in smart device photography, particularly in the development of in-house image processors and sophisticated image signal processing (ISP) algorithms.
The NRC is strategically situated in the renowned technological hub of Sophia-Antipolis, which allows it to collaborate closely with local academic institutions, research organizations, and industry partners. This collaboration fosters a rich environment for innovation and knowledge exchange, driving forward the frontiers of image processing technology.
Key areas of research at the Nice Research Center include:

- Graphic Chip Design: Developing high-performance chips that enhance the image processing capabilities of Huawei's smart devices.
- Image Signal Processing: Creating advanced algorithms that improve image quality, reduce noise, and enhance colors in photos and videos.
- Artificial Intelligence in Imaging: Leveraging AI to automate and optimize various aspects of image enhancement, making it more intuitive and user-friendly.

Through its cutting-edge research and technological breakthroughs, the NRC plays a pivotal role in maintaining Huawei's competitive edge in the global market for smart device imaging. The work done at the NRC not only improves the functionality of Huawei's products but also sets new standards for the industry. Overall, the Nice Research Center embodies Huawei's commitment to innovation, quality, and technological excellence, contributing significantly to the company's mission of creating a fully connected, intelligent world.

1.3 Project Plan and Thesis Organization

We organize the discussion on this work as follows. In Chapter 2, we first introduce some definitions and general notions on Image Processing, along with metrics to evaluate the performance of Image Enhancement algorithms. Subsequently, in Chapter 3, we describe the in-house clustering algorithm (*CARDIE*), along with other clustering techniques and a new method to measure if clusters are relevant to a task. Additionally, we present an analysis of the *MIT5K* dataset, which we will use throughout the study.

Following that, in Chapter 4, we explore two different Image Enhancement tasks, *Tone Mapping* and *Denoising*, and introduce some algorithms for these tasks, both classical and deep learning-based. Moreover, we showcase some visual results from the outputs of these algorithms.

Now that we have both baseline algorithms and a way to cluster the images, in Chapter 5, we study different resampling techniques and introduce new methods for data augmentation specifically for the *Tone Mapping* task.

Finally, in Chapters 6 and 7, we discuss the results of the resampling techniques on the two tasks, respectively. We conclude with some final considerations and ideas for future work in the Conclusion.

Chapter 2

General Notions on Images and Image Processing

In this chapter, we will first introduce some general concepts and definitions in Image Processing, and discuss some metrics that are useful to measure the performance of Image Enhancement algorithms.

2.1 Definitions

In this section, we will define important concepts in the context of image processing.

Color spaces and Color models Color models are mathematical organizations of colors. Color spaces, instead, describe how we can map a color (so a continuous quantity) to a discrete number in the color model. For example, RGB is a color model, as it organizes the colors into three channels, while sRGB is a color space in the RGB model.
Some of the color spaces/models are:

- CieL*a*b*: this color space was proposed in order to have *perceptual linearity*. A color space is perceptually linear if a change in the color space corresponds to a proportional change in the visual aspect (two colors that are visually close should also be near each other in the color space, and vice versa). It's based on the opponent model of vision, where we have red-green and blue-yellow opponents. It's defined by three axes: L, the luminosity which ranges from 0 to 100; a* which measures the red-green opponents; b* which measures the blue-yellow opponents. Using this color space, we can define any color that can be perceived by the human visual system.

- HSV: it is a cylindrical model, designed for artists because it resembles how the human eye defines colors. We can visualize this color space as a cylinder, where the vertical axis describes the value V (brightness), the angle around the vertical axis describes the hue H and the distance from the axis describes the saturation S.

- RGB, the most common color model. It contains different color spaces, some of which are: **sRGB**, i.e. standard RGB which is used in most displays and has a lower gamut; **ProPhoto**, the RGB color space with the widest gamut, encompassing about 90 % of the range of CIELab and of the HVS (Human Visual System, i.e. the range of colors visible for humans).

Color cast A color cast is a usually unwanted phenomenon that manifests itself in photographies by a uniform tint of some color on the whole image or on parts of it.

Dynamic Range In general terms, a dynamic range is the ratio between the maximum value a quantity can assume and the minimum value. In the context of photography and images, the Dynamic Range of a scene describes the range of light intensity values that the scene can assume. It can be calculated as the ratio between the maximum measurable brightness to minimum measurable darkness in an image.
The dynamic range of reality can be seen as infinite, going from total darkness to the intensity of light of the sun. The human eye can perceive a wide range of luminances, on the other hand the typical devices we use to visualize images are limited to 8-bits per channel, so they can display a range from 0 to 256 different luminances. Therefore we will have to approximate the dynamic range of reality in some way, in order to be able to represent it on a display.

Exposure time It's the length of time that a photocamera's sensor is exposed to light. The higher the exposure time, the larger the amount of light that is captured by the sensor. Therefore, as we can observe in Fig 2.1, a photograph taken with a lower exposure time will be darker than one taken with a larger exposure time.

An *over-exposed* image appears too bright and doesn't retain details in its brightest

Figure 2.1: Results of taking photographs of the same scene with different exposure times

parts; an *under-exposed* image, instead, appears too bright and doesn't retain details in its darkest parts.

High Dynamic Range and Low Dynamic Range As we can see in Fig 2.1, by looking at images taken with larger exposure time, we can better observe the details in the darker areas; on the other hand, in images taken with lower exposure time, we can better observe details in the brighter parts of the image.
HDR (High Dynamic Range) imaging is used to create images that are more adherent to what the Human Vision System (HVS) would perceive, retaining the details of both the darker and brighter areas of the scene.

Instead, the pictures we typically can see on the displays of our devices are LDR (Low Dynamic Range) images, i.e. compressed images with pixels whose values range from 0 to 255.
HDR images can be created in different ways: by merging different LDR images taken with different exposure times (which we can see in the example in Fig 2.2), which estimates the true luminance signal; or by recording the HDR directly, using specialist scientific digital cameras.

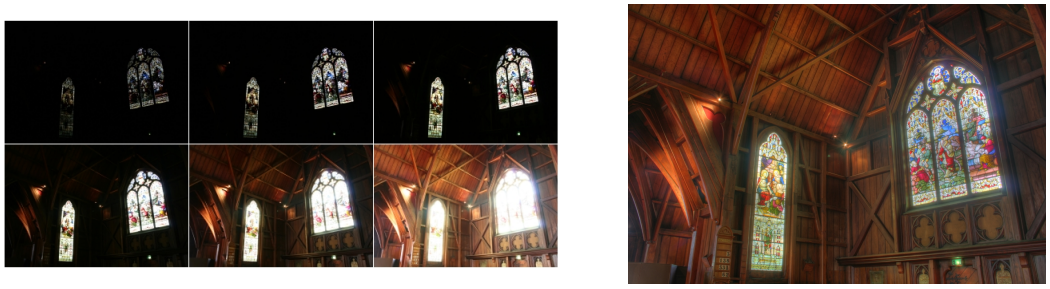


Figure 2.2: Example of creation of HDR image, on the left the images at different exposure times, on the right final HDR image

GLCM Texture Features In order to extract features that describe the texture of the images, we can use the Gray level Co-occurrence Matrix Haralick et al. [1973], a classical method to define texture in images which was proposed in the 1970s to analyze satellite pictures.
GLCM creates a matrix for each of the 4 directions (0°, 45°, 90°, 135°), with dimensions NxN, where N is the number of gray-levels in an image (e.g. an image in 8 bits will have N=255). We have a matrix for each direction; each matrix, for each position (i, j), contains the number of times pixels with gray-level (i, j) are near to each other on the axis of the corresponding direction. For example, for the 0° matrix, the element at position (2,1) will contain the number of times the pixels with grayvalues 2 and 1 are adjacent horizontally.

Using these matrices, we can extract different features and we will focus on three of them:

- contrast: $\sum_i \sum_j P(i,j)(i-j)^2$ It is also called *sum of squared variances*, as it assigns a weight equal to the square of the gray-level difference. Therefore equal levels next to each other will have a 0 weight, and the weight will increase as we go further from the diagonal (i.e. when gray-levels with higher difference are close to each other)

- dissimilarity: $\sum_i \sum_j P(i,j)|i-j|$ Similarly to *Contrast*, it assigns more weight to elements far from the diagonal and 0 to the elements on the diagonal

- homogeneity: $\sum_i \sum_j \frac{P(i,j)}{1+(i-j)^2}$ It is effectivily the inverse of the *Contrast* feature, as it assigns higher weights to elements near the diagonal, thus computing how smooth and uniform the image is

where $P(i, j)$ is the value of the GLCM matrix at position (i, j).

Pyramid In the context of image processing, a pyramid is a way to show an image with a multiresolution representation, by recursively applying smoothing and subsampling. The two most used ways to generate pyramids are the following:

- Gaussian Pyramid: at each level, the image is smoothed using Gaussian blurring and downsampled (usually by a factor 2) at each level, recursively. This method is used for image compression, image blending and object detection (we can detect the same object at multiple scales)
- Laplacian Pyramid: in addition to blurring and downsampling, at each level, we save the difference between the image and its blurred version. In this way, we can extract the edges of the image at different resolutions. Laplacian pyramids are used for image compression and noise reduction

2.2 Metrics to measure the performance of Image Enhancement algorithms

There are different metrics we can use to measure the performance of a Image Enhancement algorithm. In the following subsections, we describe the most useful ones for our application.

2.2.1 PSNR

One of the most popular metrics to compare images is the Peak Signal to Noise Ratio (PSNR). We measure the PSNR between the expected image and the image produced by the algorithm. In the case of our dataset, we have ground truths images which were produced by image processing experts and an output image produced by our tone mapping algorithm.
The PSNR is calculated as:

$$PSNR = 20log_{10}\left(\frac{L-1}{RMSE}\right)$$

where the RMSE is the root mean squared error and L is the number of maximum possible intensity levels. The RMSE is calculated between the pixels of the ground truth image and the predicted image.

2.2.2 SSIM

Another possible metric is Structural Similarity Index Measure (SSIM) Wang et al. [2004]. While the PSNR takes into consideration only the comparison between one pixel at a time, SSIM works with windows, so it considers neighbourhood of pixels which are close to each other and therefore inter-dependent.

The purpose of SSIM is to measure how structurally close the HDR and LDR images are, in terms of structural information. The comparison between the two images is done in considering the following features: luminance, contrast, structure.
The formula of the SSIM metric between two windows (x and y) is :

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

where μ_a is the mean of a window a, σ_a^2 is the variance of a window a, c_1 and c_2 are two constants that are used to stabilize the division when the denominator is close to zero.

2.2.3 TMQI

Tone Mapped Quality Image index (TMQI) Yeganeh and Wang [2012] strives to create an objective metric specifically for tone-mapping algorithms. It combines two main quantities: structural similarity and naturalness.
The structural similarity term is a modified version of the SSIM, taking into account two main relevant facts: first, we should not penalize differences of signal when both the signals in the HDR and LDR images are significant or insignificant: second, we should instead penalize the situations where the signal is significant in one image but insignificant in the other. This *significance* is defined based on studies on the Human Visual System Barten [1999], which show that there is not a specific threshold of values of contrast that is significant, but nonetheless we can define a probability function of observing contrast variations.
The second term, statistic naturalness, tries to define how natural an image looks to the human eye. The authors model the concept of naturalness by taking into consideration brightness and contrast. They compute the histograms of the mean and standard deviation of 3000 images that are considered natural then they approximate the distributions of the means and standard deviations with a Gaussian and a Beta probability density function, respectively. Mean and standard deviation are useful in order to measure the average intensity (mean) and contrast (standard deviation). Moreover, the authors consider brightness and contrast to be independent, therefore they compute their joint pdf as the product between the two. The final formula to calculate the statistical naturalness is:

$$N = \frac{1}{K} P_m P_d$$

where P_m is the distribution of the means, P_d is the distribution of the standard deviations and K is a normalization constant, useful to have a naturalness value ranging from 0 to 1.
The final formula of TMQI is $TMQI = aS^\alpha + (1-a)N^\beta$ where S is the structural similarity component, a is a constant to measure the importance of the two components and α and β determine their sensitivities.

2.2.4 Color Video VDP

Color Video Visual Difference Predictor (Color Video VDP) Mantiuk et al. [2024] is another metric that can be used to measure the difference between two videos or two images. It is composed of the following steps: both the test and ground truth images are mapped to the XYZ color space, then the linear colors are transformed in the DKL opponent color space, defining an achromatic and two chromatic channels. The achromatic channel is decomposed into two temporal channels: transient and sustained.
Each of the resulting 4 channels is decomposed using a Laplacian pyramid into spatial bands. Then the bands from both images are passed through models of contrast sensitivity and masking models.
The contrast sensitivity model parametrizes the ability of the HVS of seeing patterns of different spatial and temporal frequency, size and luminance levels. The contrast masking model, instead, converts the local differences in the images to perceived differences, accounting for the fact that it is more difficult for humans to notice differences in areas where we have a high frequency (e.g. very textured areas).
We also have a cross-channel component, which takes into consideration the fact that, if we have a very strong contrast in one of the channels, we are less likely to notice the contrast in the other channels.
Finally the channels are pooled and the results are regressed to a score.
The score is expressed in Just Objectionable Difference (JOD). If the value is 10, it means that the reference and test images are the same: moreover, a difference of 1 JOD means that 75 % of the population will choose the image with the highest scores.

Chapter 3

Unsupervised Clustering of Images

In this chapter, we will discuss the novel unsupervised clustering algorithm for Image Enhancement, *CARDIE* and a method to quantify how relevant clusters are according to IE algorithms. Then, we will show the results of *CARDIE* on a famous image dataset. Finally, we will compare our clustering algorithm to other approaches.

3.1 Introduction

In machine learning and deep learning algorithms, the composition of the training dataset is fundamental. For example, in the context of classification tasks, if we have an unbalanced dataset, where some classes are under-represented, the performance of the algorithm could be severely impacted.
A similar problem arises in Image Enhancement algorithms that employ deep learning (e.g. Tone Mapping or Denoising algorithms, which will be discussed in the next chapter). We can observe that performances quickly degrade when we have images whose features differ significantly from the average of the distribution in the training dataset. As the most frequent solution to this unbalance in the training dataset for classification is to resample the dataset to reach a more balanced distribution, we want to apply a similar solution to our Image Enhancement task.
In our case, it is difficult to measure how unbalanced the features in the training dataset are. In fact, we do not have classes. We will use the in-house unsupervised clustering algorithm to create classes that are *relevant* to the IE algorithm. This algorithm is called **CARDIE**, Clustering Algorithm on Relevant Descriptors for Image Enhancement.

In general, the organization of this work can be summarized by the following points:

- from each image, extract features that are significant to the image enhancement task
- use these features to cluster the training images in an unsupervised fashion, obtaining classes

- measure how the composition of the dataset in terms of these classes impacts the performance of the algorithms (we will use two main algorithms, one for Tone Mapping and one for Denoising)
- resample the training dataset in order to have a more balanced training dataset and observe if/how the performance changes

In the next sections, we will thoroughly describe the first two steps, in addition to a new method to test whether some image clusters are *relevant* to a task.

3.2 Clustering algorithm on relevant descriptors

In order to cluster the image we employ the in-house clustering algorithm, which is composed of two main steps:

1. extract descriptors that are meaningful for the Image Enhancement task
2. cluster the descriptors using HDBSCAN McInnes et al. [2017]

3.2.1 Extracting descriptors

The algorithm extracts two kinds of descriptors: a first value based on luminance, a set of values describing the principal hues present in the image.
First, we define an *average* luminance in an image in our dataset, which will be later used to judge how *bright* or *dark* an image is. It is defined as:

$$\bar{L}(\mathcal{D}_K, p) = \frac{1}{K} \sum_{j=1}^{K} L(x_j, p). \tag{3.1}$$

where K is the number of images in the subset we consider to calculate the average (it is a free parameter of the clustering algorithm), x_j is the $j-th$ image in the subset, $L(x_j, p)$ is the luminosity of the pixel p in the image x_j and $\mathcal{D}_K$ is the subset of our dataset.

Given this quantity, we compute two thresholds: $\bar{L}_{\text{low}}^{\mathcal{D}_K}$ and $\bar{L}_{\text{high}}^{\mathcal{D}_K}$, as the 20th and 80th percentile of $\bar{L}(\mathcal{D}_K, p)$. For each image x_j in our dataset, first we compute $\bar{L}_j$, i.e. the median value (50-th percentile) of its luminance; then we will assign the luminance descriptor following these rules:

- label the image x_j as *Low* if $\bar{L}_j < \bar{L}_{\text{low}}^{\mathcal{D}_K}$
- label the image x_j as *High* if $\bar{L}_j > \bar{L}_{\text{high}}^{\mathcal{D}_K}$
- label the image x_j as *Average* otherwise

Second, we compute some descriptors based on hue. For each image x_j, we compute its hue distribution $\mathcal{H}(x_j, \theta)$ and assign $\mathcal{C}_m$ as dominant color if:

$$\int_{\mathcal{C}_m - \Delta\theta/2}^{\mathcal{C}_m + \Delta\theta/2} \mathcal{H}(x_j, \theta) d\theta \geq \frac{1}{l} \tag{3.2}$$

i.e. if more than $1/l$ of its hue distribution is contained in a $\Delta\theta$ interval around the $\mathcal{C}_m$.

In the end of the feature extraction, we will have, for each image, a row with 5 color descriptors and 3 luminance descriptors, all containing boolean values. We can therefore apply the unsupervised clustering algorithm to this limited number of features, obtaining a relatively low complexity of the model.

3.2.2 Unsupervised Clustering

After extracting the features, we exclude the descriptors whose variance is lower than some threshold σ_d. Then we perform unsupervised clustering using HDBSCANMcInnes et al. [2017]. At the end of this process, we will have a cluster label for each image. We perform a small gridsearch to find the best values for σ_d and m_{min}, i.e. the minimum number of samples per cluster. We measure the performance of the clustering process using the Silhouette Score.

3.3 Evaluating Image Enhancement Algorithms

We also study a method to evaluate how relevant the clusters obtained with a certain clustering method are according to a Image Enhancement algorithm. The main idea of this procedure is the following: if two images in different clusters are enhanced in *different ways*, it means that the clusters are relevant to the IE algorithm, as they are treated differently; vice versa for two images in the same cluster, they should be treated equally by the IE algorithm, thus they should be enhanced in the *same way*.
We approximate an Image Enhancement Algorithm $\mathcal{A}$ as a function of the following form:

$$L(x_j^{pp}, p) = \mathcal{A}(L(x_j^i), p)$$

where $L(x_j^i, p)$ is the luminance evaluated on the j-th input image on the pixel at position p, x_j^{pp} is the post-processed image. We assume a form of $\mathcal{A}$ such that:

$$L(x_j^{pp}, p) = \frac{L(x_j^i, p)^\gamma}{L(x_j^i, p)^\gamma + \mu^\gamma}$$

We randomly subsample P=100 pixels from each image and fit the equation to each image, obtaining a distribution of fit parameters $(\gamma_j^{\mathcal{A}}, \mu_j^{\mathcal{A}})$

We want to compare clusters generated with our algorithm to see if samples in the same cluster have been enhanced similarly. So we compute for each cluster, the following distributions:

$$\Gamma_l^{\mathcal{A}} = \left\{\gamma_1^{\mathcal{A}}, ..., \gamma_{n_l}^{\mathcal{A}}\right\} \qquad M_l^{\mathcal{A}} = \left\{\mu_1^{\mathcal{A}}, ..., \mu_{n_l}^{\mathcal{A}}\right\}$$

where n_l is the number of samples in a certain cluster.

We want to test for which combination of l and a (clusters) the corresponding distributions $(\Gamma_l^{\mathcal{A}}, M_l^{\mathcal{A}})$, $(\Gamma_a^{\mathcal{A}}, M_a^{\mathcal{A}})$ are statistically different.

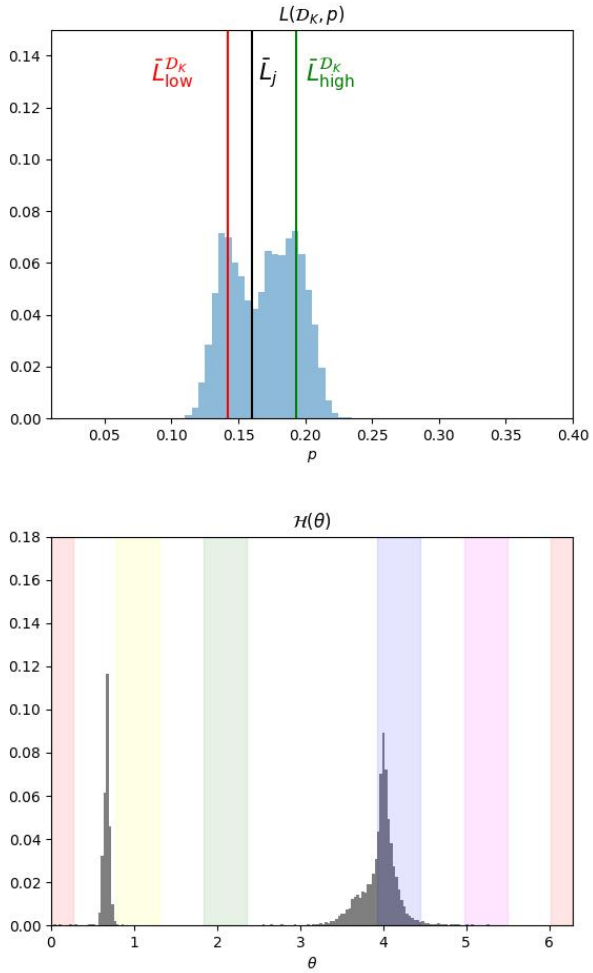


Figure 3.1: Examples of $L(\mathcal{D}_K, p)$ and $\mathcal{H}(\theta)$ for image labelled as Average Blue by the clustering algorithm

The null hypothesis is that, for example, taking clusters c1 and c2, the set of parameters (for the tone mapping operator A) is drawn for the same distribution. We perform a two-sided Kolmogorov–Smirnov (KS) test with pvalue=0.05. If the test is failed, it means that the IE algorithm has treated the two clusters differently, and vice versa if the test succeeds. For all possible combinations of clusters, we perform two KS tests, one for each set of parameters.

The final result of the two tests on the two sets of parameters can be summarized with a quantity $\mathcal{I}^{\mathcal{A}}_{i,a}$ which is equal to 1 when at least one of the two KS tests fails (which means that at least one of the set of parameters has a different distribution, so the algorithm has a different behaviour for the two clusters), otherwise it is 0. We can write it more

formally as:

$$\mathcal{I}_{l,a}^{\mathcal{A}} = \begin{cases} 1 & \text{if KS}\left(\Gamma_a^{\mathcal{A}}, \Gamma_l^{\mathcal{A}}\right) < 0.05 \text{ or KS}\left(M_a^{\mathcal{A}}, M_l^{\mathcal{A}}\right) < 0.05, \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

In the plots displayed in the following section, we represent the results of the KS tests with a block for each combination of two clusters. With the color green, we represent blocks where $\mathcal{I}_{l,a}^{\mathcal{A}}$ is equal to 1 (so the clusters are treated differently), vice versa $\mathcal{I}_{l,a}^{\mathcal{A}}$ is equal to 0 when the block is red and the clusters are treated in the same way by the IE algorithm.

3.4 Results of Clustering algorithm on MIT5K

In this section, we will apply the clustering algorithm to an image dataset and show the results, to better illustrate the method with some examples.

3.4.1 The MIT5K dataset

We perform our tests on the MIT5K dataset Bychkovsky et al. [2011], which is composed of 5000 images, in RAW format.
The images have been taken by different photographers, and they show a variety of different scene and subjects. Each image has been retouched by 5 photography students using Adobe Lightroom with the purpose of producing visually pleasing images. Therefore, in the final dataset, we have 5000 input images and 5 retouched versions which we select as ground truths.
Given that this manipulation of pictures is a very subjective task and that the 5 retouchers all have different styles, the retouched version of the same input image can be very different one from the other.
In our experiments, we used the retouched versions of ExpertC, the third student retoucher, as ground truth.

3.4.2 Extracting descriptors and creating clusters from MIT5K

In our case, we calculate the *average luminance image* using K=100 pictures and we set the possible colors as $\mathcal{C} = \left(0, \frac{\pi}{3}, \frac{2\pi}{3}, \frac{4\pi}{3}, \frac{5\pi}{3}\right)$, i.e. *(red, yellow, green, blue, magenta)*, lastly, we set $\Delta\theta = \frac{\pi}{6}$.

In Fig. 3.1, we show an example of luminance and hue distribution for a random image in our dataset.

Using these descriptors, we cluster the images and we obtain **12** clusters, with no noise points. We show some statistics related to these clusters in Tab 3.1. We also show some examples of images and how they get clustered in Fig 3.2. From these images, we can confirm visually that the clustering algorithm correctly classifies the images based on their luminance and color descriptors.
We perform some data analysis on the clusters that the algorithm produced. In Fig 3.3, we show the composition in terms of luminance and color.

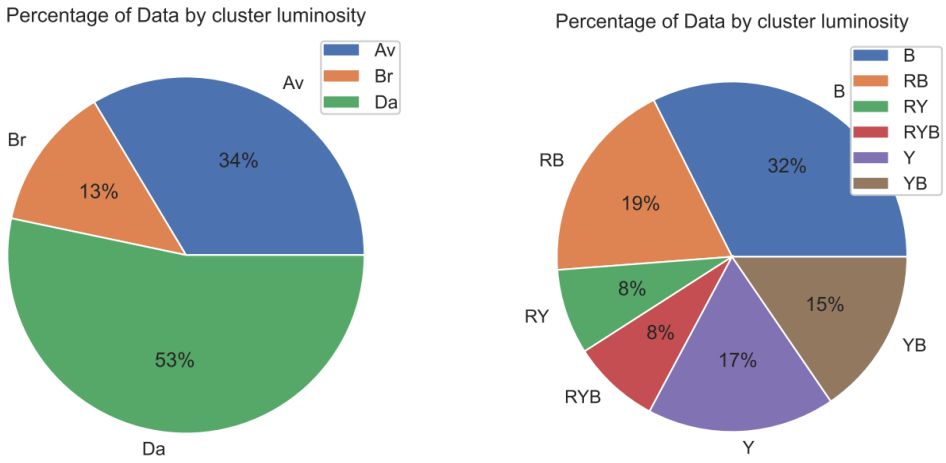
Table 3.1: Clusters statistics

c_l	labels	% c_l
DaRB	{dark lum, red, blue}	10.27
DaRYB	{dark lum, red, yellow, blue}	8.32
DaB	{dark lum, blue}	19.6
DaYB	{dark lum, yellow, blue}	15.63
AvB	{average lum, blue}	9.95
AvRB	{average lum, red, blue}	5.57
AvRY	{average lum, red, yellow}	5.49
AvY	{average lum, yellow}	12.09
BrB	{bright lum, blue}	2.88
BrRB	{bright lum, red, blue}	2.61
BrRY	{bright lum, red, yellow}	2.48
BrY	{bright lum, yellow}	5.15

(a) BrightBlue (b) DarkBlue (c) AvgRedBlue

Figure 3.2: Examples of images from MIT5K dataset and their respective clusters

We can observe that more than half the images in our dataset are classified as dark, while we have about a third of average luminosity images and few that are bright. For what concerns the hue, blue images are the most represented, while the least represented combinations are RY and RYB.



(a) Distribution of the luminance in the clusters (b) Distribution of the colors in the clusters

Figure 3.3: Distribution of levels of luminance and of hues in the clusters

3.4.3 Results on the Relevance tests using CARDIE

In this section, we show the results of the KS tests, with the method described above, using HDRNet as a IE algorithm.
We note that we performed some trials to see whether this method was sensitive to the *p_value* parameter and saw that the results were stable. Nevertheless, we show the results employing $p\_value = 0.05$. In Fig. 3.4, we can observe all the results for the KS tests and we can see that there are few combinations of clusters that are treated in the same way by HDRNet, therefore, we can conclude that the clusters obtained using CARDIE are relevant to our task. We can also note that the few pairs that are treated in the same way belong to clusters with similar composition (e.g. *(AvRB and AvB)* or *(BrRB and BrRY)*).

3.5 Other ways to create the clusters

In order to justify the use of CARDIE, we compare the previously discussed results with other clustering algorithms.
We compare them with three different ways to cluster the image:

- first, we tried to employ the descriptors in the MIT5K dataset. They consist of human-annotated semantic information about the location, environment and subject of the photographs.

- then we employ the ResNet50 model trained on the Places365 dataset Zhou et al. [2017] to extract other semantic information about the images, the kinds of labels given to the images are very similar to the ones described before

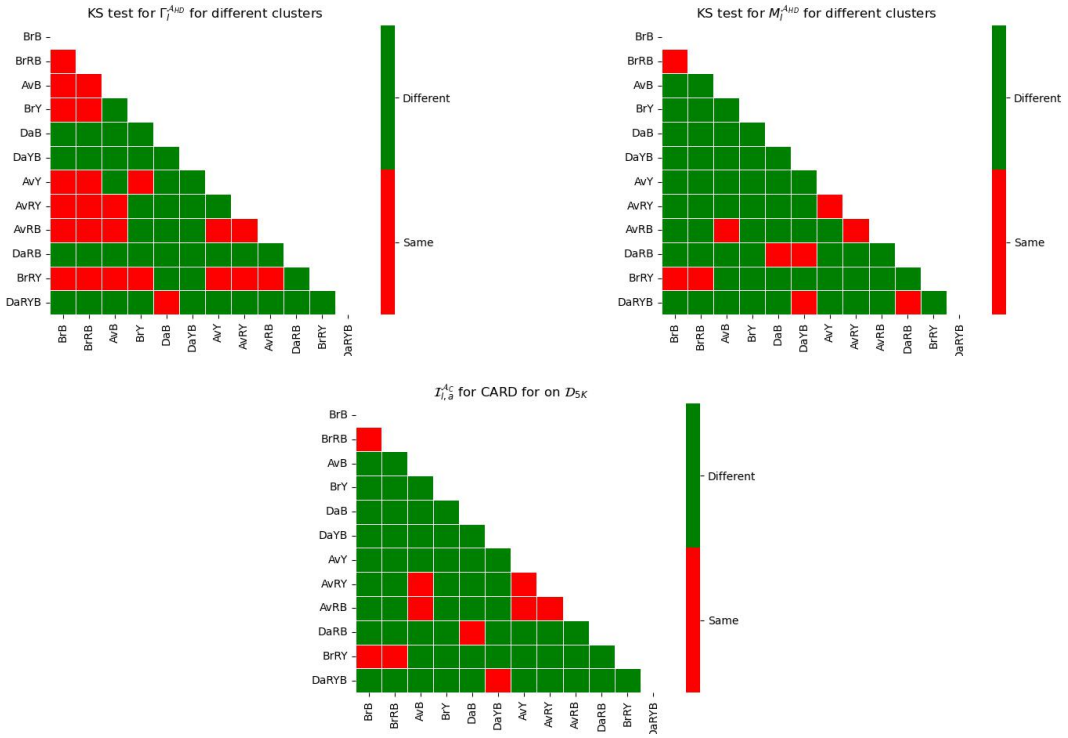


Figure 3.4: Results for the test of relevance on HDRNet and the CARDIE clustering algorithm: KS test for the Γ parameter distribution (left top), KS test for the M parameter distribution (left right) and cluster difference indicator $\mathcal{I}_{l,a}^{A}$ (bottom)

- lastly, we use the model mentioned in the paragraph 4.1.4, NamedCurves, to divide the images into different clusters based on hues

3.5.1 Clustering with NamedCurves

We will describe the Tone Mapping algorithm NamedCurves 4.1.4 more thoroughly in 4.1.4; for now, we simply use its method to extract color probability maps, which we will use to classify the images according to their most represented hues.
NamedColor uses the Van De Weijer Color Naming module to extract 6 different probability maps for each image. We try to extract the most represented colors in the image, starting from the probability maps.
First, for each pixel, we get the color with the highest probability. Then, we take the first three colors with the highest number of represented colors, if the number of pixels of that color is at least 5 % of the total number of pixels. We discard the Achromatic color, except in the case where it is the only color that satisfies the 5 % rule.

We also discard the color classes that have less than 100 images assigned to them, classifying them as *Outliers*.

In Fig 3.5, we can observe the resulting division of the images into the color clusters.

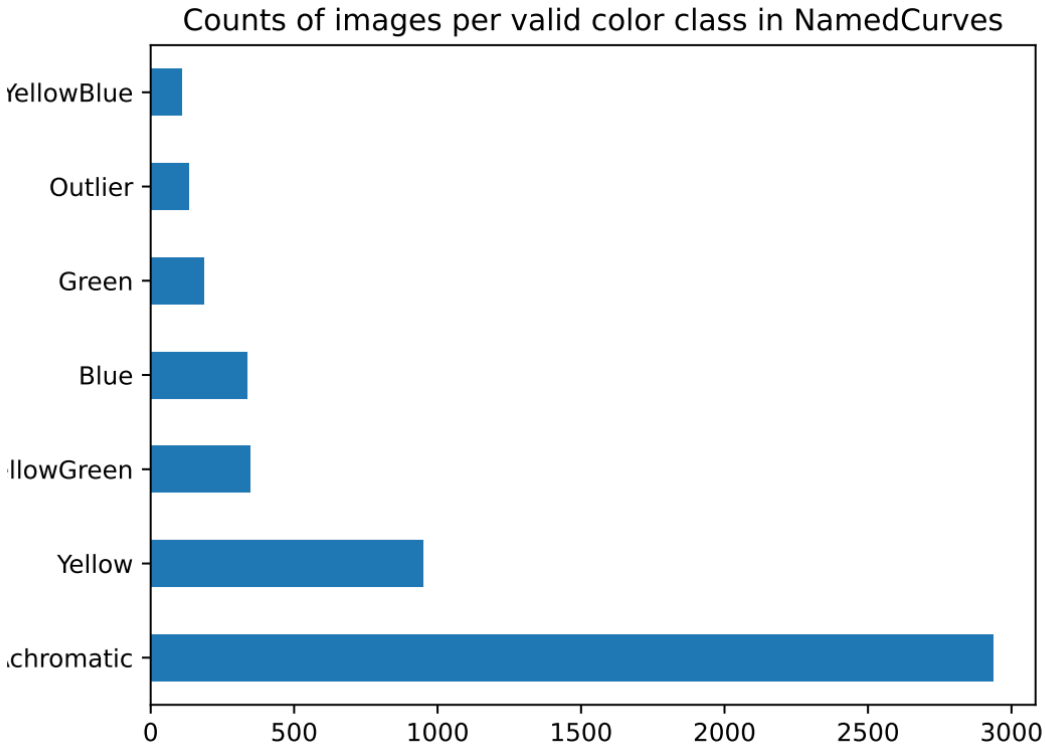


Figure 3.5: Number of images for each cluster extracted from the Color Naming module of Named Curves

We can notice that more than half the images are classified as purely Achromatic (so they have less than 5 % of chromatic color classes).
As more than half the images are classified as *Achromatic*, we disregard this clustering technique.

3.5.2 Clustering with ResNet Places365

We also try to create clusters using *semantic* information about the images. Therefore, we employ ResNet trained on the dataset Places365, in order to extract some descriptors for each image.
This model outputs many different descriptors which describe interesting proprieties of the images (e.g. indoor/outdoor) but there is no automatic process to select which are the most relevant. We will focus here on only 4 main attributes, namely *natural light (Nat), no horizon (NH), man-made (Man), open area (Op)*, which we selected among the almost

Table 3.2: ResNet Places' clusters statistics

c_l	labels	% c_l
NatOp	{natural light, open area}	8.86
NatNHOp	{natural light, no horizon, open area}	12.17
NatManOp	{natural light, man made, open area}	12.60
NHMan	{no horizon, man made}	29.76
NatNHManOp	{natural light, no horizon, man made, open area}	36.61

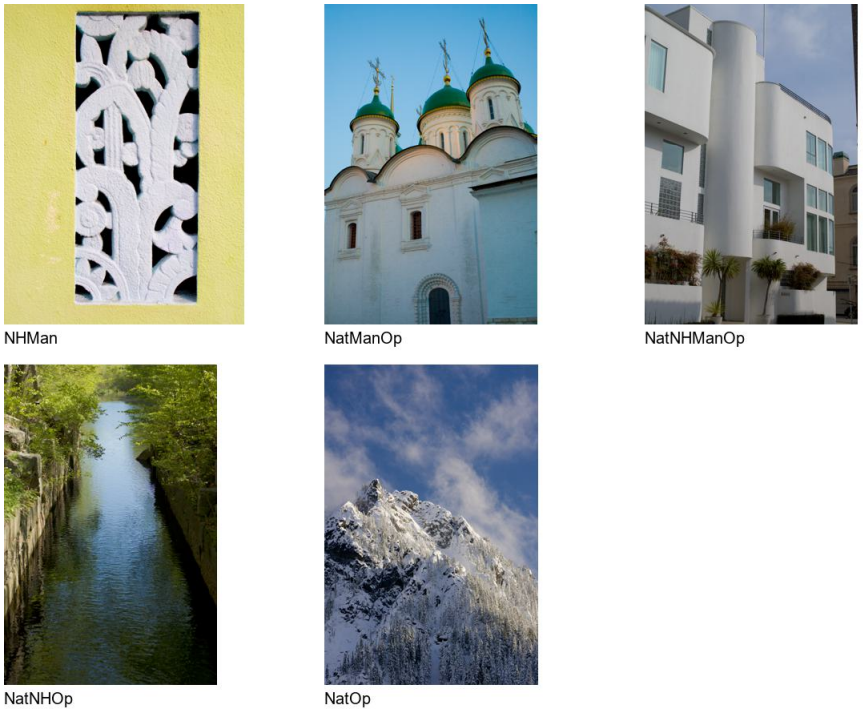


Figure 3.6: Sample of images clustered using semantic information obtained by *ResNet50-places365* (ground thruth images are displayed for clarity)

200 original descriptors. We *think* these features are the most relevant for our task, as they are the most frequent in the dataset. Using these descriptors, we cluster the images using HDBSCAN as with CARDIE. We obtain 5 clusters, and we show their statistics in Tab 3.2. Moreover, in Fig: 3.6 we show some samples of images and their clusters obtained with *ResNet-Places365*. We can visually confirm that these clusters correctly identify the semantic informations present in the pictures.

3.5.3 Clustering with descriptors of MIT5K

Other than the images, the MIT5K dataset Bychkovsky et al. [2011] contains also semantic descriptors for each image. These descriptors were manually annotated, so they present high-quality data that we can use to cluster the images.
For each image, we have one annotation for each of the following categories: *location, time of the day, light, subject*. We remove the column *light* as it is highly correlated with *location* and *time of day*. Then we convert all the attributes to columns containing only boolean values and remove all the columns with *unknown* values.
Using the resulting dataset, we proceed as before (hyper-parameter search and HDBSCAN) and create the image clusters.
We obtain **17** clusters. The clusters are not very easily interpretable, as their labels are quite complex, therefore we do not show statistics and plots on them.

3.5.4 Summary of clustering results and relevance tests

Table 3.3: Number of clusters $\# c_l$, the silhouette score SC' and percentage of noise points $\% c_s$ for the clustering algorithms studied here.

	$\# c_l$	SC'	$\% c_s$
CARDIE	12	0.67	0.00
MIT5K	17	0.90	0.52
ResNet Places365	5	0.87	1.86
NamedCurves	6	/	2.64

In Tab 3.3, we show some statistics about the clusters obtained with all the techniques mentioned above. We can observe that MIT5K has the highest Silhouette Score, while CARDIE has the lowest percentage of noise points. Moreover, the clustering based on the descriptors of MIT5K has the highest number of clusters, which can make the clusters less easily interpretable.
We will proceed to perform the Test of relevance for all the different clustering algorithms (except for the one based on *Named Curves*), which we have deemed to be not interesting for our purposes.

For each of the three clustering algorithms (*CARDIE, MIT5K, ResNet-Places*), we perform the Relevance Test described in Sec 3.3. We can see the results of the tests in the plots in Fig. 3.7.

From these plots in Fig 3.7, we can see that the clusters obtained with the semantic descriptors generated by *ResNet50* pretrained on *Places365* seem to be relevant, as most of the clusters have different distributions of parameters, so different clusters are treated differently by the Tone Mapping algorithm. There is only one combination of clusters that does not pass the KS Test, but it is composed of clusters with similar descriptors (*NoHorizon, OPen area*) and (*NoHorizon, OPen area, Man made*).

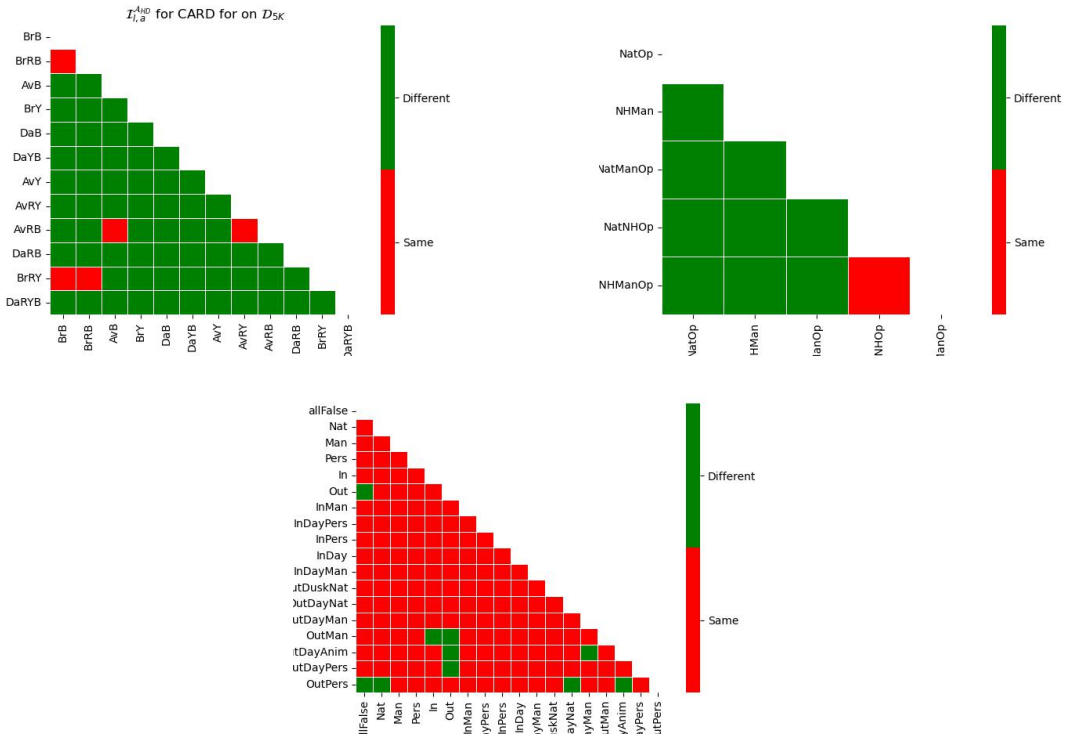


Figure 3.7: Results for the test of relevance on HDRNet and three different clustering algorithms: CARDIE(left top), ResNet-Places365 (left right) and MIT5K descriptors (bottom)

Instead, for what concerns the relevance test on the clusters obtained with the descriptors of *MIT5K*, we can see that the vast majority of the combinations of clusters are treated in the same way by the Tone Mapping algorithm. Therefore, we will disregard the latter method to create the clusters, focusing on *CARDIE* and *ResNet-Places*.

Chapter 4

Tone Mapping and Denoising

In this chapter, we will study two different Image Enhancement techniques: Tone Mapping and Denoising, which we will use as baseline algorithms for our resampling strategies based on CARDIE and other clustering methods.

4.1 Tone Mapping

Tone Mapping is an Image Enhancement operation that is used to map HDR images into LDR images.
Tone Mapping Operators (TMOs) can have three main different purposes [Reinhard and Devlin, 2005]:

- *visual system simulators*: they try to simulate both the properties and the limitation of the HVS. For example, they may try to limit the contrast in night scenes
- *best subjective quality*: they try to produce images as similar as possible to an ideal of subjective quality: in this context, TMOs are often used for artistic purposes
- *scene reproduction*: they try to represent the image as close as possible to the original scene on a display

In this work, we will focus on this last objective for Tone Mapping.
In order to show images on most devices, we need the range of pixels to be between 0 and 255, instead, HDR images have a wider range. Thanks to this wider range, HDR images have more details and are more representative of reality.
The purpose of Tone Mapping Operators, is to find a *tone curve* that maps the HDR space into the LDR space, such that the resulting lower range image maintains the characteristics of the HDR image (e.g. brightness, contrast, appearance).

Tone mapping operators can be classified into two main families: global and local operators. Global operators compress the range by using characteristics computed on the whole image, so they perform the same operation on all the pixels; instead, local operators work on a neighbourhood of pixels. Local operators can yield a better performance, since

their effect is similar to the one of the HVS, but they can result in artifacts around the edges of the image and they are more computationally expensive than global operators.

In Salih et al. [2012] the authors divide the Tone Mapping Operators in 4 groups:

- global
- local
- frequency: they compress the image in the frequency domain
- gradient: they compress the gradient image

4.1.1 Typical Tone Mapping pipeline

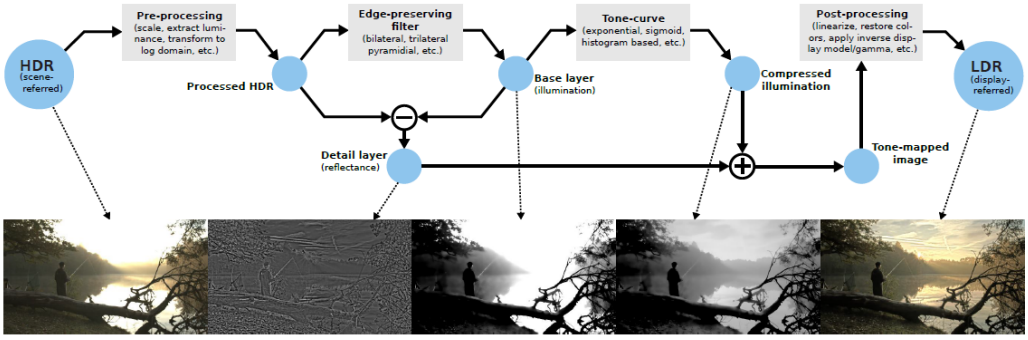


Figure 4.1: Typical pipeline of a tone mapping

In [Salih et al., 2012], the authors describe a typical tone mapping pipeline, which is represented in Fig 4.1 and is composed of the following steps:

1. The HDR image is pre-processed, using different techniques, yielding a **Processed HDR Image**
2. The Processed HDR Image undergoes an Edge-preserving smoothing filter (e.g. the median or bilateral filter), yielding a **Base Layer** Image
3. The Base Layer Image is subtracted from the Processed HDR Image, creating a **Detail Layer Image**, which contains only the details of the original picture, which we don't want to compress
4. The Base Layer Image is compressed using a Tone Curve operator, yielding the **Compressed Illumination** image
5. Now we re-introduce the details in the compressed image, summing the Compressed Illumination Image and the Detail Layer Image, obtaining the **Tone mapped Image**
6. Finally, we apply some post-processing on the Tone-Mapped Image and get the final LDR Image

4.1.2 Reinhard Devlin

Reinhard and Devlin in 2005 Reinhard and Devlin [2005] proposed one of the most famous and used classical Tone Mapping algorithm. It is a *global* operator.

It is a classical Tone Mapping algorithm, so it does not make use of deep learning techniques. Therefore it does not require training and it is generally faster than deep-learning-based methods. It models the tone mapping curve based on how the HVS perceives illumination.
In particular, it constructs the tone curve taking inspiration from the function that produced the potential as a function of intensity I:

$$V = \frac{I}{I + \sigma(I_a)} V_{max} \qquad \sigma(I_a) = (f I_a)^m$$

where I_a is the adaptation level, which can be described as a function of the light intensity that a certain photoreceptor has been exposed to in the past, σ is the semisaturation constant, which describes the state of the long term adaptation of the photoreceptors as a function of I_a. Instead, m and f are parameters that control, respectively, the contrast and the intensity.

Based on electro-physiological studies and experiments, the authors set the initial value of m as:

$$m = 0.3 + 0.7k^{1.4}$$

The parameter k is called *key* and it depends on how bright or dark the image is. We calculate is as:

$$k = (L_{max} - L_{av})/(L_{max} - L_{min})$$

where L is the luminance, defined as L = 0.2125R + 0.7154G + 0.0721B and all the operations (average, maximum and minimum) are computed on the log scale. The value of m changes the shape of the tone curve, making it possible to decide whether to increase details in the high, low or medium-intensity regions.

Instead, the value of f is studied in the exponential domain, by defining $f = \exp(-f')$. The initial value of f' is set to 0 and can range from -8 to 8.

In general, the adaptation level I_a should be calculated independently for each color channel, as in the HVS the cones (that respond to the light emitted by different colors) act in an independent way from each other. Instead, in some cases, we can notice strong color casts on the image, therefore it is necessary to take into account the full luminance value instead of the single color channel. Thus, we can compute I_a as:

$$I_a = cI_{r|g|b} + (1 - c)L$$

By varying the value of the parameter c, we can change how independent the color channels are. If we set $c = 1$, we will have totally independent channels, if we set $c = 0$, we will only take into account the value of the luminance.

Moreover, we can decide how much the adaptation level is influenced by the single pixel intensity or by the global average of the colors.:

$$I_a = aI_{r|g|b} + (1-a)I^{av}_{r|g|b}$$

If we set the parameter $a = 1$, the adaptation will be equal to the pixel intensity, instead, if $a = 0$, it will be completely dependent on the global average.
In the end, the adaptation level will be described by the following equations, depending on the parameters a and c:

$$I^{local}_a = cI_{r|g|b} + (1-c)L \qquad I^{global}_a = cI_{r|g|b} + (1-c)L^{av} \qquad I_a = aI^{local}_a + (1-a)I^{global}_a$$

In Tab 4.2, we can see a summary of the 4 user-defined parameters, their description, initial value and operating range.

TABLE 1
User Parameters

Parameter	Description	Initial value	Operating range
m	Contrast	$0.3 + 0.7k^{1.4}$	$[0.3, 1.0)$
f'	Intensity	0.0	$[-8.0, 8.0]$
c	Chromatic adaptation	0.0	$[0.0, 1.0]$
a	Light adaptation	1.0	$[0.0, 1.0]$

Figure 4.2: Recap of user parameters in the Reinhard Devlin algorithm

4.1.3 HDRNET

HDRNET a tone-mapping algorithm that uses convolutional neural networks, was proposed by Google in 2017 Gharbi et al. [2017]. Its main purpose is to tackle image enhancement in real time on mobile devices. Given this final objective, it is necessary that the algorithm has limited needs in terms of computing power.
The authors introduce three main novelties:

1. the employment of a bilateral grid, where we have for each pixel, its location (x,y) and a third dimension that is a function of the pixel's color. Most predictions are done on this grid. Then, using a slicing operation, we reconstruct the output image at full resolution applying the bilateral grid of coefficients.

2. instead of learning directly the output, the network learns a local affine transformation from input to output. They introduce a new multiplicative node to apply this transformation to the input

3. to decrease computational costs, most of the computations are done in the low-resolution version of the image. Instead, the loss is calculated directly on the full-resolution image. In this way, we can train and optimize the network directly on the full-resolution image, while still having limited computational needs

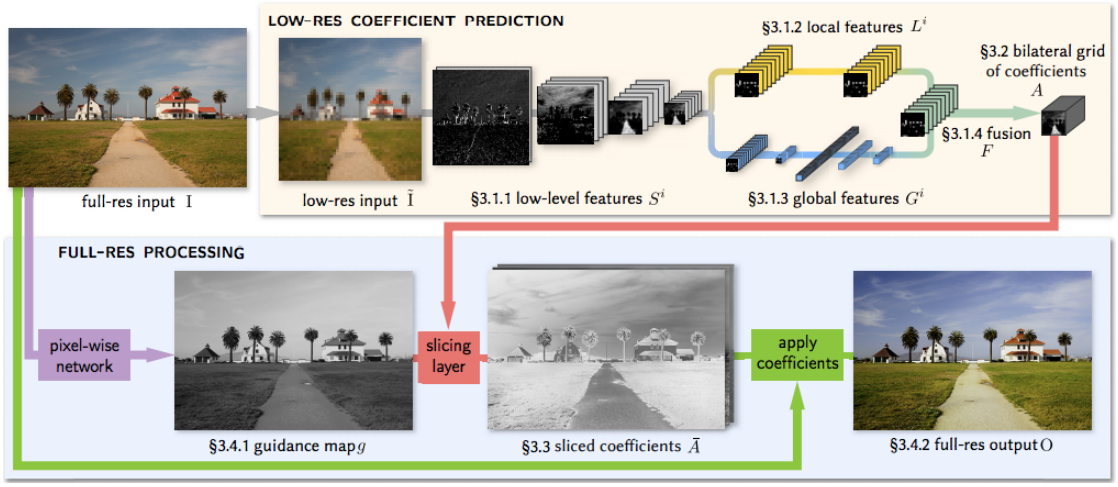


Figure 4.3: Pipeline of HDRNET

We can observe the sketched pipeline of HDRNET in Fig 4.3. We have two main streams: the full-resolution and the low-resolution.
Most operations are done at low resolution, in order to save computational resources.
First, the low-resolution image undergoes a first set of convolutional layers, where we extract low level features (S_i). Then the low-res stream is divided into two paths: one that learns global features (G^i) and one that learns local features (L^i). These two paths are then fused by a learnable linear function F. The low-resolution stream predicts an array A which is a bilateral 3D grid of coefficients.

The full-res stream, instead, leans a guidance map g to upsample the coefficients that are generated by the low-res stream. The final output is created by applying the upsampled coefficients to the original full-resolution image.

The loss is calculated on the final output, which is at full-resolution, so we directly optimize for the impact of the transformations at full-resolution.

4.1.4 Named Curves

In Serrano-Lozano et al. [2024], another approach, based on colors, is used to tone map images. The main idea of this work is to divide the image into color components, in order to manipulate the image separately for each color, using tone curves. We can observe an abstract representation of this approach in figure 4.4.
First, in order to standardize the images (which might be taken with different cameras, so with different characteristics), the images are passed through a backbone model, producing the standardized image $\hat{y}_b$. Then, $\hat{y}_b$ is decomposed into the different colors, using the Color Naming module from Van De Weijer et al. [2009]. This model is applied to every pixel, outputting a probability map for each color category.
The color categories in Van De Weijer et al. [2009] are 11 (*red, blue, green, yellow, pink,*

purple, orange, brown, white, grey, black) and have been chosen after taking into consideration the color names that are most frequent in various cultures. Out of these 11 categories, as some of these colors share similar hues and differ only in intensity, some were grouped together, yielding only 6 categories (*orange-brown-yellow, pink-purple, white-grey-black, red, blue, green*).
For each RGB color channel c and for each color group n, we want to estimate the tone curve. We approximate the tone curve as a Bezier curve (i.e. a parametric smooth curve, defined by a set of discrete points). From the input axis, we sample M control points, calculate the corresponding coordinate on the output axis and parametrize the curve using those points. We fix the first control point in (0,0), so we have M-1 parameters. The block which is used to approximate this curve is called Bezier Control Points Estimator (BCPE).
Once we have the six approximated tone curves, we fuse them using an attention mechanism. The loss of the model is a combination of the L2 loss between the standardized image and the target, the L2 loss between the output and the target and the SSIM loss between the output and the target.

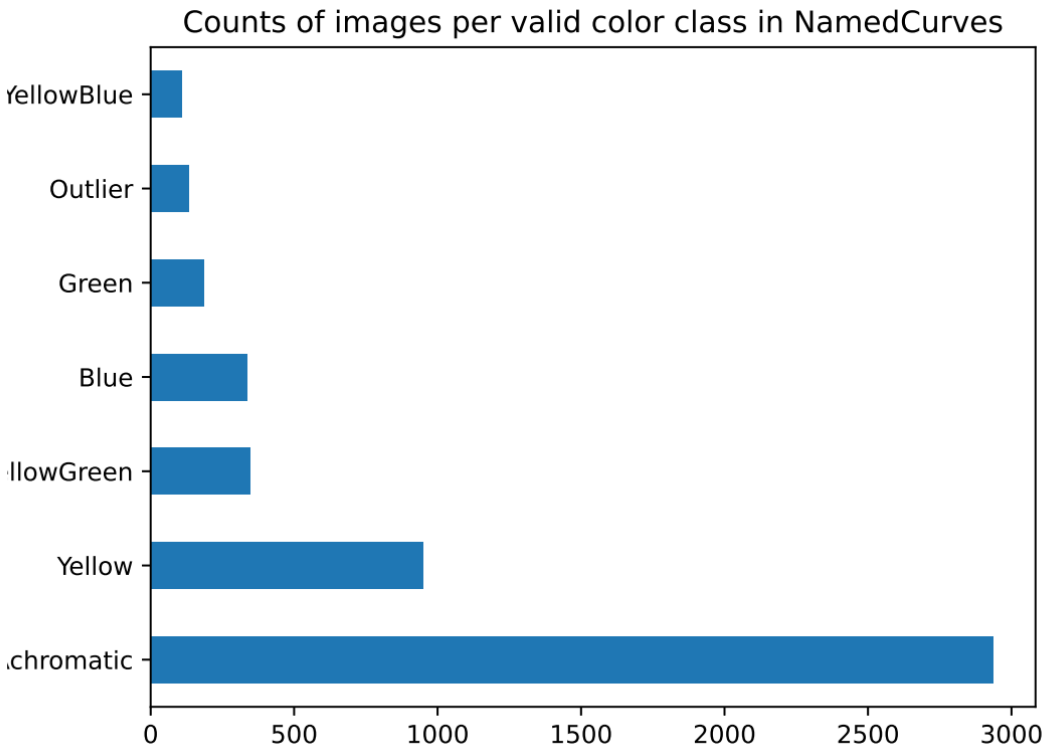


Figure 4.4: Depiction of the NamedCurves network, image from Serrano-Lozano et al. [2024]

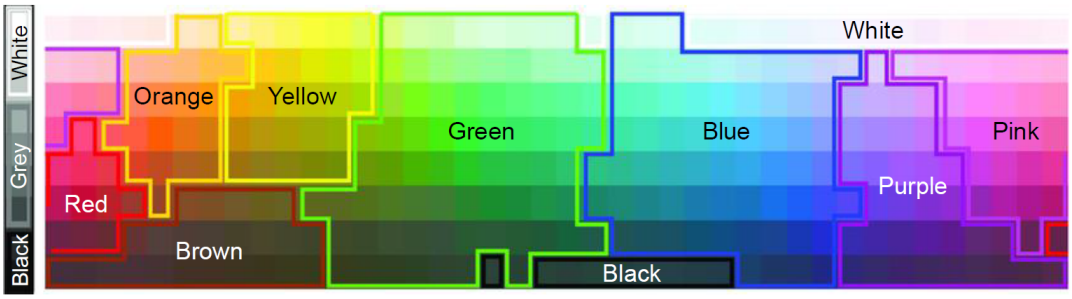


Figure 4.5: Plot of the 11 colors defined in Van De Weijer et al. [2009]

4.1.5 Samples of images produced with a classical Tone Mapper (Reinhard Devlin) and a deep-learning based one (HDRNet)

In this section, we show some examples of images from the MIT5K dataset, processed by two different Tone Mapping algorithms: the classical one by Reinhard Devlin and a deep-learning based one (HDRNet). We want to compare the two outputs both visually and with their PSNR value.

We compare the outputs of our Tone mapping algorithms with the retouched photograph done by one of the experts (ExpertC). We compute the difference between the images using the PSNR.

For HDRNet, we performed a small ablation study of the parameters. We tried the following:

- learning rate : **1e-4**, 1e-3, 1e-2
- weight decay 1e-8, 1e-4, **1e-10**
- batch size: 4, 8, **12**

In bold, the parameters that performed better and thus were chosen for the final training.

In the following images, we show an example of the results of the two tone mapping algorithms. First we have the original picture and the retouched versions by ExpertC.

We can see that the image tone mapped with HDRNET is much more similar to the one retouched by ExpertC. Moreover, for what concerns subjective quality, the second image looks more similar to what the human eye would probably see in this scene. The respective PSNR are 16.68 dB for Reinhard Devlin and 26.98 dB for HDRNET. Nevertheless, we can still notice that the image produced by HDRNET is quite different from the ExpertC image, in particular, the brighter areas (near the sun) are less detailed.

Figure 4.6: Example of image from MIT5K dataset. On the left, the original image, on the right the retouched image by ExpertC

Figure 4.7: Example of image tone mapped using Reinhard Devlin (on the left) and with HDRNET (on the right).

4.2 Denoising

Denoising is one of the most studied topics in Image Processing, its purpose is to improve the quality and clarity of an image, by removing the interference cause by noise, while preserving the sharpness of the edges. In practical settings, when we want to capture some type of data, we may encounter different sources of noise that can distort the original signal (in our case an image) and cause a loss of useful information. In image processing, we measure the three color values (Red, Green, Blue) which are all subject to noise distortions. These distortions are due to the fact that sensor count photons with a certain degree of randomness.
If we apply additional processing steps (e.g. deblurring) to an image with noise perturbations, often we can amplify the noise, therefore it is fundamental that we denoise the images as soon as possible in the image processing pipeline.
There exist many classical algorithms to perform denoising (e.g. bilateral filters, median filters), we focus on a classical algorithm (Non-Local Means) and a deep-learning based

one (NAFNet).

4.2.1 Non-local Means

The first, simplest denoising algorithms usually replace the color value of a pixel with the average of the color values of the pixels in its neighbourhood, leveraging the hypothesis that if we average similar pixels we can reduce the noise perturbations.
It is not certain, though, that similar pixels should be near to eachother, therefore the Non-local Means denoising algorithm Buades et al. [2011] averages pixels that are similar but could also be far from each other. It is not totally *non-local* as it still performs the averages by looking at patches of the image, for computational reasons.
Given a patch of the image $B(p, r)$, centered around pixel p and with size $(2r+1)x(2r+1)$ and colors $u = (u_1, u_2, u_3)$, the resulting color at pixel p will be:

$$\hat{u}_i(p) = \frac{1}{C(p)} \sum_{q \in B(p,r)} u_i(q) w(p, q)$$

where $C(p)$ is a normalization constant. Therefore, we compute a weighted average over the patch.
The weight factor $w(p, q)$ depends on the similarity of the pixels p and q and is computed as an exponential kernel depending on the squared Euclidean distance d^2:

$$w(p, q) = exp\{-\frac{max(d^2 - 2\sigma^2, 0)}{h^2}\}$$

where σ is the standard deviation of the noise and h is a parameter that depends on σ. This weight kernel selects patches that are similar to the pixel p to compute the average: if the patch is similar ($d^2 < 2\sigma^2$) then the value of the kernel will be 1, otherwise if the distance is large, then the kernel's value decreases rapidly.

4.2.2 NAFNet

We study one of the State of the art networks for denoising using limited computing resources, i.e. Nonlinear Activation Free Network (NAFNet) Chen et al. [2022]. Its main advantage is that it has a lower computational cost with respect to other denoisers, without decreasing the performance.
NAFNet achieves lower computational costs by replacing non-linear activating functions (like ReLU and GELU) with multiplications.
In Fig. 4.8, we have the different blocks described in Chen et al. [2022]. The first one, which the authors call Plain Block, is the most basic one, composed of convolution, depthwise convolution and a ReLU activation function. In the second one, which is the block in the authors' baseline architecture, we add Layer Normalization (to smooth the training process), we substitute ReLU with GELU (which helps with the task of deblurring) and add Channel Attention. In the last one, we have the final version of the basic block. It is a simplification of the second block, where: we substitute the GELU activation function with a SimpleGate (which is basically element-wise multiplication) and we substitute

Channel Attention with Simplified Channel Attention (which is composed of channel-wise multiplication and a global average pooling operation).
All these substitutions do not have any impact on the performance of the network, but substantially decrease its computational complexity.

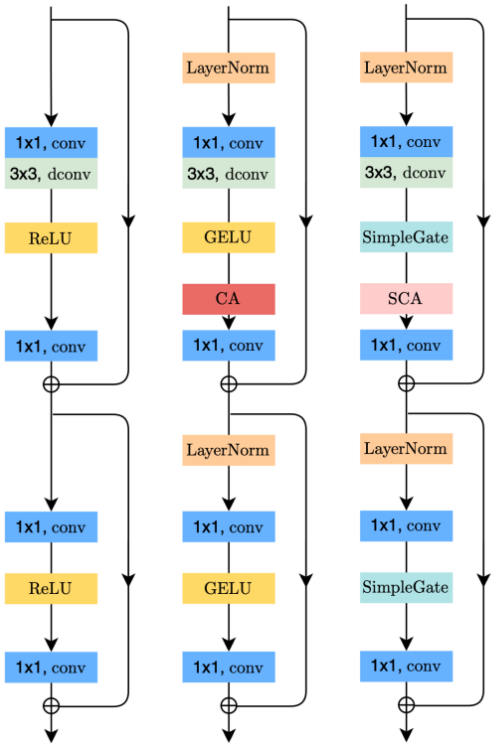


Figure 4.8: Different blocks described in Chen et al. [2022]

4.2.3 Creating a synthetic dataset for denoising

In order to create a dataset for denoising, we keep using the *MIT5K* dataset, and we add some noise on top of it. Therefore we create a synthetic dataset for denoising.
We use as ground truth (and as starting point to add the noise to) the images retouched by ExpertC.
We performed some experiments using a simple Gaussian noise, but it demonstrated to be too simple for the algorithm (NAFNet) to remove and thus difficult to improve with the resampling tecnhique. Therefore, we try to find a more complicated distribution of noise, which is more similar to "real" noise we may encounter in actual images taken with cameras.
We model the noise using a Gaussian and Poisson distribution, following the analysis in Sánchez-Beeckman et al. [2024]. Indeed, in cameras we have two main kinds of noise: shot noise and read noise. Shot noise is related to the arrival process of light photons to the

sensor, this type of noise is dependent on the signal and can be modeled with a Poisson distribution Foi et al. [2008]; on the other hand, read noise is a summary of different noises that occur in the reading process of the sensor, it is approximated as a constant noise, independent of the input signal and modeled with a white Gaussian noise distribution.
The general formula of the distribution of the resulting noise is:

$$x_{read} \sim \mathcal{N}(x_{true} + O, ax_{true} + b)$$

where x_{true} is the value of the input without noise, x_{read} is the noisy input, O is an offset (which we set as zero, for simplicity), a and b are parameters that depend on the ISO value used to capture the image.
As we do not have the ISO values for the images in the MIT5K dataset, we choose to randomize the generation of the noise, selecting boundaries for the values of a and b, from which to sample random values, different for each image. In Fig 4.9, we show an example of the synthetic noisy images, together with its ground truth.

Figure 4.9: Sample of synthetic noisy image (on the left) and ground truth original image (on the right)

4.2.4 Sample of images produced with a classical Denoiser (non-local means) and a deep-learning based one (NAFNet)

In Fig. 4.9, we can observe the outputs of a classical denoising algorithm (Non-local means) and a deep-learning based one (NAFNet). We can note that both pictures present some visible differences from the original ground truth: the first one is very smooth and doesn't contain the details of the most textured parts, while the second one has more details but still contains a certain quantity of noise, especially in the part with the sky and clouds.

Figure 4.10: Sample of results of applying Non-local means (on the left) and NAFNet (on the right)

Chapter 5

Resampling

In Image Enhancement tasks, sometimes algorithms based on deep learning can struggle due to some type of class imbalance, which is different from the usual one we encounter in classification tasks (as we don't really have classes in this case). In our case, instead, the algorithm can struggle because some types of images are under or over-represented in the training dataset. For example, if we have a training dataset composed of images with very low luminance and we test on an image with high luminance, our image enhancement model usually results in low performances.
One of the most common approaches to deal with class imbalance is to resample the training dataset to correct the percentage of classes. We want to apply this technique to image enhancement algorithms and compare the performance before and after the resampling. Therefore, our purpose is to use the previously described clustering algorithm, which produces clusters of classes that depend on the general features of the training dataset's images (e.g. a class could be composed of the images that have average luminosity and whose main color is blue). From this clustered feature space, we can resample images to balance our dataset.
In the following section, we review the literature on resampling techniques. Then, we discuss which resampling and data augmentation strategies we employ in our work.

5.1 Techniques for resampling

In Mohammed et al. [2020], two main resampling techniques are presented and compared: *Oversampling* and *Undersampling*. Undersampling consists of removing some samples from the majority class, while oversampling consists of producing new samples or duplicating the original ones from the minority class.
The main disadvantage of undersampling is that, by deleting random elements from the majority class, we can erase information that could be important to the learning process. Also oversampling has some limitations: it can lead to overfitting (as we duplicate some samples) and it can increase the complexity of the algorithm as we create more data.
The authors experiment with these two resampling techniques applied to an imbalanced dataset using classical machine learning models (Naive Bayes, Decision Trees, SVM,...).

Their results indicate that, in general, random oversampling performs better than random undersampling.

Synthetic Minority Over-sampling TEchnique Chawla et al. [2002] tries to solve one of the problems of oversampling, i.e. the fact that duplicating existing samples may lead to overfitting. Instead of duplicating the sample, we generate new ones following this algorithm:

- cluster the minority class using a clustering algorithm (here the authors used KNN)
- choose one of the k nearest neighbours (a) of a sample (x)
- compute the difference d between the sample and the chosen neighbour ($d = x - a$)
- extract a random number n between 0 and 1
- compute $d2 = n \cdot d$
- create the new synthetic sample by adding this new difference to the original sample $y = x + d2$
- repeat this procedure until we reach the desired number of synthetic data (e.g. if we want to duplicate the minority class, we do this two times for each original sample)

There are many extensions to SMOTE. For example, in Bordeline SMOTE [Han et al., 2005, text], only the borderline samples from the minority class are oversampled. The motivation for this choice comes from the fact that borderline samples are the most complicated to learn for a machine learning algorithm.

5.2 Resampling for Image Enhancement

Among the three techniques discussed before (oversampling, undersampling and SMOTE), we choose to employ **oversampling**. We disregarded undersampling, as it would delete important information contained in the dataset; also SMOTE is not optimal for images, as the synthetic new sample images would not be realistic.
Therefore, we have to deal with the drawback of oversampling, i.e. the possibility of overfitting, in other ways and we study some data augmentation techniques.
In order to perform data augmentation, we can follow two approaches:

1. augment the images with general, universal data augmentation techniques
2. augment the images with techniques ad hoc for the Image Enhancement task

We start by following the easier route, i.e. the first one.

5.2.1 Augmentation by flippling color channels

The first augmentation we try for our dataset is to permute the color channels in RGB. For example, we take the matrix with the intensity values for the Red channel and paste it to the Green channel and vice versa.
Note that, since our dataset is composed of input images (e.g. for Tone Mapping, hdr) and output images (ldr), we need to augment the pairs of corresponding images in the same way. Therefore, we permute the channels with the same order both for the inputs and for the outputs.
As shown in Alg 1, for each image in the training dataset, for each of the possible permutations of the three color channels, we flip the image and calculate in which cluster it belongs. To assign the flipped image to a cluster, we choose the cluster whose centroid is closer to our flipped image.
If the new cluster of the flipped image is one of the clusters we want to augment, we add the flipped image to the training dataset, along with its output.
We stop the iteration when one of these conditions applies: we do not have other images to augment, or when we reach the desired number of images in every cluster.

Algorithm 1 Augment dataset by flipping color channels

```
while counts[minority_clusters] < threshold do
    for each training image img do
        for each permutation permut do
            newimg ← Flip(img, permut)
            newcluster ← GetCluster(newimg)
            if newcluster in minority_clusters then
                AddToData(newimage)
                counts[newcluster] ← counts[newcluster] + 1
```

5.2.2 Augmentation by Inverse Tone Mapping

Another possibility for augmenting the images is to follow route 2, i.e. using a technique which is dependent on our purpose. In this case, we focus on the task of Tone Mapping and propose a solution ad hoc for this objective.
As before, we oversample the images in the minority clusters, by simply copying them. Then, we apply the data augmentation technique to the copies, in order to prevent overfitting.
For each copy, we have the input (HDR) and ground truth output (LDR). We want to create new inputs by applying an inverse tone mapping operation to the output. As tone mapping transforms a HDR image into a LDR image, by applying inverse tone mapping, we can transform a LDR image into a HDR image, obtaining new (slightlty different) inputs that can augment our dataset.
To obtain the inverse tone mapped image, we can simply use classical global tone mapping algorithms, which can be expressed with a function that maps the input (HDR) to the

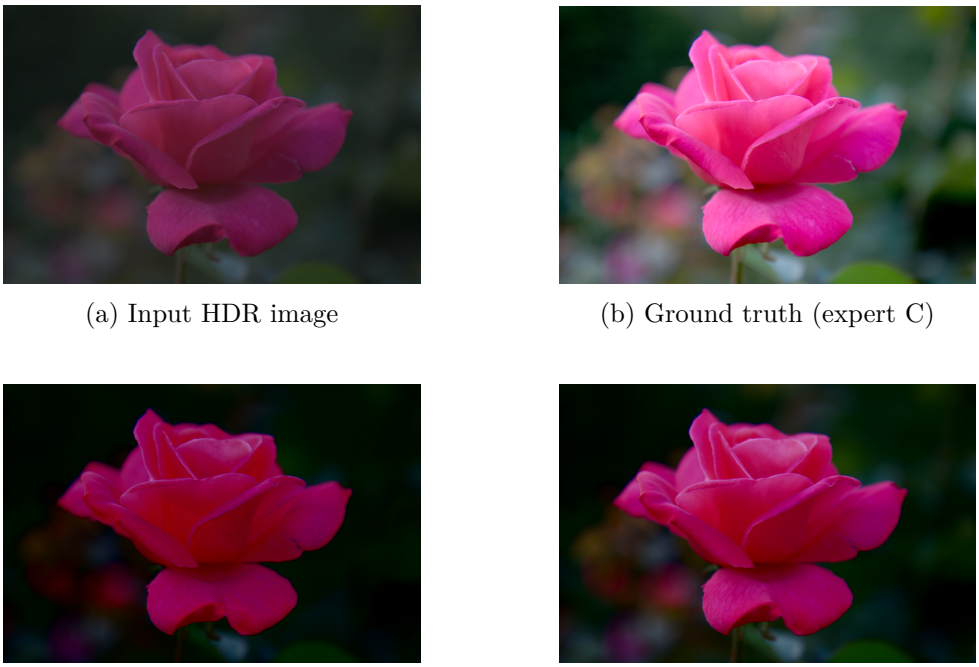
output (LDR). If the tone mapping function can be inverted, we can simply obtain its inverse and apply it to our output (LDR) in order to get a modified input (HDR).
Moreover, by varying the parameters in the inverse tone mapping function, we can obtain different versions of HDR images from the same output LDR image.
As a first example, we can use gamma correction, which can be seen as a simple tone mapping operator Eilertsen et al. [2017]. The formula of simple gamma correction is:

$$V_{out} = AV_{in}^{\gamma}$$

where V_{out} is the output signal, V_{in} is the input signal. A and γ are user-defined parameters. We can invert this function to get V_{in} as a function of V_{out}:

$$V_{in} = \left(\frac{1}{A} \cdot V_{out}\right)^{\frac{1}{\gamma}}$$

We fix the value of A to 1, but we vary the value of γ for each copy in order to have different augmentations for the same image.



(a) Input HDR image

(b) Ground truth (expert C)

(c) Augmented input $\gamma = 0.3$

(d) Augmented input $\gamma = 0.6$

Figure 5.1: Example of augmentation using inverse gamma correction, with different γs

As we can see in the example in Fig 5.1, the inverse gamma correction produces different augmented images depending on the value of γ, which are similar but not identical to the original input HDR image.

Chapter 6

Experiments and Results on Tone Mapping

In this chapter, we explore the hypothesis that the performance of a Tone Mapping algorithm can vary depending on the clusters we precedently defined using different algorithms. Due to the considerations carried out in Sec. 3.5.4, we focus this analysis on the clusters created with two unsupervised clustering algorithms: *CARD* and the one based on descriptors generate by *ResNet-Places365*.
As dataset, we employ the full 5000 images from MIT5K, select ExpertC's retouched images as the ground truths. We split the dataset into training and test sets with 75 % of training samples. We choose to use HDRNET, described in Sec. 4.1.3, as a baseline Tone Mapping algorithm.

6.1 Performance with respect to Clusters created with CARDIE

We run the clustering algorithm on the full dataset, getting 12 clusters.
Now we plot the performances in terms of PSNR of HDRNET, divided into the different clusters produced by CARD.
First, in Fig 6.1, we show the statistics on the training dataset. Then, in Fig 6.2, we show the same plot for the test set.
In the plots, we have two quantities for each cluster. On the scale on the left, we show the mean PSNR for the cluster, instead, on the scale on the right, we plot the number of images in that cluster. We also have a horizontal line with the average PSNR for the whole set.

From these two plots, we can observe that the performance of the tone mapping algorithm is lower in some clusters. In particular, on the test set, the performance is below average in almost all clusters with high luminosity (except for the BrY cluster). Also one

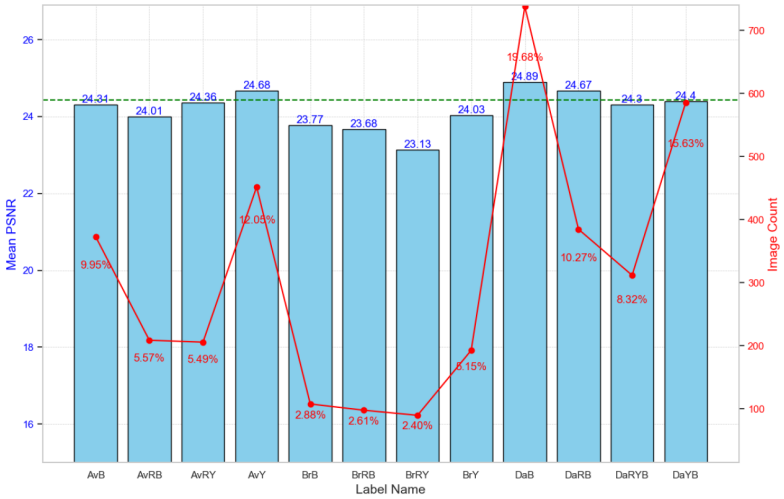


Figure 6.1: Average PSNR values of HDRNET predictions on the train set, aggregated by cluster. Count of number of images per cluster on the scale on the right

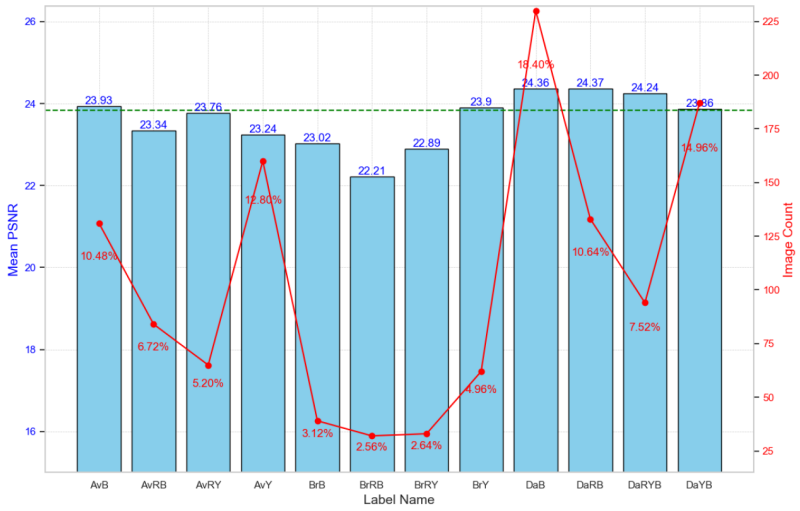


Figure 6.2: Average PSNR values of HDRNET predictions on the test set, aggregated by cluster. Count of number of images per cluster on the scale on the right

of the clusters with average luminosity presents a lower performance (the *AvRB* cluster). We think this may be due to the fact that images with these characteristics (so especially bright images) are less present in the training dataset. We can confirm this also by looking at the plots, where we can see that the bright clusters are composed of few images in the training dataset.
From this analysis, we can confirm our hypothesis, i.e. that having training datasets that are unbalanced in terms of characteristics of images will lead to poor performances on

those kinds of images. We can also see that our clustering algorithm seems to be able to divide the images into relevant clusters, that capture the different characteristics of the images which have an impact on the performance.

6.2 Performance with respect to Clusters created with ResNet-Places365

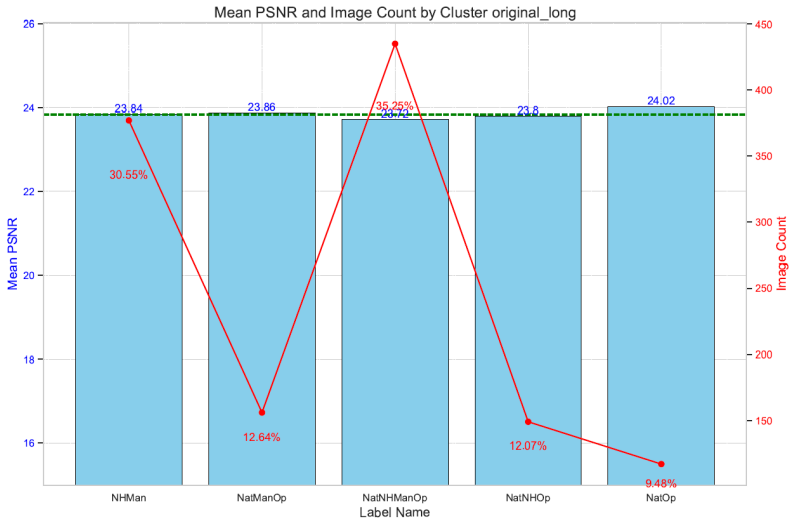


Figure 6.3: Average PSNR values of HDRNET predictions on the test set, aggregated by cluster of ResNet-Places365. Count of number of images per cluster on the scale on the right

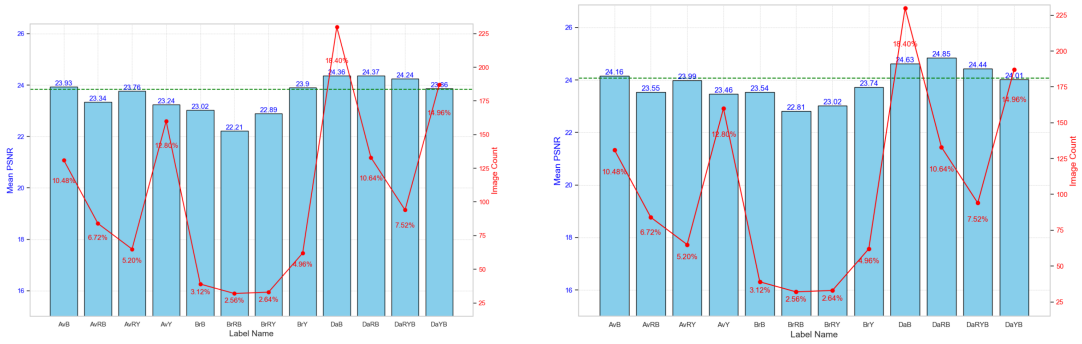
Carrying out the same procedure as in the last subsection, but in this case for the clusters created with the descriptors generated with *ResNet-Places365*, in Fig.6.3, we show the performance in the same type of plot as before.
We can notice that the performance of the algorithm is very uniform among the different classes; moreover, the performance seems to not be correlated to the number of images in the clusters, in fact clusters with few images still have average performance.
Due to this analysis, we can say that, even if the clusters seemed to be relevant to the Tone Mapping algorithm as seen in Sec 3.3, the performance does not vary according to them. Therefore, we disregard also this clustering algorithm for further analysis, and we will not run the resampling tests on these clusters.

6.3 Performance of HDRNET with training dataset resampled according to CARDIE's clusters

In this section, we will show how the performance of HDRNET changes when we resample the dataset with the techniques described in Sec 5.2; we discard the results of the Flipping Augmentation, as they didn't show any noticeable difference in performance over the original dataset.

6.3.1 Simple oversampling of the bright classes

In order to resample the minority clusters in the training dataset, we first try to simply oversample the minority classes. Simple oversampling consists of duplicating the images in the minority class as many times as we need, in order to make the dataset more balanced.
In this experiment, we oversample all the Bright classes (BrB, BrY, BrRB, BrRY).
We copy the images from the bright clusters three times, in order to reach about 400 samples in the class. Then, we train the network and compare the performances between the original dataset and the resampled one.



(a) Before resampling

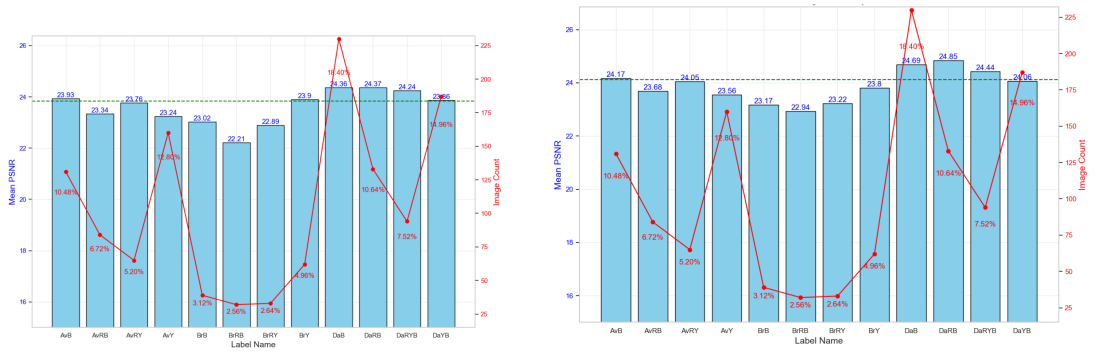
(b) After resampling

Figure 6.4: Comparison of the performance of HDRNET on the clusters after the oversampling of all the bright clusters, performance on the test set

In Fig 6.4, we can observe the comparison between the performances on the test set of the network trained on the original training dataset, and the model trained on the oversampled dataset. We can observe that the performance hasn't improved a lot on average, but it has improved more on the oversampled clusters, which was expected. Also some other cluster shows improvement after the resampling, we think this could be due to the fact that, even in 'dark' or 'average' classes there could be some bright spots, which can benefit from the resampling of bright images.
We think this oversampling technique may be too naive and, in the next section, we move on to more sophisticated ways, which include data augmentation.

6.3.2 Oversampling with Inverse Tone Mapping Augmentation

In this section, we show the results on the performances of HDRNet trained using the Inverse Tone Mapped dataset.



(a) Average PSNR values of HDRNET predictions on the test set, aggregated by cluster. Before resampling

(b) Average PSNR values of HDRNET predictions on the test set, aggregated by cluster. After resampling with ITM augmentation

Figure 6.5: Comparison of the performance of HDRNET on the clusters after the oversampling using inverse tone mapping of all the bright clusters, performance on the test set

From Fig 6.5, we can observe the comparison between the results of the model trained on the original training dataset and the model trained with the dataset created by augmenting the bright clusters and augmenting them using the inverse tone mapping procedure.
We can see, similarly to the results on the simple oversampling, that there is a slight improvement of performances, especially on the bright clusters. The improvement is larger than the one found using simple oversampling.

6.4 Summary of average performances on the test set of the different models

Original model	Simply oversampled model	ITM model
23.83	24.05	24.11

Table 6.1: Summary of average results on three models trained with three different datasets: the original one, the simply oversampled one and the augmented with Inverse Tone Mapping one

In Tab 6.1, we can see a summary of the average performance on the three different models we trained. We can observe a slight improvement with the simply oversampled

training dataset and a slightly bigger improvement with the dataset augmented via Inverse Tone Mapping.

6.5 Sample Images showing differences

In this section, we will show some outputs of the model which show noticeable differences between the different models.
First, we show a picture where the simple oversampling improved the performance even if the cluster was not one of the resampled ones. We can observe, in Fig 6.6, that the output of the oversampled model is more visually natural, especially in the region near the sun, whereas in the original output, there are some artifacts in the brighter region of the image. The two outputs have a difference of 1.5 dB in PSNR.
This image shows an example of a non-bright image with bright spots that improves in the model trained with the oversampled dataset.

Figure 6.6: Sample image from the AvB cluster, 1.5 improvement with the oversampled dataset

In Fig 6.7, we can observe the image where the oversampling model improved the most. We can clearly see a green color cast in the original output and few artifacts that make the image look unnatural. The output of the oversampled model is definitely better and resembles a lot the target. Nevertheless, this is an edge case, as the input HDR and LDR images are very similar and both very bright.

In Fig 6.8, we can see the opposite situation compared to the horse image. This is the output where the difference in performance between the original model and the

Figure 6.7: Sample Image from the BrRB cluster, showing improvement of 12 dB in PSNR with the oversampled model

oversampled model is the largest, so the oversampled model produces a worse picture. Visibly, we can notice a brown color cast, and some differences in the bright parts of the image.

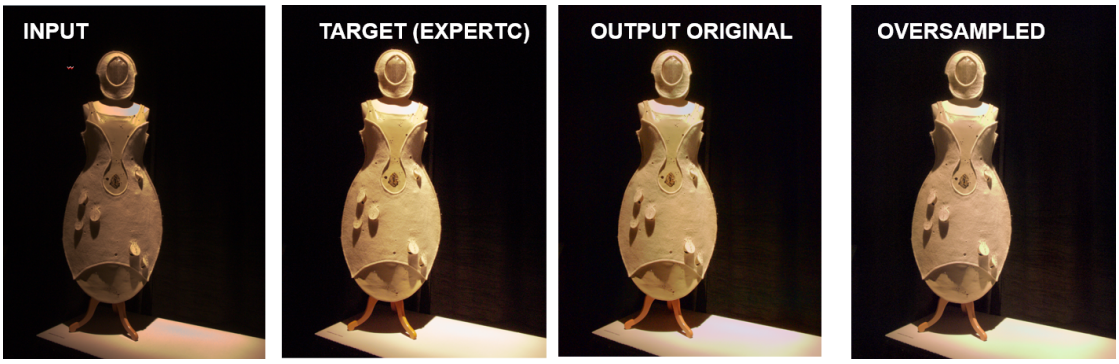


Figure 6.8: Sample Image where the oversampling degrades the quality of the image (-7dB)

Lastly, in Fig 6.9, we can see an example of a case where the inverse tone mapping augmentation improves the performance of the model compared to the simple oversampling. We can notice that the output of the oversampled model is not at all natural to the human eye, with a dark green color cast and some artifacts in the window. Instead, the output of the ITM augmented model is much more similar to the original one, especially for what concerns the window and the shadows, while the color of the house is still far from the target.

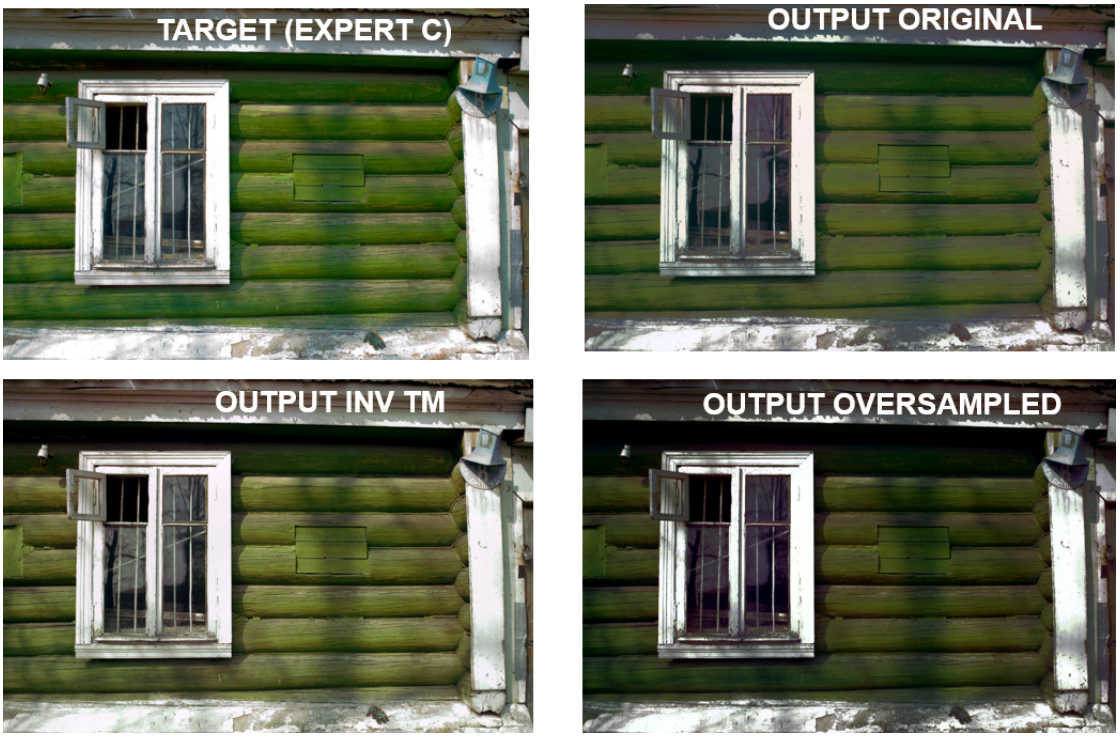


Figure 6.9: Sample Image where the ITM augmentation is better compared to the simple oversampling

Chapter 7

Experiments and Results on Denoising

In this chapter, we will explore the performance of the denoising algorithm NAFNet, described in Sec. 4.2.2, with respect to the clusters defined using CARDIE.
We employ the MIT5K dataset, with synthetic noise generated as illustrated in Sec: 4.2.3, we split it into training (75 %) and test (25 %) sets.
As we added a high level of noise on the synthetic noisy dataset, this perturbation may impact the statistics computed by CARDIE to create the clusters. Therefore, before extracting the features needed by the unsupervised clustering algorithm, we apply a classical denoising algorithm. As classical denoising algorithm, we employ OpenCV's implementation of the Non-local Means algorithm described in Sec. 4.2.1.

Table 7.1: Clusters statistics on the denoising dataset

c_l	labels	% c_l
AvB	{average lum, blue}	6.84
BrB	{bright lum, blue}	6.88
DaB	{dark lum, blue}	17.34
DaY	{dark lum, yellow}	24.26
AvY	{average lum, yellow}	19.52
BrY	{bright lum, yellow}	25.16

In Tab. 7.1, we can observe there are two under-represented classes, namely, *AvB* and *BrB*; in 7.1 we have the corresponding plot with the performance in terms of PSNR, of the images in the different clusters defined by CARDIE. Despite the unbalanceness of the training dataset, the performance is quite uniform, and it is not correlated with the number of images in each cluster.
In order to continue with our analysis, we follow two different routes, which arise from

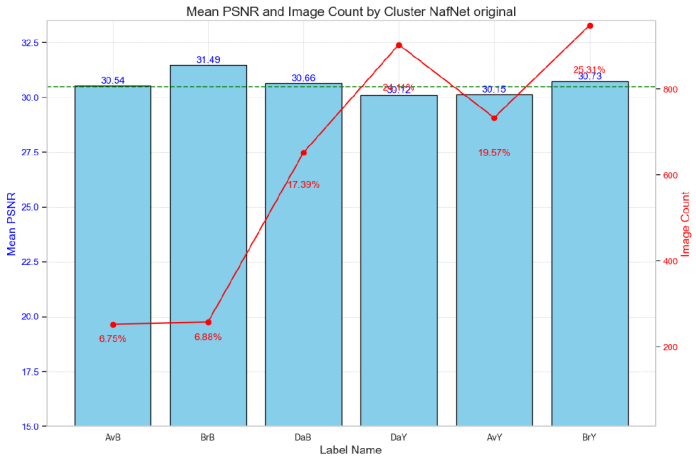


Figure 7.1: Performance of NAFNet grouped by CARDIE's clusters

two different hypothesis:

- as we created a synthetic dataset, with random levels of noise, coming from the same distribution, we do not have any clusters whose images are harder for the algorithm to learn. If, instead, we had natural noisy data, maybe we would have different performance for some clusters. Therefore, we create an artificial situation where some classes are harder to learn: we add more noise on the least represented classes (*AvB* and *BrB*).
- the descriptors we used to create CARDIE's clusters are not relevant to the denoising task, so we explore other features that can define alternative clusters. As the areas in an image that contain a more varied texture seem to be harder to denoise, we choose to construct clusters based on the context of *texture*, defined by GLCM's features (see Sec. 2.1).

We will further detail both procedures and show the corresponding results in the following sections.

7.1 NAFNet performance with respect to CARDIE's clusters with additional noise

The first hypothesis for why the performance does not vary according to the clusters is that, as we created a synthetic dataset, the denoising algorithm will treat the different clusters in the same way (on average, as we added noise depending on randomly sampled parameters). Therefore, to simulate a more realistic situation where some of the images have different levels of noise, we create another synthetic dataset.
In this new dataset, some classes have a higher level of noise and therefore are harder

for the Denosing algorithm to learn. We still use the noise distribution explained in Sec. 4.2.3, using higher values for the a and b parameters for the two under-represented clusters (AvB and BrB).

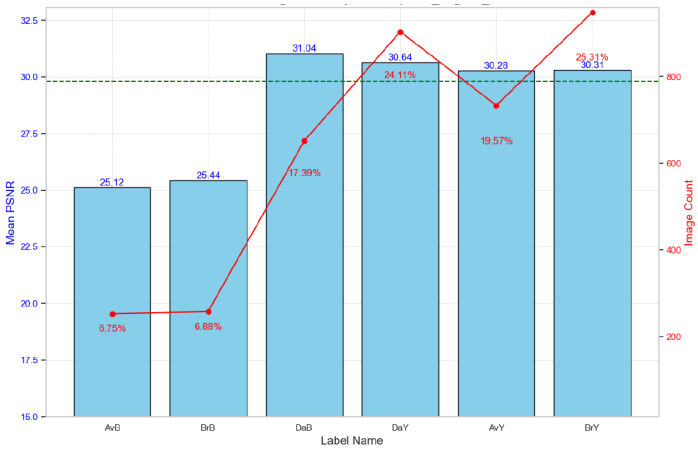


Figure 7.2: Performance of NAFNet on the synthetic dataset with additional noise

In Fig. 7.2, we show the performance of NAFNet, in terms of PSNR, on the new synthetic dataset with additional noise on the two un-represented clusters. We can indeed observe that the algorithm struggles to remove the noise from images in the first two clusters, showing that we achieved our purpose of creating a harder problem; in the next paragraph, we will try to resample those classes in order to improve the performance.

7.1.1 Performance with oversampling and some sample images

In the following experiment, we try to resample the two under-represented clusters defined by CARDIE, AvB and BrB. We resample them with simple oversampling, as before, by copying the under-represented clusters 3 times.
In Fig. 7.3, we show the performance of NAFNet on the clusters, after the resampling. We can notice that every class improves the performance after the resampling, in particular, as was expected, the two under-represented classes have the biggest improvement (+2.35 dB in AvB, +2.73 in BrB). On average, we have an improvement of +1.1 dB.

We can also visually observe the improvement of the resampled model with a sample of outputs, in Fig. 7.4, where we can note that the output of the model trained with the oversampled dataset is smoother and contains less noise. Nonetheless, the output still has a high level of noise, as we can notice in particular in the part of the image that contains some writing. This may be due to the fact that we added too much noise to the synthetic dataset, moreover in future work we plan to train the model for a larger number of epochs,

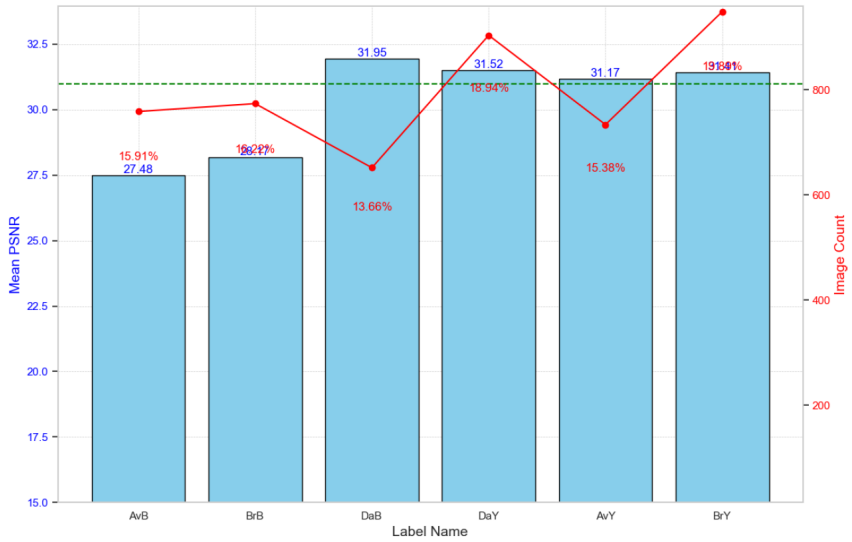


Figure 7.3: Performance of NAFNet on the synthetic dataset with additional noise, after resampling

which may improve the image quality.

Original model	Oversampled Model (CARDIE)	Delta
29.80 dB	30.98 dB	1.1 dB

Table 7.2: Summary of average results on the original model and the oversampled one (NAFNet)

7.2 NAFNet performance with respect to clusters based on clusters

In this section, we take into consideration the second hypothesis presented in Sec. 7, i.e. that we may need another type of descriptors to create the clusters used to resample the training dataset.
We try to find an alternative feature that could be meaningful to the *Denoising* task.
By applying common sense and examining the images, it becomes apparent that the denoising algorithm has difficulty producing high-quality results in areas with more texture, where the surfaces are less smooth (e.g. in images with some foliage or with some kinds of clothes' textile). Thus, we try to find a numerical measure to define how *textured* an image is, then we use this measure to create the clusters.
One of the most established ways to perform texture analysis (especially used in the medical field, for example to classify the masses in mammograms Mohanty et al. [2011]) is the

(a) Output of baseline (26.01 dB)

(b) Output of resampled model (27.7 dB)

Figure 7.4: Sample of output of the baseline model trained on the dataset with additional noise (left) and on the resampled dataset (right)

GrayLevel Co-occurrence Matrix (GLCM), which we described in Sec. 2.1.
For simplicity (and for computational reasons) we choose to use only one of the features we can extract from the co-occurrence matrix. As the three features we discussed are all pretty similar, we choose *Contrast* to represent texture. We compute this metric for all the images and plot its distribution in Fig. 7.5; in Fig. 7.6 where we can also observe a sanity check for the texture feature extraction. On the right top we have the image in the MIT5K dataset with the highest contrast and on the right bottom the one with the lower contrast. Visually, we can confirm that our definition of *texture* is reasonable, thus we use it to define the new clusters.

7.2.1 Creating clusters with texture

Using the histogram in Fig: 7.5, we try to divide the histogram into bins, in order to create clusters. We divide the possible texture values into equally spaced segments and assign a label (number) depending on the bin (the higher the number, the more contrast is in the image).
We choose to use 9 bins, and group the last 4 clusters into a single one, as they had a

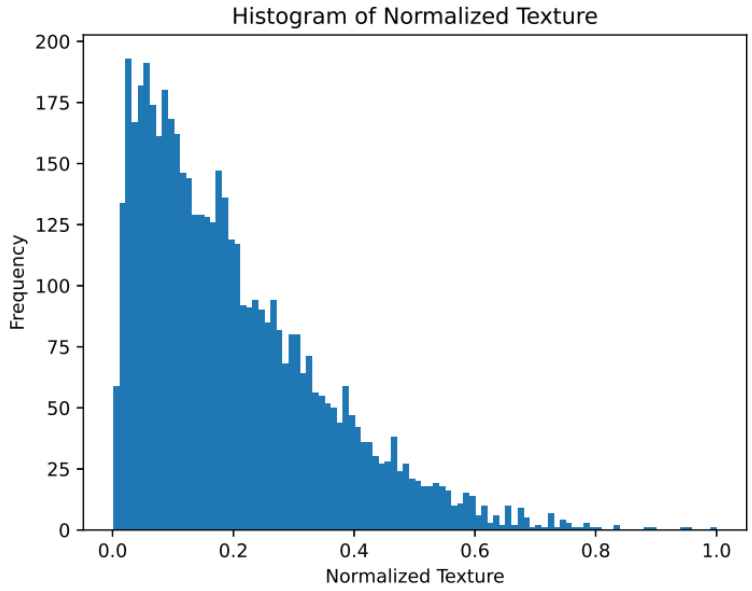


Figure 7.5: Histogram of contrast of the MIT5K dataset

Figure 7.6: Image with the lowest contrast (left) and highest contrast (right)

very small number of samples. We acknowledge that this technique to create the clusters is quite subjective and it may be needed to perform a more thorough study on the texture distribution. Anyway, we show the result of these arbitrary classes in Tab.7.3.
From the image in Fig: 7.7, we can observe that the performance of the denoising algorithm decreases as the texture increases (as it was expected) and the clusters with high contrast are under-represented with respect to the ones with low contrast.

Following the reasoning carried out above, we can proceede with the resampling experiments, based on the clustering according to the *texture* metric, which we will show in the next section.

Table 7.3: Clusters with texture statistics

c_l	textute range	% c_l
0	0-0.11	35.80
1	0.11-0.22	28.10
2	0.22-0.33	18.12
3	0.33-0.44	10.02
4	0.44-0.55	5.08
5	0.55-1.0	2.86

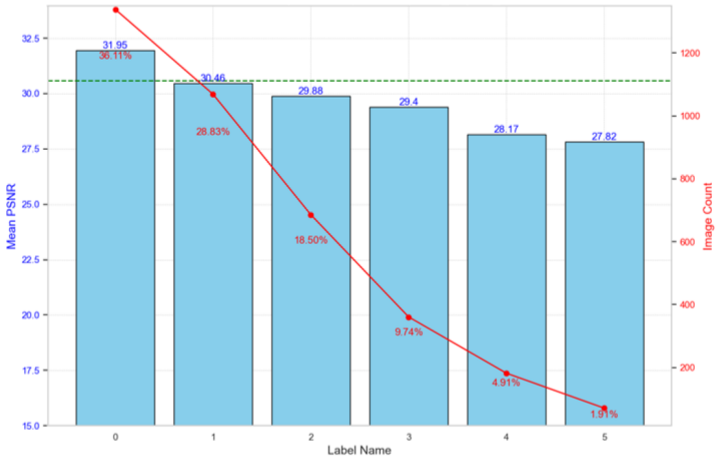


Figure 7.7: Performance of NAFNet according to the different texture clusters

7.2.2 Performance with oversampling on the texture clusters and some sample images

In this section, we describe our experiments with resampling the training dataset used for NAFNet according to the clustering method described above, which is based on GLCM's texture metric. We will also show some images that visually confirm our results.
Starting from the distribution of classes in Tab. 7.3, we oversample (by simply copying the images 4 times) the most under-represented class, i.e. class 5, which is also the one containing the images with higher texture.
In Fig. 7.8, we show the performance of NAFNet according to the clusters, after the resampling. We can see that the PSNR in all the clusters is higher with respect to the one achieved by the baseline, of course, the biggest improvement is displayed by the images with the highest texture. In Tab.7.4 we show a summary of the different performances of the baseline and oversampled models, which show an improvement (on average) of 1.8 dB.

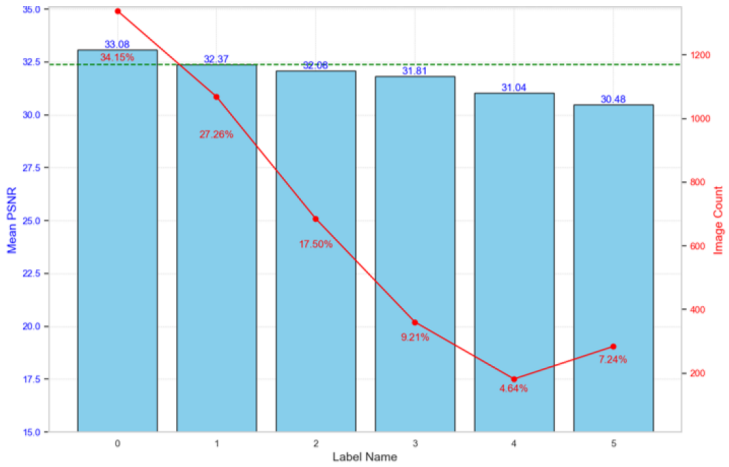


Figure 7.8: Performance of NAFNet according to the different texture clusters, after resampling

Original model	Oversampled Model (texture)	Delta
30.57 dB	32.38 dB	1.81 dB

Table 7.4: Summary of average results on the original model and the oversampled one (NAFNet), using texture clusters

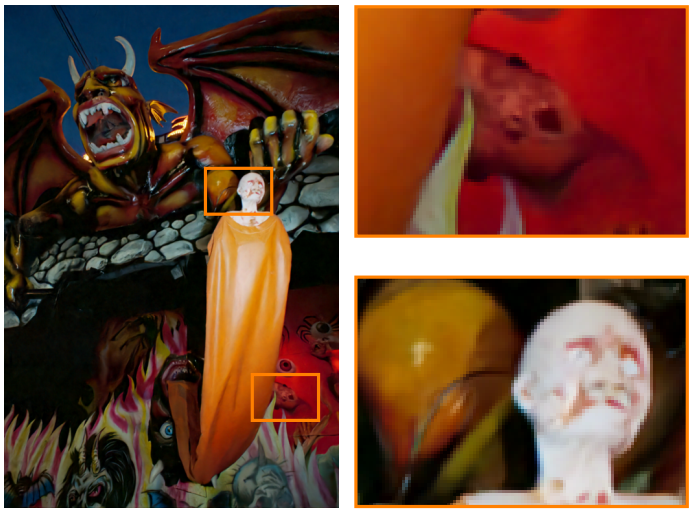
In Figures from 7.11 to 7.10, we can observe some examples of the visual differences of the outputs of the baseline model and of the resampled model. The images are quite similar from a certain distance, but if we zoom into the details, we can observe that the output of the resampled models are less blurry and contain more edges, especially where faces or particular textures are present. We can therefore conclude that also this resampling strategy was successful in improving the performance of the denoising algorithm, both visually and in terms of PSNR.

(a) (b)

Figure 7.9: Sample image 2, original output (32.97 dB)

(a) (b)

Figure 7.10: Sample image 2, resampled output (26.67 dB)



(a) (b)

Figure 7.11: Sample image, original output (30.07 dB)

(a) (b)

Figure 7.12: Sample image, resampled output (37.17 dB)

Conclusion and Future Work

In conclusion, we presented here a study on how the composition of the training dataset can impact the performance of some Image Enhancement algorithms.
In particular, we analyzed the in-house unsupervised clustering algorithm, *CARDIE*, focusing on the *MIT5K dataset*. Using the clusters obtained by *CARDIE*, we resampled the training dataset of some state-of-the-art algorithms in Image Enhancement, focusing on Tone Mapping and Denoising. We compared the performance of the algorithms before and after the resampling operation, showing that it leads to substantial improvements, both quantitatively and visually.
Therefore, we can conclude that the clusters generated by CARDIE are pertinent to the Image Enhancement tasks under consideration, and resampling based on these clusters is a beneficial strategy to enhance the performance of the algorithms.
Moreover, resampling based on the texture shows promise in enhancing the performance of Denoising algorithms, even though it might require thorough investigation due to its subjective nature.
Broadening our perspective, the long-term objective, which this thesis is part of, is to generalize CARDIE to a wide range of CNN-based algorithms, enabling automatic clustering and resampling of datasets for various tasks. In particular, we aim to develop new sufficiently general descriptors to apply CARDIE to a large class of algorithms.
Another aspect we think requires further attention in the future is the study of CARDIE's scalability. Regarding the dataset we employed here (MIT5K), the time consumption of the clustering algorithm is quite low (about 15 minutes). However, it will be necessary to study how this changes when we use larger datasets.
We believe this work could be a step towards making CNN-based Image Enhancement algorithms more explicable to users, as the features used to create the clusters are fairly descriptive and easy to grasp for humans. By understanding the composition of the training dataset in terms of these features, and how it impacts the performance of the deep learning algorithm, we could gain a more precise idea about what the learning process is focusing on.

Acknowledgements

Firstly, I would like to thank my parents, Rinuccia and Michelino, who supported me (both economically and emotionally) through my studies. Thank you for offering me a safe space when I was at my lowest.

To my boyfriend, Alberto, who was there from the beginning of this journey. Thank you for understanding my quirks and for helping me figure out how to become an adult.

To all the friends I had before starting university and the ones I made along the way. Thank you for making my life more fun and for being there when I wanted to complain or celebrate.

Thank you to everyone here at the Nice Research Center: you made me feel welcome and I sincerely enjoyed the time I spent with you. In particular, I'd like to thank Luca, my industrial supervisor, for all the support and guidance throughout this project. Your insights and feedback have been super helpful, and I couldn't have done this without you. Thanks to Nicola, for helping me with the Denosing part and for teaching me about luxury watches; thanks to Philippe, for being my personal French tutor here at Huawei; thanks to Praveen, for his insights and precious suggestions.

I would also like to thank my academic supervisors, prof. Pietro Michiardi and prof. Paolo Garza, for providing helpful insights in my thesis.

Lastly, I'd like to thank my laptop, which was my faithful companion through the 5 years and a half I spent at university and that sadly passed away during the last month on my internship. You will be missed.

Bibliography

Peter GJ Barten. *Contrast sensitivity of the human eye and its effects on image quality*. SPIE press, 1999.

Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Non-Local Means Denoising. *Image Processing On Line*, 1:208–212, 2011. https://doi.org/10.5201/ipol.2011.bcm_nlm.

Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input/output image pairs. In *CVPR 2011*, pages 97–104. IEEE, 2011.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. *arXiv preprint arXiv:2204.04676*, 2022.

Gabriel Eilertsen, Rafal Konrad Mantiuk, and Jonas Unger. A comparative review of tone-mapping algorithms for high dynamic range video. In *Computer graphics forum*, volume 36, pages 565–592. Wiley Online Library, 2017.

Alessandro Foi, Mejdi Trimeche, Vladimir Katkovnik, and Karen Egiazarian. Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. *IEEE transactions on image processing*, 17(10):1737–1754, 2008.

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

Hui Han, Wenyuan Wang, and Binghuan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. pages 878–887, 01 2005.

Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, 1973. doi: 10.1109/TSMC.1973.4309314.

Rafal K Mantiuk, Param Hanji, Maliha Ashraf, Yuta Asano, and Alexandre Chapiro. Colorvideovdp: A visual difference predictor for image, video and display distortions. *arXiv preprint arXiv:2401.11485*, 2024.

Leland McInnes, John Healy, Steve Astels, et al. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205, 2017.

Roweida Mohammed, Jumanah Rawashdeh, and Malak Abdullah. Machine learning with oversampling and undersampling techniques: overview study and experimental results. In *2020 11th international conference on information and communication systems (ICICS)*, pages 243–248. IEEE, 2020.

Aswini Kumar Mohanty, Swapnasikta Beberta, and Saroj Kumar Lenka. Classifying benign and malignant mass using glcm and glrlm based texture features from mammogram. *International Journal of Engineering Research and Applications*, 1(3):687–693, 2011.

Erik Reinhard and Kate Devlin. Dynamic range reduction inspired by photoreceptor physiology. *IEEE transactions on visualization and computer graphics*, 11(1):13–24, 2005.

Yasir Salih, Aamir S Malik, Naufal Saad, et al. Tone mapping of hdr images: A review. In *2012 4th International Conference on Intelligent and Advanced Systems (ICIAS2012)*, volume 1, pages 368–373. IEEE, 2012.

Marco Sánchez-Beeckman, Antoni Buades, Nicola Brandonisio, and Bilel Kanoun. Combining pre-and post-demosaicking noise removal for raw video. *arXiv preprint arXiv:2410.02572*, 2024.

David Serrano-Lozano, Luis Herranz, Michael S Brown, and Javier Vazquez-Corral. Namedcurves: Learned image enhancement via color naming. *arXiv preprint arXiv:2407.09892*, 2024.

Joost Van De Weijer, Cordelia Schmid, Jakob Verbeek, and Diane Larlus. Learning color names for real-world applications. *IEEE Transactions on Image Processing*, 18(7):1512–1523, 2009.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Hojatollah Yeganeh and Zhou Wang. Objective quality assessment of tone-mapped images. *IEEE Transactions on Image processing*, 22(2):657–667, 2012.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.