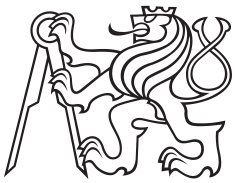
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Domain Generalization in Image Retrieval through Training Data Synthesis

Albert Möhwald

Supervisor: Ing. Tomáš Jeníček
Field of study: Open Informatics
Subfield: Cybersecurity
January 2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Möhwald** Jméno: **Albert** Osobní číslo: **474592**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Kybernetická bezpečnost**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Překonání problému různých vizuálních domén ve vizuálním vyhledávání pomocí generování trénovacích dat

Název diplomové práce anglicky:

Domain generalization in image retrieval through training data synthesis

Pokyny pro vypracování:

1. Familiarize yourself with conditional Generative Adversial Networks (GAN) for the purpose of training data synthesis.
2. Consider the downstream task of image retrieval across visual domains, such as day and night. Employ two different visual domain pairs in evaluation, such as day with night or natural images with sketches.
3. Train GAN to synthesize training data for a deep embedding model in order to aid the downstream task of image retrieval across visual domains.
4. Examine different GAN architectures, comparing more traditional models with the contemporary ones.
5. Evaluate different visual domains of synthesized training data, quantifying their usefulness during training for the downstream task.
6. Provide code that follows best practices and its documentation.

Seznam doporučené literatury:

[1] Tomas Jenicek and Ondrej Chum. No fear of the dark: Image retrieval under varying illumination conditions. In ICCV, 2019.
[2] Che-Tsung Lin, Sheng-Wei Huang, Yen-Yi Wu, and Shang-Hong Lai. Gan-based day-to-night image style transfer for nighttime vehicle detection. IEEE T-ITS, 2020.
[3] Filip Radenovic, Giorgos Tolias, and Ondˇrej Chum. Deep shape matching. In ECCV, 2018.
[4] Hsin-Ying Lee, Hung-Yu Tseng, Qi Mao, Jia-Bin Huang, Yu-Ding Lu, Maneesh Singh, and Ming-Hsuan Yang. Drit++: Diverse image-to-image translation via disentangled representations. IJCV, 2020.
[5] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In ECCV, pages 319–345. Springer, 2020. 2, 3, 5, 8

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Tomáš Jeníček skupina vizuálního rozpoznávání FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **15.09.2022** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **19.02.2024**

Ing. Tomáš Jeníček podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Acknowledgements

First of all, I would like to express my sincere thanks to my supervisor Ing. Tomáš Jeníček for his patience, high amount of time, numerous constructive consultations, and guidance he provided me with this thesis.

I would also like to give my thanks to prof. Mgr. Ondřej Chum, Ph.D., MSc. Nikos Efthymiadis, and Ing. Nikolaos–Antonios Ypsilantis for being helpful in the problem discussion and explaining their prior work. Without their help, this work would not get this far.

Declaration

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 10. února 2023

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 10. January 2023

Abstract

Image retrieval based on deep neural networks relies on a large number of diverse training images. The challenge of severe visual appearance changes, specifically day–night and photo–sketch, is addressed using synthesized training images as a form of data augmentation. GANs are studied to learn image–to–image translation allowing to generate more training data of the scarce domain. For the day–night task, image representation is cast as metric learning and the GAN generator is used to generate diverse day–night pairs for the training. Various generator architectures are evaluated, including a novel lightweight GAN architecture that preserves the content between the original and synthesized images with edge consistency and simultaneously trains an edge detector to operate on both day and night images. For the photo–sketch task, no training sketches are available, which is tackled with an edge detector, that transforms photos into edge images, and then, the GAN generator is used to thin the edges that approximate sketch images. The proposed data augmentation approach outperforms prior work and improves over the current state–of–the–art Tokyo 24/7 day-night image retrieval benchmark while maintaining performance on the Oxford and Paris benchmarks. However, similar GAN–based augmentation does not surpass handcrafted augmentation used for sketch recognition.

Keywords: Image Retrieval, Data Augmentation, Generative Adversarial Network, Image-to-image Translation

Supervisor: Ing. Tomáš Jeníček

Abstrakt

Vizuální vyhledávání obrázků založené na hlubokých neuronových sítích se opírá o velké množství různorodých trénovacích obrázků. Problém značných vizuálních změn, konkrétně den–noc a fotografie–skica, je řešen pomocí syntetizovaných trénovacích obrázků formou rozšíření datové sady. GANy jsou studovány za účelem naučení překladu obrazů, aby bylo možné generovat více trénovacích dat z nedostatkové domény. Pro úlohu den–noc je reprezentace obrazu naučena přes učení metriky, kde je GAN generátor použit pro generování různorodých denních–nočních párů. Vyhodnoceny jsou různé architektury generátorů, včetně nové GAN architektury, jež zachovává obsah mezi vzorovým a syntetizovaným obrázkem pomocí konzistence hran a současně trénuje detektor hran pro operování na denních i nočních obrázcích. V úloze fotografie–skica nejsou k dispozici žádné trénovací skicy, což je řešeno pomocí detektoru hran, který transformuje fotografii do hranového obrázku a následně je použit GAN generátor k ztenčení hran, čímž přiblíží takto vzniklý obrázek skicám. Navržený přístup rozšiřování datové sady překonává předchozí práce a současný state–of–the–art na benchmarku Tokio 24/7 pro vizuální vyhledávání ve dne i v noci, přičemž zachovává výkonnost benchmarků Oxford a Paris. Rozšíření datové sady založené na GAN však nepřekonává ruční rozšíření používané pro rozpoznávání skic.

Klíčová slova: Vizuální vyhledávání, Rozšíření datové sady, Generující adversariální sítě, Překlad obrazu

Překlad názvu: Překonání problému různých vizuálních domén ve vizuálním vyhledávání pomocí generování trénovacích dat

Contents

Figures

Tables

Chapter 1

Introduction

Given an image query depicting an object, the task of instance–level image retrieval is to find other images depicting the same object instance in a collection of images. In a globalized world, image collection can contain all images across the Earth which extends the task to large–scale instance–level image retrieval. For example, when a picture of a famous landmark such as the Orloj Astronomical tower is taken, one can upload this picture to an image retrieval system that finds other images of the uploaded landmark, see Figure 1.1. This is useful in real–world applications such as autonomous robots, autonomous cars, augmented reality (see Figure 1.2), *etc.* because these applications often need to perform localization in the real–world, where GPS and compass sensors are slow or imprecise, and high–tech sensors, such as Lidars, are too expensive to produce. Retrieved images can contain exact coordinates and other metadata, from which the system can calculate the precise position and orientation of the sender, and then, navigate the sender toward its target destination.

Figure 1.1: Examples of instance–level large–scale image retrieval. The input image (Query) is sent to an image retrieval system, which retrieves the image (Retrieved) containing the famous landmark instance from a large image database. Retrieval with night image or edge image queries is challenging for the image retrieval system.

When these applications rely mostly on camera sensors, the camera also captures changes in the outside environment. That means, input query images can be obtained under different illumination conditions such as day/night, at different year seasons such as summer/winter, or under different weather conditions such as sunny/rain. When considering these image pairs, the colors can be significantly shifted and textures can be weakened. In more difficult

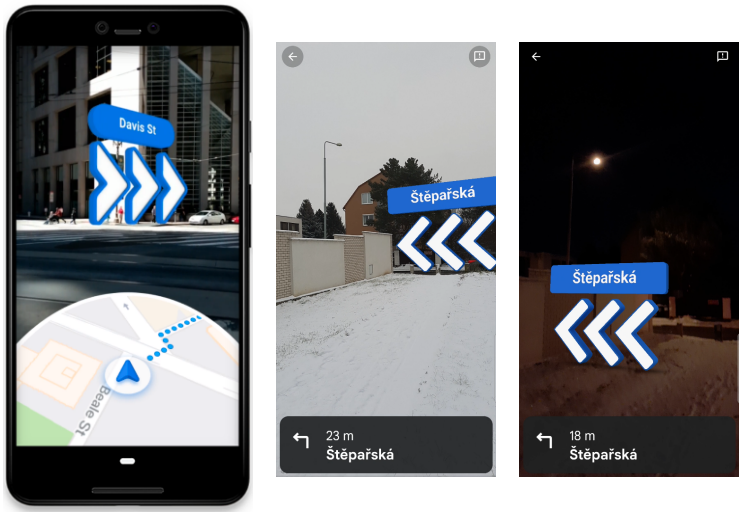


Figure 1.2: An application of instance–level image retrieval in augmented reality. The first image (left) illustrates the task of navigation using augmented reality.[1] The last two images (middle, right) are screenshots of Live View navigation from Google Maps used by myself in the same place but at different times. At night–time, placing the arrow in the correct spot is sometimes imprecise, suggesting that the dark image was not correctly synchronized with the map.

problems, such as with sketches or image outlines, the color and texture information may not be available at all. These conditions can significantly decrease the accuracy of the image retrieval system. This thesis only focuses on the two significant shifts – day/night and photo/sketch.

The reason why the image retrieval system can perform poorly on night images and sketches lies in its data–driven design. Image retrieval uses machine learning for image processing. Machine learning algorithms outperform other handcrafted algorithms and surpass humans in many tasks. The downside of machine learning is that training data must be leveraged to achieve higher performance. Specifically for image retrieval, training pairs of images that depict the same object are needed, which makes it even harder to obtain corresponding pairs of images taken during day and night or to obtain the corresponding photo and sketch. When those pairs are provided, the machine learning algorithm creates its internal representation structure from training examples, so that when the algorithm sees a new image, it has the ability to detect common patterns with the training images and generalize in the retrieval with image queries unseen to the algorithm during the training. To find patterns reliably, the algorithm needs a lot of training examples. The reason why the image retrieval system would generalize poorly on night images or sketches is that a significant amount of those images are missing in the training dataset. This is because people tend not to take many images during the night, see Figure 1.3, and it is difficult or almost impossible to gather a sketch of each object in the real–world. To learn the retrieval system

[1] Image taken from https://arvr.google.com

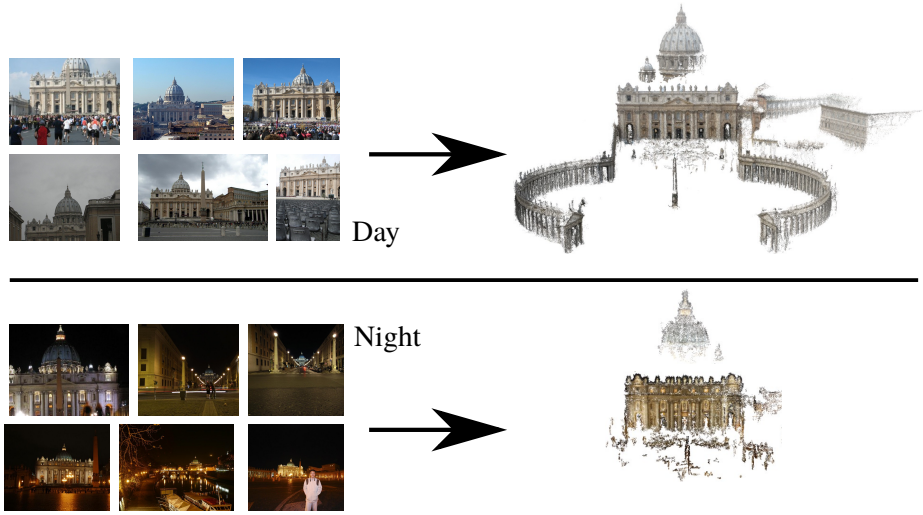


Figure 1.3: Day and night photos of a single famous landmark. Photos of St. Peter's Basilica in Rome were gathered online and split into day and night images. The day images are enough to cover the whole church, but night images are not enough. This is caused by (i) the day image count being 1980, while there are only 495 night images, and (ii) night images containing less information than day images, making the whole church reconstruction in the dark more challenging. All images in this figure are taken from [1].

to search accurately among the significant image shifts, the best training examples are pairs day–night or photo–sketch images, which are expensive and very time intensive to acquire.

To overcome this challenge, this thesis studies night image and sketch synthesis. For this purpose, a generative adversarial network (GAN) is employed, which is a machine learning algorithm that is learned to transform a day image into a night image or a photo into a sketch. By using GAN for image retrieval training, pairs of day–night and photo–sketch can be crafted from any other day or photo image in the training dataset, so that the image retrieval system becomes robust against these changes.

The rest of this thesis is structured as follows. In Chapter 2, basic terminology, definitions, and related work review are provided to help the reader understand the problem and possible solution concepts. In Chapter 3, a new lightweight GAN is introduced as well as the process of data augmentation in image retrieval. In Chapter 4, used datasets and technical details are covered. In Chapter 5, experiments are described and evaluated. In Chapter 6, the thesis is concluded.

Chapter 2

Background

2.1 Neural Networks

The goal of a **neural network** f is to approximate a function f^*, which maps an input $\boldsymbol{x}$ to an output $\boldsymbol{y}$, so that $\boldsymbol{y} = f^*(\boldsymbol{x})$, where the neural network f learns values of parameters $\boldsymbol{\theta}$ optimizing f for the best approximation of f^* for every training $\boldsymbol{x}$ and $\boldsymbol{y}$ [2]. In other words, the goal of neural network training is to adjust $\boldsymbol{\theta}$, so that $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$. For clarity, network parameters $\boldsymbol{\theta}$ are omitted in the rest of this thesis.

Neural networks are often **deep networks** because they are composed of multiple functions to allow approximation of more complex functions, for example, four functions $f^{(1)}$, $f^{(2)}$, $f^{(3)}$, and $f^{(4)}$ are connected into a chain forming $f(\boldsymbol{x}) = (f^{(1)} \circ f^{(2)} \circ f^{(3)} \circ f^{(4)})(\boldsymbol{x}) = f^{(4)}(f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x}))))$ meaning, $f^{(1)}$ is the first layer, $f^{(2)}$ is the second layer, and so on, where specifically the last layer is called **output layer** and the intermediate layers $f^{(2)}$, and $f^{(3)}$ are called **hidden layers** because training data does not provide desired output to any of these layers [2]. A deep network is **feedforward** when no recursion is present in the network function composition, when the deep network uses any recursion, it is called **recurrent** neural network. In this thesis, only feedforward neural networks are used.

2.1.1 CNN

Building a deep network as a multilayer perceptron (composed of fully–connected layers each followed by nonlinearities) has two flaws: the spatial information is ignored, and the network has a lot of trainable parameters [3]. Convolutional networks exploit the adjacency in the images with each output activation being dependent only on a small local region in the input, which makes the convolutional layer much more parameter–efficient [2].

The goal is to learn a **kernel**, which contains much smaller amounts of parameters (*e.g.* 3×3 for images) than the linear layer. In a **convolutional layer**, the outputs are obtained from a sliding window over the inputs, in which only elementwise multiplication with the kernel is performed, followed by the sum of the product. Formally, for an input $\boldsymbol{x} \in \mathbb{R}^{C^{(in)} \times H^{(in)} \times W^{(in)}}$, output $\boldsymbol{y} \in \mathbb{R}^{C^{(out)} \times H^{(out)} \times W^{(out)}}$, bias $\boldsymbol{b} \in \mathbb{R}^{C^{(out)} \times H^{(out)} \times W^{(out)}}$, and kernel $\boldsymbol{w} \in$

$\mathbb{R}^{C^{(out)} \times H^{(out)} \times W^{(out)}}$, the 2D convolution for position i can be implemented [4] as:[1]

$$\boldsymbol{y}_{C_i^{(out)}} = \boldsymbol{b}_{C_i^{(out)}} + \sum_{k=0}^{C^{(in)}-1} \boldsymbol{w}_{C_i^{(out)},k} \star \boldsymbol{x}_k, \tag{2.1}$$

where the convolution operation at measurement t is defined as $(\boldsymbol{w} \star \boldsymbol{x})_t = \sum_{\tau=-\infty}^{\infty} \boldsymbol{w}_{t-\tau} \boldsymbol{x}_\tau$[2]. The output $\boldsymbol{y}$ is often referred to as a **feature map**.

Examples of popular CNNs for the classification task are AlexNet [5], VGG [6], ResNet [7], Inception [8].

2.1.2 GAN

After the success of AlexNet [5], CNNs still struggled with approximating complex data distributions *e.g.* dog image distribution, because the goal of CNN training is to produce a discriminative model able only to recognize between training classes. The goal of GAN training is to produce a generative model able to estimate a probabilistic model. In 2014, Goodfellow *et al.* [9] introduced GAN training, which bridges this gap with discrimination between fake generated data and real data. The GAN training is composed of two networks: the generator G, and the discriminator D, which are trained jointly. The goal of the **generator** is to generate fake data $G(\boldsymbol{z})$ indistinguishable from $\boldsymbol{x}$ from random noise $\boldsymbol{z}$, while the goal of the **discriminator** is to classify, whether those data are real or fake, see Figure 2.1 [9]. G and D play the following zero–sum game:

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x} \sim P_X} \log D(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{z} \sim P_Z} \log(1 - D(G(\boldsymbol{z}))), \tag{2.2}$$

where P_X is training data distribution, P_Z is random noise distribution, and $\mathbb{E}_{\boldsymbol{x} \sim P_X}$ is the expectation of a function with respect to distribution P_X, which is the mean value the function takes on when $\boldsymbol{x}$ follows P_X [9]. The equilibrium of this game is when the generator produces data distribution indistinguishable from real data $p(\boldsymbol{x})$ and the discriminator always guesses at 0.5 confidence that the classified data are real or fake [9].

In practice, the optimization with gradient descent backpropagation is unstable and does not always reach the equilibria [10], because the objectives of the generator and discriminator are conflicting each other. The current main good practices are to use CNN–based discriminator and generator [11] and to modify the training algorithm of Goodfellow *et al.* [9] into this optimization:

$$\begin{aligned} &\max_D \mathbb{E}_{\boldsymbol{x} \sim P_X} \log D(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{z} \sim P_Z} \log(1 - D(G(\boldsymbol{z}))) \\ &\max_G \mathbb{E}_{\boldsymbol{z} \sim P_Z} \log D(G(\boldsymbol{z})). \end{aligned} \tag{2.3}$$

During one optimization step, there are two consecutive steps: one for the discriminator and then second for the generator [9]. In the discriminator step,

[1] Formulated in accordance to the PyTorch implementation documentation in https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html

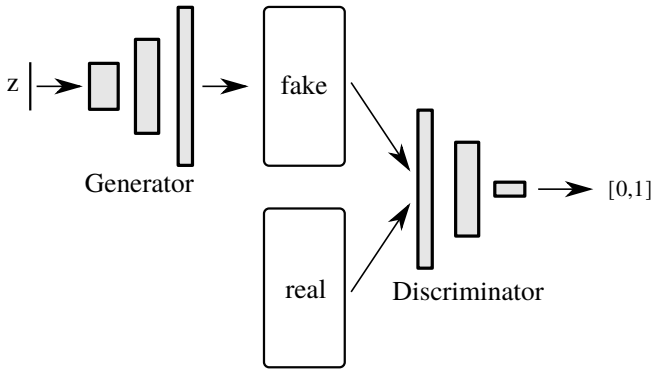


Figure 2.1: Optimization step of GAN training [9]. The Generator takes random noise z and produces fake data. The Discriminator predicts whether the fake data and input real data are real or fake.

the gradients for the discriminator are calculated with the fixed generator [9] and then the discriminator weights are updated by a small amount scaled down by a small learning rate. In the generator step, the same principle is used, and additionally, the generator loss function is modified from the theoretical $\log(1 - D(G(\boldsymbol{z})))$ minimization to the practical $\log(D(G(\boldsymbol{z})))$ maximization for a heuristic reasons [12].

LSGAN

In the generator optimization step, when the fake data are generated far away from the true data distribution, but on the correct side of the discriminator decision boundary, the generator weights do not update, because the loss function for the discriminator is binary cross–entropy, which leads to the vanishing gradients [13]. To deal with this issue, Mao *et al.* [13] proposed to remove the sigmoids from the architecture of the discriminator and to replace binary cross entropy loss with mean square error loss:

$$\begin{aligned} &\min_D \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim P_X}(D(\boldsymbol{x}) - b)^2 + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim P_Z}(D(G(\boldsymbol{z}) - a))^2 \\ &\min_G \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim P_Z}(D(G(\boldsymbol{z}) - c))^2, \end{aligned} \tag{2.4}$$

where a is the label for the fake data, b is the label for the real data, and c is the label that the generator wants the discriminator to believe for the fake data [13]. One possible choice of labels can be $a = 1, b = 0, c = 1$, which are the labels used in this work in the adversarial loss.

From my experiments, using LSGAN adversarial loss as in Equation 2.4 converges much faster than the original GAN loss as in Equation 2.2 for most of the models. Unfortunately, in the literature, the adversarial loss is dominantly expressed with Equation 2.2, although it is usually implemented as in Equation 2.3 or Equation 2.4. In this thesis, I decided that formalization should correspond to the implementation and express adversarial loss always as it is implemented in experiments.

cGAN

Sometimes, it would be desirable to synthesize a more specific output, conditioned on the specific input. This can be achieved when the generator and optionally the discriminator receive additional information $\boldsymbol{c}$, which can be *e.g.* label, vector, image, *etc.*, resulting in the extension of Equation 2.2:

$$\begin{aligned} &\max_D \mathbb{E}_{\boldsymbol{x}\sim P_X} \log D(\boldsymbol{x}, \boldsymbol{c}) + \mathbb{E}_{\boldsymbol{z}\sim P_Z} \log(1 - D(G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{c})) \\ &\max_G \mathbb{E}_{\boldsymbol{z}\sim P_Z} \log D(G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{c}). \end{aligned} \tag{2.5}$$

which is known as conditional GAN (cGAN) training [14].

In the following Sections, all GAN–based models are designed from the conditional GAN basis. Therefore, for brevity, the word *conditional* is omitted in the term *conditional GAN* in the rest of this thesis.

GAN Evaluation

Evaluation of GAN is difficult. Yet, no qualitative metric was defined because the generated image quality is subjective to define, and it can be specific per visual domain [15]. Moreover, each generator has its own discriminator which makes each generator optimize different criterion making quantitative evaluation also hard [16].

The first and widely popular metric for GAN quantitative evaluation which correlates well with human perception is Fréchet inception distance (FID) [17]. FID is designed to measure image quality, and diversity, and to detect mode collapse [17]. Briefly, FID of a generator is calculated with these steps:

1. feature maps of the real image and fake image distributions are extracted with Inception-V3 model [8] from the last pooling layer with 2048 activations,
2. assuming feature maps follow normal distribution, then feature mean $\boldsymbol{\mu} \in \mathbb{R}^{2048}$ and covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{2048\times2048}$ are calculated,
3. the final FID is calculated as Fréchet distance [18]:

$$\mathrm{FID}(\boldsymbol{x}, \boldsymbol{y}) = ||\boldsymbol{\mu}^x - \boldsymbol{\mu}^y||_2^2 + \mathrm{tr}(\boldsymbol{\Sigma}^x + \boldsymbol{\Sigma}^y - 2(\boldsymbol{\Sigma}^x\boldsymbol{\Sigma}^y)^{\frac{1}{2}}), \tag{2.6}$$

where $\boldsymbol{\mu}^x$ and $\boldsymbol{\mu}^y$ are the means of real image feature maps and fake image feature maps, respectively, $\boldsymbol{\Sigma}^x$ and $\boldsymbol{\Sigma}^y$ are covariance matrixes of real image feature maps and fake image feature maps, respectivelly, tr is the matrix trace [17]. FID is more generarly known as 2-Wasserstein distance [19].

Other popular quantitative GAN metrics are KID [20] (Kernel inception distance), or IS [16] (Inception score). Nevertheless, only FID [17] is used in this thesis for GAN evaluation because specifically IS is less precise than FID [17] and KID became popular more recently.

2.1.3 Initialization

When a deep network is trained from scratch, proper network weights initialization is one of the key elements in the learning with backpropagation. In

the early times, VGG [6] started the training with a moderate depth network and then more layers were progressively added. A similar procedure was adopted by [21, 22] for GAN training. However, such a process is not general for different architectures and requires a lot of manual tuning.

The problem with good initialization lies deeper in individual layer activation statistics. One of the simple weight initializations is initialization from a normal distribution $\mathbf{N}(0, 1)$. Suppose a single linear layer network $\boldsymbol{y} = f(\boldsymbol{x}) = \boldsymbol{W}^\top \boldsymbol{x}$ with input $\boldsymbol{x}$, weights $\boldsymbol{W}$, and activations $\boldsymbol{y}$. Although when the standard deviation of $\boldsymbol{x}$ is 1, the standard deviation of the activations $\boldsymbol{y}$ may not be necessarily 1, and activations in a deeper network would probably have a larger standard deviation if they would be initialized the same with $\mathbf{N}(0, 1)$, causing large covariance shifts [23], which results with gradient explosions or vanishing gradients [24, 2]. This issue can be simply solved by adjusting the initial standard deviation to a value, after which the covariances of deeper activations do not shift and the optimized loss function does not have extreme values and does not oscillate.

The most popular weight initialization nowadays is Kaiming initialization [25] (also known as He initialization), which is a simpler version of Xavier initialization [26]. The procedure of He [25] initializes each layer with a specific non–linearity–aware initialization and also scales covariances of each layer weights by the input size from the previous layer. With this initialization, the covariance shift is compensated with the non–linearity–specific gain. Another popular method for covariance shift prevention is batch normalization [27], which inserts normalization layers that transform the batch mean to zero and variance to one. For the GAN generator, instance normalization [28] further simplifies the image synthesis by preventing instance–specific covariance shifts. In contrast to batch normalization, instance normalization allows to remove contrast–specific information from the input image [28].

In the architectures introduced in this thesis, both normal and Kaiming [25] initializations as well as batch [27] and instance [28] normalizations are employed.

2.2 Transfer Learning

In terms of deep networks, **finetunning** (or transfer learning) of a network is network weights initialization with weights of a pretrained network for one task and training the finetuned network for another task [29]. With network finetunning, training the final model takes much less time, than when it would be trained from scratch, and also requires a lower amount of training data for the new task [29].

In this thesis, a CNN pretrained for the image classification task is finetuned to represent images in a metric space for the image retrieval task (described in the following Section 2.3.2).

2.2.1 Transfer Learning Notation

Besides learning for the new task, the task can remain the same and can be adapted to different attributes of data, such as day–night images or photo–sketch. More formally in accordance with [30, 31], let X and Y be the space of input, output examples, respectively, and let $\mathbf{x}$ and $\mathbf{y}$ be the random variables valued in X and Y, respectively. A domain $\mathcal{D}$ is defined as a ordered pair $\mathcal{D} = (\mathrm{X}, p(\mathbf{x}))$, and the task is to approximate the conditional probability distribution $p(\mathbf{y}|\mathbf{x})$ [30, 31].

In this thesis, two domains are considered, where the source domain is $\mathcal{D}^s = (\mathrm{X}, p(\mathbf{x}^s))$, the target domain is $\mathcal{D}^t = (\mathrm{X}, p(\mathbf{x}^t))$. For day–night image retrieval, the task is to learn an embedding $f^{D/N} : \mathrm{X} \to \mathrm{Y}$ that represents the set of images X as the set of descriptors Y, preceded by a generator $G : \mathrm{X} \to \mathrm{X}$ synthesizing from the day image distribution $p(\mathbf{x}^s)$ to fake night image distribution $p(G(\mathbf{x}^s))$ indistinguishable from true night image distribution $p(\mathbf{x}^t)$ helping the embedding to output the same descriptors for both input day images from the domain $\mathcal{D}^s$ and input night images from the domain $\mathcal{D}^t$. For sketch recognition, the task is to learn a classifier $f^{Sketch} : \mathrm{X} \to \mathrm{Y}$ that classifies a sketch $\mathbf{x}^t$ with a label y with the help of a transformation $T : \mathrm{X} \to \mathrm{X}$ that approximates $p(\mathbf{x}^t)$ with $p(T(\mathbf{x}^s))$, so that $p(\mathrm{y}|\mathbf{x}^t)$ can be approximated with $p(\mathrm{y}|T(\mathbf{x}^s))$ [32].

2.2.2 Domain Adaptation

Domain adaptation is a process that aims to train a network learned on the visual domain $\mathcal{D}^s$ to have the same performance on a new visual domain $\mathcal{D}^t$ with access to some examples in $\mathcal{D}^t$ [33].

The task of day–night image retrieval is enhanced with domain adaptation through image–to–image translation from the common day domain to the scarce night domain (described in the following Section 2.4). This is not the case for sketch recognition because no true sketch images are available during the classifier training [32].

2.2.3 Domain Generalization

Domain generalization is the process of training the model to perform well on a wider variety of visual domains without the need to finetune the model on each individual domain, so that the trained model can generalize well to new unseen visual domains during the inference time [34].

In this thesis, the task of sketch recognition is close to domain generalization with the common constraint that no sketches are available during the training and the difference that prior information about sketches to be classified is exploited [32].

2.3 Image Retrieval

Given a **query** and a collection (database) of images, image retrieval is the task of searching images relevant to the query. Depending on the type of image retrieval, we talk about **context–based image retrieval**, if the query is given as text, or the dataset is analyzed with text metadata. On contrary, we talk about **content–based image retrieval**, if only the content of the input query and database are analyzed, e.g. pixels, shapes, colors, etc. This thesis focuses only on content–based image retrieval.

According to the retrieval task, the relevance of retrieved results can be defined in different ways. Relevance examples are as follows:

- *Identical photo* – searching for images having identical parts with query image,
- **Instance–level** – searching for images depicting the specific object as the query image,
- *Identical class* – searching for images sharing the class (e.g. truck, dog, car, etc.) with query image,
- *Scenery* – retrieved images have the same structure as the input query but can depict different objects or places.

In this thesis, a retrieved image is relevant to the query, if that image depicts an identical object.

2.3.1 Retrieval with Local Features

In 2004, two breakthroughs allowed to perform content–based image retrieval on a larger scale [35]: the first is SIFT descriptor [36], which is a very popular compact vector representation of local image features *e.g.* corners, edges, *etc.* capturing image content invariant to scale changes and rotations; the second is Bag–of–Visual–Words (BoW) [37], where each image is represented as quantities of quantized SIFT descriptors.

Nowadays, the instance–level retrieval with BoW can be less precise because there exist more efficient learned local descriptors, such as HardNet [38], SOSNet [39], *etc.*, but their information would be lost in the descriptor quantization. VLAD [40] and ASMK [41] solves this by representing an image as an aggregation of descriptor embeddings, which can be used with shorter descriptors due to the additional dimension.

Image retrieval with local descriptors is not the topic of this thesis, but a brief overview has been provided to provide a clearer context.

2.3.2 Retrieval with CNNs

The first method that outperformed handcrafted image representation in instance image retrieval is NetVLAD [42], which mimics the local descriptor aggregation with the end–to–end learnable CNN. Later, NetVLAD was outperformed with a simpler representation by Radenovic *et al.* [43], which

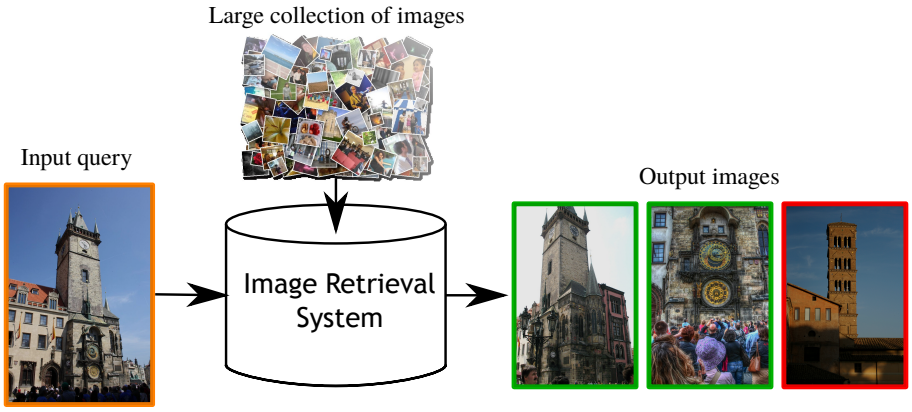


Figure 2.2: The task of image retrieval. Given an image query (orange), the goal of the image retrieval system is to find images relevant to the query image in a large collection of images. Relevant images (green) form positive pairs with the query image, while non–relevant images (red) form negative pairs with the query image.

uses CNN to learn a single global descriptor per image. The method of [43] is described in the rest of this Section.

CNN Architecture Modifications

To train the network to recognize object instances, a pretrained CNN (*e.g.* VGG [6], ResNet [7], *etc.*) on the simpler classification task is taken and finetuned for a metric learning task [43]. Architecture modifications occur at the very last layers of the CNN, where fully connected (linear) layers are replaced with a pooling layer and L2 normalization layer. In computer vision terminology, **backbone** is the neural network without the last discarded layers.

Defined in accordance with [43], the task of the **pooling layer** is to summarize and downsample feature map $\boldsymbol{z} \in \mathbb{R}^{H \times W \times D}$ obtained from the previous convolutional layer into a global descriptor $\boldsymbol{y} \in \mathbb{R}^{D}$, where D is the number of feature maps in the last convolutional layer, H and W is height and width of the feature map, respectively, and $\boldsymbol{z}_k$ denotes the set of $H \times W$ features for feature map $\boldsymbol{z}$ at position $k \in \{1, \ldots, D\}$. Assuming the nonlinearity after the last convolutional layer is ReLU, then each feature in $\boldsymbol{z}$ is non–negative. Simple examples of pooling layers are SPoC [44] (Equation 2.7) and MAC [45] (Equation 2.8):

$$\boldsymbol{y} = [y_1 \ldots y_k \ldots y_D]^\top, \text{where } y_k = \sum_{z \in \boldsymbol{z}_k} z, \tag{2.7}$$

$$\boldsymbol{y} = [y_1 \ldots y_k \ldots y_D]^\top, \text{where } y_k = \max_{z \in \boldsymbol{z}_k} z. \tag{2.8}$$

For SPoC and MAC, more generalized pooling was later proposed with

GeM [43] (Equation 2.9):

$$\boldsymbol{y} = [y_1 \dots y_k \dots y_D]^\top, \text{where } y_k = \left(\frac{1}{|\boldsymbol{z}_k|} \sum_{z \in \boldsymbol{z}_k} z^{p_k} \right)^{\frac{1}{p_k}}, \tag{2.9}$$

where $\boldsymbol{p} = [p_1 \dots p_k \dots p_D]^\top$ is the pooling parameter, learned with back-propagation. The pooling layer is followed by L2 normalization layer, which outputs descriptors L2–normalized allowing to compare image global descriptors simply with the inner product.

Siamese Learning and Loss Functions

After the network architecture is constructed for image embedding, the CNN backbone pretrained for the classification task together with an additional pooling layer is finetuned for the ranking problem (in the case of the image retrieval task, ranking list of images by the relevance to the query image input). By being able to rank images, retrieved images of the same object instance can be further ranked by their similarity to the query.

To incorporate the ranking into the training, the loss function needs to compare images with other images, rather than only the correctness of each single image label, value, or set of values. The straightforward way to achieve this is to rank similar images by their embedding distance to the query. This desired task is for the embedding training called **metric learning**, i.e. learning the relative distance between inputs. When the metric is learned, three types of images can be selected:

- **anchor** – the reference image, which is the query,
- **positive** – the image which should have a short distance to the anchor,
- **negative** – the image which should have a long distance to the anchor.

In the case of the image retrieval task, positives are images relevant to the anchor, while negatives are images not relevant to the anchor. There are two main loss functions for the metric learning task: contrastive loss [46] (Equation 2.12) and triplet loss [47] (Equation 2.11). Let $\boldsymbol{y}$, $\boldsymbol{y}^+$, $\boldsymbol{y}^-$ be the descriptor for anchor, positive, negative, respectively. Let $d : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ be a distance function, e.g. euclidian distance: $d(\boldsymbol{u}, \boldsymbol{v}) = ||\boldsymbol{u} - \boldsymbol{v}||_2^2$ for $\boldsymbol{u} \in \mathbb{R}^D$, $\boldsymbol{v} \in \mathbb{R}^D$. Then, the loss functions are defined as follows:

$$l(\boldsymbol{y}, \boldsymbol{y}^+, \boldsymbol{y}^-) = d(\boldsymbol{y}, \boldsymbol{y}^+) + \max(0, m - d(\boldsymbol{y}, \boldsymbol{y}^-)) \tag{2.10}$$

$$l(\boldsymbol{y}, \boldsymbol{y}^+, \boldsymbol{y}^-) = \max(0, m + d(\boldsymbol{y}, \boldsymbol{y}^+) - d(\boldsymbol{y}, \boldsymbol{y}^-)), \tag{2.11}$$

where $m > 0$ is margin hyperparameter. Sometimes, the contrastive loss obtains N negatives $\boldsymbol{y}^- \in \mathbb{R}^{N \times D}$, which modifies the Equation 2.12 to:

$$l(\boldsymbol{y}, \boldsymbol{y}^+, \boldsymbol{y}^-) = d(\boldsymbol{y}, \boldsymbol{y}^+) + \sum_{n=1}^{N} \max(0, m - d(\boldsymbol{y}, \boldsymbol{y}_n^-)). \tag{2.12}$$

Neural network, which learns any of these losses is called **siamese network**, since it processes different input images (anchor, positive and negative), while

the forward pass shares weights for each input. As a result, $\boldsymbol{y}$, $\boldsymbol{y}^+$, $\boldsymbol{y}^-$ are obtained from the same network. In this thesis, only contrastive loss is utilized in metric learning.

Training Data Mining

Usually, there are no ground–truth pair annotations within the training dataset that would directly describe which image is positive or negative to each anchor. Moreover, for n training images, there are $\binom{n}{2}$ pairs making it difficult to include them all in the siamese training. On top of that, most of these pairs are negative, and therefore random pair sampling is very likely to pick negative pair having its descriptors already far enough resulting in zero gradient of the loss function [48].

When there are class labels per image (*e.g.* landmark, building, *etc.*), it is not much useful since matching class labels do not necessarily match with the identical object because different instances of objects can exist within the same class (*e.g.* Orloj Astronomical Tower and Eifell Tower share the same landmark class, but they are different instances).

With GPS information, it is possible to find enough negatives and utilize the information as weak labeling for images [42]. However, images geographically close together are not necessarily positives, since their camera orientation is unknown.

In this thesis, the image retrieval baseline of Radenovic et al. [43] overcomes these issues with automatic label mining from SfM 3D reconstruction, which offers camera position and orientation per image, as well as image clusters of landmarks [1]. In the baseline [43], positives are selected at random from a set of images that shares enough points with the query but do not display too extreme scale change, while negatives are selected as **hard negatives** [48], i.e. negative pair from different clusters with the most similar descriptor.

2.3.3 Retrieval Evaluation

Starting with a simple case, consider only one query q. Then, for q, there are samples selected by the search engine, where some of them can be relevant and others non–relevant. There are four possible cases for each classified retrieved sample. Let

- $\mathrm{tp}(q)$ be the number of selected samples, which are relevant,
- $\mathrm{fp}(q)$ be the number of selected samples, which are not relevant,
- $\mathrm{tn}(q)$ be the number of not–selected samples, which are not relevant,
- $\mathrm{fn}(q)$ be the number of not–selected samples, which are relevant.

Two common measures – precision $p \in [0, 1]$ and recall $r \in [0, 1]$ – are defined:

$$\begin{aligned} p(q) &= \frac{\mathrm{tp}(q)}{\mathrm{tp}(q) + \mathrm{fp}(q)}, \\ r(q) &= \frac{\mathrm{tp}(q)}{\mathrm{tp}(q) + \mathrm{fn}(q)}. \end{aligned} \tag{2.13}$$

Unpacking these scores a bit further, precision indicates how many relevant samples were selected from all selected samples, whereas recall indicates how many relevant samples were selected from all relevant samples.

The order of the selected samples is also relevant. Precision and recall do not measure the rank of selected samples. However, they can be utilized to calculate a single measure capturing ranks of retrieved samples. Consider $p(r(q))$ as precision function of recall. Computing the precision of recall at each rank of retrieved samples results in a precision–recall curve. Average precision AP is the area under the precision–recall curve:

$$AP(q) = \int_0^1 p(r(q))dr. \tag{2.14}$$

To reach the best average precision, all selected samples must be relevant, and therefore image retrieval model obtains the precision of 1 per each recall level [49].

In practice, the search is relevant only up to the top k retrieved samples. Then, average precision at k can be defined as:

$$AP@k(q) = \frac{1}{\min(k, \mathrm{tp}(q))} \sum_{i=1}^{k} \left[\frac{1}{i} \mathrm{rel}(q,i) \sum_{j=1}^{i} \mathrm{rel}(q,j) \right], \tag{2.15}$$

where $\mathrm{rel}(q,i) = 1$ iff i–th retrieved sample with the query q is relevant, otherwise $\mathrm{rel}(q,i) = 0$ [50].

To evaluate a retrieval model more precisely, it is desirable to measure $AP@k$ with more than one query. Then, the most straightforward way of computing more average precision with multiple queries is to calculate the **mean average precision** mAP across the set of all evaluation queries Q:

$$mAP@k(Q) = \frac{1}{|Q|} \sum_{q \in Q} AP@k(q). \tag{2.16}$$

In this thesis, only mAP is utilized as the benchmark for image retrieval performance.

2.3.4 Image Retrieval Challenges

Unexpected conditions can occur during image retrieval training or during inference causing a significant retrieval performance drop. Specifically, retrieved images can be very different from the input query, but they can be relevant, and vice–versa. The list of possible image retrieval challenges, where differences tacked in this thesis are **bold**, is as follows:

- *Scale and/or viewpoint change.* The input image can be captured zoomed-in or zoomed-out to most of the images in the image database.
- *Occlusion.* An object in the input image blocks the view of the retrieved object.
- *Different visually similar objects.* Images with very similar structures, but depicting different objects, e.g. Arch of Titus in Rome/Arc de Triomphe in Paris.

- **Illumination change**. Image has different illumination conditions, e.g. sunny, lighting, etc., and/or time conditions, for example, day/night, etc. [51]
- **Different visual domain**. More generally to illumination changes, images can have different textures and/or colors, e.g. sketches, art paintings, etc. [52]
- *Image Retrieval with Big Data*. Image database contains large volumes and varieties of data, that are required to be retrieved at high–velocity, which active area of research especially in remote sensing [53].
- *Adversarial examples*. Model integrity can be violated with a carefully crafted adversarial noise invisible to the human added to the image can cause the network to misclassify the input [54]. In image retrieval, [55] performed model evasion using a concealed query that obtained targeted retrieval results without disclosing any information about the target query image.

2.4 Image–to–Image Translation

When the image input visual domain significantly differs from training images, the image retrieval system generalizes poorly. The straightforward solution to this problem is to add these adversarial inputs to the retrieval training dataset to increase training data diversity and image retrieval robustness. However, adding this kind of data is not enough, because these adversarial inputs may not be relevant to most images in the database. This can be tackled with cGAN [14] (covered in the previous Section 2.1.2), which can synthesize images based on the input image that could be positive.

Image–to–image translation (image translation) is the task of transforming an image in one visual domain into an image in the target visual domain. The output image can have different textures, colors, and/or styles, but the image content structure is preserved from the input image.

In terms of machine learning, **unsupervised image–to–image translation** (also known as *unpaired image–to–image translation*) is the task of image translation without ground–truth image pairs, while in **supervised image–to–image translation** (also known as *paired image–to–image translation*) each training image has its corresponding image in the target domain. In practice, however, high–quality image pairs are difficult to obtain.

2.4.1 Image Translation Notation

Let X be the set of source domain images in RGB colorspace, where image $\boldsymbol{x}$ is from the source domain if $\boldsymbol{x} \in X$. Let P_X denote the data distribution of the source domain images. Image $\boldsymbol{x}$ follows distribution P_X, when $\boldsymbol{x} \sim P_X$. Target domain Y, target image $\boldsymbol{y}$, and target domain data distribution P_Y are defined similarly.

The goal of GAN training is to train a generator network $G : X \to Y$, which translates image $\boldsymbol{x}$ in the source visual domain into corresponding $\boldsymbol{y}$ in

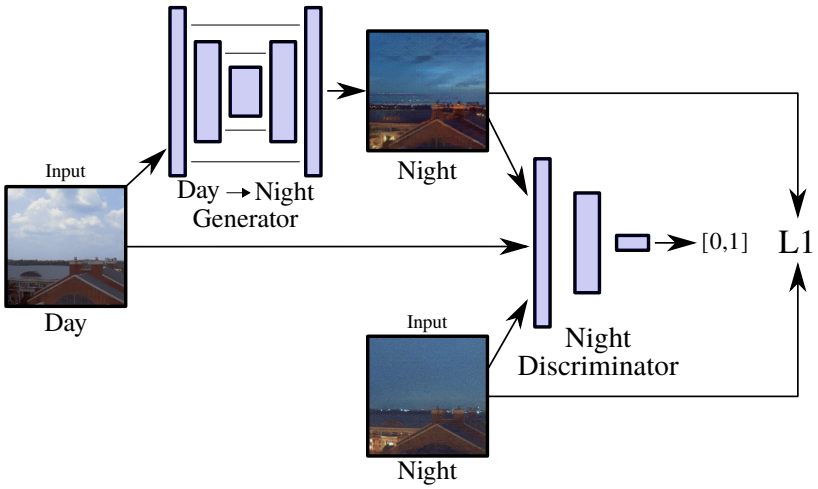


Figure 2.3: Optimization step of the pix2pix training [56]. The Day→Night Generator takes an input day image and translates it into a fake night image. After this forward pass, structure consistency is ensured with L1 loss minimalization. The Night Discriminator then determines whether the translated image and the input night image are real or fake with the input day image condition.

the target visual domain preserving the image content. The discriminator $D : Y \rightarrow [0, 1]$ is trained jointly with the generator and tries to classify, whether the target domain image $\boldsymbol{y}$ is real or fake. In paired image translation, the discriminator can also receive the image from the source domain and $D : Y \times X \rightarrow [0, 1]$.

2.4.2 pix2pix

In pix2pix training, the generator translates the input image into the output fake image in the target domain, while the discriminator tries to distinguish if the ground–truth image and generated image are fake in the target domain [56]. In addition, in order GAN training does not fall into mode collapse, L1 loss regularization between generated image and the ground–truth image is applied in pixel space, forcing the content of the fake image to be identical with the ground–truth image [56].

The pix2pix optimization is similar to the cGAN [14] formulation (covered in the Section 2.1.2) with two differences:

- no noise for the generator input is no more needed because the generator would learn to ignore it and uses only the information provided from the image condition,
- regularization is employed with L1 loss between real and fake images in the target domain [56].

These modifications can be formulated as a pix2pix optimization of a generator G, and discriminator D following the Park *et al.* [56] formulation altered

with LSGAN [13] adversarial loss (covered in the previous Section 2.1.2):

$$\begin{aligned} &\min_D \mathcal{L}_{adv}^{(D)}(D, G, X, Y) \\ &\min_G \mathcal{L}_{adv}^{(G)}(D, G, X) + \lambda \mathcal{L}_{reg}(G, X, Y), \end{aligned} \tag{2.17}$$

where the adversial loss for discriminator $\mathcal{L}_{adv}^{(D)}$, and the adversial loss generator $\mathcal{L}_{adv}^{(G)}$ are:

$$\begin{aligned} \mathcal{L}_{adv}^{(D)}(D, G, X, Y) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim P_X}[(D(G(\boldsymbol{x}), \boldsymbol{x}))^2] + \frac{1}{2}\mathbb{E}_{\boldsymbol{y}\sim P_Y}[(D(\boldsymbol{y}, \boldsymbol{x}) - 1)^2] \\ \mathcal{L}_{adv}^{(G)}(D, G, X) &= \mathbb{E}_{\boldsymbol{x}\sim P_X}[(D(G(\boldsymbol{x}), \boldsymbol{x}) - 1)^2]. \end{aligned} \tag{2.18}$$

The regularization $\mathcal{L}_{reg}$ for the generator is defined as:

$$\mathcal{L}_{reg}(G, X, Y) = \mathbb{E}_{\boldsymbol{x}\sim P_X, \boldsymbol{y}\sim P_Y}[||\boldsymbol{y} - G(\boldsymbol{x})||_1] \tag{2.19}$$

The relative importance of regularization can be controlled with λ in Equation 2.17, which is a new hyperparameter needed to be tuned. The default setting of this weight for the vanilla pix2pix is $\lambda = 100$ [56]. This optimization is displayed in Figure 2.3.

2.4.3 CycleGAN

CycleGAN is the first popular model able to perform unpaired image translation, which is based on pix2pix [56]. However, in contrast with pix2pix training, no ground–truth image nor image structure binding is available, and no conditional information is provided, hence the generator is likely to output any image in the target domain or to fall into mode collapse [57]. CycleGAN tackles this issue with **cycle consistency**, where the fake image in the target domain is translated back into the fake image in the source domain with a second generator, and finally, the reconstructed image is forced to be identical to the input image with L1 loss regularization.

The CycleGAN optimization is similar to the pix2pix optimization with three main differences:

- regularization is employed with cycle consistency [57],
- a second generator is learned the same way as the first to allow the cycle consistency to be used,
- the discriminator is no longer conditioned to the input because structures of the real and fake images do not correspond.

This results in an optimization for generators G_X, G_Y and discriminators D_X, D_Y formulated similarly to Zhu *et al.* [57] as follows:

$$\begin{aligned} &\min_{D_X, D_Y} \mathcal{L}_{adv}^{(D)}(D_Y, G_X, X, Y) + \mathcal{L}_{adv}^{(D)}(D_X, G_Y, Y, X) \\ &\min_{G_X, G_Y} \mathcal{L}_{adv}^{(G)}(D_Y, G_X, Y) + \mathcal{L}_{adv}^{(G)}(D_X, G_Y, X) + \lambda \mathcal{L}_{cyc}(G_X, G_Y, X, Y), \end{aligned} \tag{2.20}$$

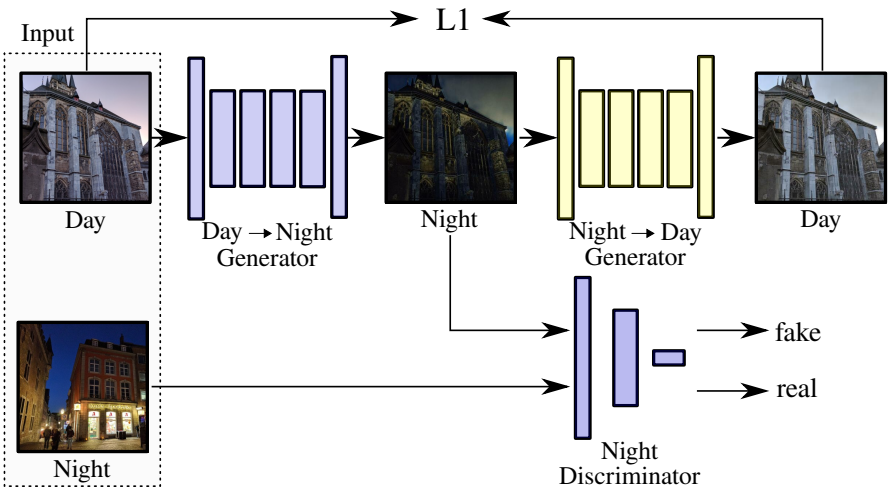


Figure 2.4: Optimization step of the CycleGAN training [57]. The Day→Night Generator takes an input day image and translates it into a fake night image, and then, Night→Day Generator takes the fake night image and translates it back into a reconstructed fake day image. After these two passes, cycle consistency is ensured with L1 loss minimalization. The Night Discriminator then determines whether the intermediate translated image and the input night image are real or fake. Only one direction of the CycleGAN training is displayed.

where the adversial loss for discriminator $\mathcal{L}_{adv}^{(D)}$, and the adversial loss generator $\mathcal{L}_{adv}^{(G)}$ in $X \to Y$ translation are:

$$\begin{aligned}\mathcal{L}_{adv}^{(D)}(D_Y, G_X, X, Y) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim P_X}[(D_Y(G_X(\boldsymbol{x})))^2] + \frac{1}{2}\mathbb{E}_{\boldsymbol{y}\sim P_Y}[(D_Y(\boldsymbol{y})-1)^2] \\ \mathcal{L}_{adv}^{(G)}(D_Y, G_X, X) &= \mathbb{E}_{\boldsymbol{x}\sim P_X}[(D_Y(G_X(\boldsymbol{x}))-1)^2].\end{aligned} \tag{2.21}$$

For the discriminator D_X, and the generator G_Y in $Y \to X$ translation the adversarial loss is defined similarly. The cycle consistency loss for both generators is:

$$\begin{aligned}\mathcal{L}_{cyc}(G_X, G_Y, X, Y) = \mathbb{E}_{\boldsymbol{x}\sim P_X}[||\boldsymbol{x} - G_Y(G_X(\boldsymbol{x}))||_1] \\ + \mathbb{E}_{\boldsymbol{y}\sim P_Y}[||\boldsymbol{y} - G_X(G_Y(\boldsymbol{y}))||_1].\end{aligned} \tag{2.22}$$

The relative importance of cycle consistency can be controlled with λ in Equation 2.20, similarly to pix2pix. In [57] and for all experiments conducted in this thesis, $\lambda = 10$. This optimization is displayed in Figure 2.4.

2.4.4 DRIT

While CycleGAN [57] is optimized for unpaired image translation, where no corresponding training image pairs are not available, DRIT [58, 59] is additionally optimized for **multimodal unpaired image translation**, which is the task of translating an image from the input domain into a distribution of potential images in the target domain (while in unimodal image translation, the translation happens into a single target image).

The straightforward solution to synthesize multimodal outputs is to inject a noise vector next to the generator input and prevent the subsequent mode collapse [9, 60]. The first model that achieved this is BicycleGAN [61], which adds an encoder optimized to output the latent code to be invertible. However, BicycleGAN only performs multimodal paired image translation (it needs pixel–aligned pairs of images in the source and target visual domain).

Another solution for multimodal image translation can be achieved with disentangled representation, where the image features are broken down into two parts: a **content** shared between the source and target images, and a **style** which is different for the source and target images. This is the main principle of how the DRIT [58, 59] architecture is designed. Similarly to the BicycleGAN [61], more additional encoders are trained, where the **content encoder** E^c is optimized to extract the latent code that is forced to be preserved in the $X \to Y$ translation, and the **style encoder** E^s is optimized to extract the latent code that is forced to be preserved among the real and fake images in their corresponding domain. For the translation between two domains, let S_X, S_Y, and C be the latent domain of X, the latent domain of Y, and content–specific latent domain, respectively. Then, the following networks are trained:

- content encoders $E^c_X : X \to C$, and $E^c_Y : X \to C$ to map images into shared content space,
- style encoders $E^s_X : X \to S_X$, and $E^s_Y : Y \to S_Y$ to map images into style–specific space,
- generators $G_X : C \times S_X \to X$, and $G_Y : C \times S_Y \to Y$ to synthesize fake images conditioned on both content and style vectors,
- discriminators $D_X : X \to [0, 1]$, and $D_Y : Y \to [0, 1]$ to classify whether images in their corresponding domain are real or fake,
- content discriminator $D^c : C \to [0, 1]$ to distinguish the extracted content between domains X and Y,

resulting in 9 networks being optimized in total [58, 59].

Whenever an image $\boldsymbol{x} \in X$ is being translated into $\hat{\boldsymbol{y}}$, E^c_X extracts its content latent code $\boldsymbol{z}^c_x$ and then the image $\hat{\boldsymbol{y}}$ is synthesized with G_Y as $\hat{\boldsymbol{y}} = G_Y(\boldsymbol{z}^c_x, \boldsymbol{z}^s_y)$, where $\boldsymbol{z}^s_y$ can be sampled from normal distribution $\mathbf{N}(0, 1)$, or extracted from image $\boldsymbol{y} \in Y$ with E_Y. To achieve this, the following 7 losses are used in the final optimization:

Content Adversarial Loss. Assuming domains X, and Y share common latent space [62], in the DRIT architecture [58] content encoders E^c_X and E^c_Y also share the last layer and the generators G_X and G_Y share the first layer. By sharing these weights, it is enforced that the image content is mapped into the same content space [58]. To assure the same content information is encoded with both content encoders E^c_X and E^c_Y, the content discriminator

D^c tries to distinguish from which domain were $\boldsymbol{z}_x^c$ and $\boldsymbol{z}_y^c$ encoded:

$$\begin{aligned}\mathcal{L}_{c-adv}^{(D)}(D^c, E_X^c, E_Y^c, X, Y) = &- \mathbb{E}_{\boldsymbol{x}\sim P_X}[\log D^c(E_X^c(\boldsymbol{x}))] \\ &- \mathbb{E}_{\boldsymbol{y}\sim P_Y}[\log(1 - D^c(E_Y^c(\boldsymbol{y})))] \\ \mathcal{L}_{c-adv}^{(G)}(D^c, E_X^c, X) = &- \mathbb{E}_{\boldsymbol{x}\sim P_X}\left[\frac{1}{2}\log D^c(E_X^c(\boldsymbol{x})) + \frac{1}{2}\log(1 - D^c(E_X^c(\boldsymbol{x})))\right],\end{aligned} \tag{2.23}$$

The loss $\mathcal{L}_{c-adv}^{(G)}(D^c, E_Y^c, Y)$ is defined similarly.

Cross–Cycle Consistency Loss. Similarly to unimodal image translation, to preserve the content of the source and target image, content consistency is preserved by reconstructing the synthesized images back to their original domain as in CycleGAN [57], but DRIT extends the cycle consistency with disentangled content and swapping the domain–specific style latent codes [58]:

$$\begin{aligned}\mathcal{L}_{ccc}(E_X^c, E_Y^c, E_X^s, E_Y^s, G_X, G_Y, X, Y) = \mathbb{E}_{\boldsymbol{x}\sim P_X, \boldsymbol{y}\sim P_Y}[&||\boldsymbol{x} - \hat{\hat{\boldsymbol{x}}}||_1 \\ &+ ||\boldsymbol{y} - \hat{\hat{\boldsymbol{y}}}||_1],\end{aligned} \tag{2.24}$$

where the reconstructed images are obtained as:

$$\begin{aligned}\hat{\hat{\boldsymbol{x}}} &= G_X(E_Y^c(\hat{\boldsymbol{y}}), E_X^s(\hat{\boldsymbol{x}})), \\ \hat{\hat{\boldsymbol{y}}} &= G_Y(E_X^c(\hat{\boldsymbol{x}}), E_Y^s(\hat{\boldsymbol{y}})),\end{aligned} \tag{2.25}$$

and the fake images are obtained in the same way as:

$$\begin{aligned}\hat{\boldsymbol{x}} &= G_X(E_Y^c(\boldsymbol{y}), E_X^s(\boldsymbol{x})), \\ \hat{\boldsymbol{y}} &= G_Y(E_X^c(\boldsymbol{x}), E_Y^s(\boldsymbol{y})).\end{aligned} \tag{2.26}$$

Domain Adversarial Loss. To make synthesized images indistinguishable from the real images, D_X and D_Y attempt to classify, whether an image is real or fake [58]. In the same way, as in CycleGAN [57], the encoders and generators are optimized to synthesize fake images indistinguishable from the real images:

$$\begin{aligned}\mathcal{L}_{d-adv}^{(D)}(D_Y, E_X^c, E_Y^s, G_Y, X, Y) &= -\mathbb{E}_{\boldsymbol{y}\sim P_Y}\log(D_Y(\boldsymbol{y})) - \mathbb{E}_{\boldsymbol{x}\sim P_X}\log(1 - D_Y(\hat{\boldsymbol{y}})) \\ \mathcal{L}_{d-adv}^{(G)}(D_Y, E_X^c, E_Y^s, G_Y, X) &= -\mathbb{E}_{\boldsymbol{x}\sim P_X}\log(D_Y(\hat{\boldsymbol{y}})),\end{aligned} \tag{2.27}$$

where $\hat{\boldsymbol{y}}$ is fake image of domain Y, obtained as in Equation 2.26 [58]. The losses $\mathcal{L}_{d-adv}^{(D)}(D_X, E_Y^c, E_X^s, G_X, Y, X)$ and $\mathcal{L}_{d-adv}^{(G)}(D_X, E_Y^c, E_X^s, G_X, Y)$ are defined similarly.

Self–reconstruction Loss. Similarly to the cross–cycle consistency loss, latent features encoded with style and content encoders should be able to be decoded back with their corresponding generator in self–reconstruction loss as [58]:

$$\mathcal{L}_{rec}(E_X^c, E_X^s, G_X, X) = \mathbb{E}_{\boldsymbol{x}\sim P_X}[||\boldsymbol{x} - G_X(E_X^c(\boldsymbol{x}), E_X^s(\boldsymbol{x}))||_1]. \tag{2.28}$$

The loss $\mathcal{L}_{rec}(E_Y^c, E_Y^s, G_Y, Y)$ is defined similarly.

KL Loss. To perform stochastic sampling at inference time, DRIT [58] used KL loss to style latent code and normal distribution $\mathbf{N}(0,1)$. This has been later replaced with latent vector L2 regularization in DRIT++ [59][2]:

$$\mathcal{L}_{KL}(E_X^c, E_X^s, X) = \mathbb{E}_{\boldsymbol{x}\sim P_X}[||E_X^c(\boldsymbol{x})||_2^2 + ||E_X^s(\boldsymbol{x})||_2^2]. \tag{2.29}$$

The loss $\mathcal{L}_{KL}(E_Y^c, E_Y^s, Y)$ is defined similarly.

Latent Regression Loss. Similarly to BicycleGAN [61], [58] encourages the latent code to be invertible with the image. A random vector $\boldsymbol{z}$ for style is drawn from the normal distribution $\mathbf{N}(0,1)$ and reconstructed with the style encoder in latent regression loss as [58]:

$$\mathcal{L}_{lat}(E_X^c, E_X^s, G_X, X) = \mathbb{E}_{\boldsymbol{x}\sim P_X, \boldsymbol{z}\sim\mathbf{N}(0,1)}[||\boldsymbol{z} - E_X^s(G_X(E_X^c(\boldsymbol{x}), \boldsymbol{z}))||_1]. \tag{2.30}$$

The loss $\mathcal{L}_{lat}(E_Y^c, E_Y^s, G_Y, Y)$ is defined similarly.

Mode–seeking Loss. The first DRIT [58] suffers from mode collapse [59] meaning, the style diversity of synthesized images diversity was low. The architecture of DRIT++ [59] mitigates the mode collapse with mode–seeking regularization that was first employed in MSGAN [63]. Given two latent vectors $\boldsymbol{z}_1 \in S_Y$ and $\boldsymbol{z}_2 \in S_Y$, the regularization is achieved by maximizing the ratio of the distance between synthesized images $\hat{\boldsymbol{y}}_1$ and $\hat{\boldsymbol{y}}_2$ and with respect to the distance between $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$:

$$\max_{E_X^c, G_Y} \frac{d_I(\hat{\boldsymbol{y}}_1, \hat{\boldsymbol{y}}_2)}{d_z(\boldsymbol{z}_1, \boldsymbol{z}_2)} = \max_{E_X^c, G_Y} \frac{d_I(G_Y(E_X^c(\boldsymbol{x}), \boldsymbol{z}_1), G_Y(E_X^c(\boldsymbol{x}), \boldsymbol{z}_2))}{d_z(\boldsymbol{z}_1, \boldsymbol{z}_2)}, \tag{2.31}$$

where $d_I(\cdot,\cdot)$, and $d_z(\cdot,\cdot)$ are the distance metrics for image, and latent vector space, respectively [63, 59]. In practice, the mode seeking loss is implemented as an inverse ratio between L1 distances to correspond with the total loss minimization [63]:

$$\mathcal{L}_{ms}(E_X^c, G_Y, X) = \mathbb{E}_{\boldsymbol{x}\sim P_X, (\boldsymbol{z}_1,\boldsymbol{z}_2)\sim\mathbf{N}(0,1)} \frac{||\boldsymbol{z}_1 - \boldsymbol{z}_2||_1}{||G_Y(E_X^c(\boldsymbol{x}), \boldsymbol{z}_1) - G_Y(E_X^c(\boldsymbol{x}), \boldsymbol{z}_2)||_1}. \tag{2.32}$$

The loss $\mathcal{L}_{ms}(E_Y^c, G_X, Y)$ is defined similarly.

Finally, the DRIT++[59] optimization can be expressed as:

$$\begin{aligned} &\min_{D_X, D_Y, D^c} \mathcal{L}^{(D)}(D_X, D_Y, D^c, G_X, G_Y, X, Y) \\ &\min_{E_X^c, E_Y^c, E_X^s, E_Y^s, G_X, G_Y} \mathcal{L}^{(G)}(E_X^c, E_Y^s, E_X^s, E_Y^s, G_X, G_Y, X, Y), \end{aligned} \tag{2.33}$$

where the discriminators minimize:

$$\begin{aligned} \mathcal{L}^{(D)}(D_X, D_Y, D^c, G_X, G_Y, X, Y) =& \lambda_{c-adv}\mathcal{L}_{c-adv}^{(D)}(D^c, E_X^c, E_Y^c, X, Y) \\ &+\lambda_{d-adv}\mathcal{L}_{adv}^{(D)}(D_Y, E_X^c, E_Y^s, G_Y, X, Y) \\ &+\lambda_{d-adv}\mathcal{L}_{adv}^{(D)}(D_X, E_Y^c, E_X^s, G_X, Y, X), \end{aligned} \tag{2.34}$$

[2] Authors confirmed, that removing KL loss does not degrade the final performance at https://github.com/HsinYingLee/DRIT/issues/37.

and the encoders and generators minimize:

$$\begin{aligned}
&\mathcal{L}^{(G)}(D_X, D_Y, D^c, E_X^c, E_Y^c, E_X^s, E_Y^s, G_X, G_Y, X, Y) = \\
&\quad \lambda_{c-adv}\mathcal{L}_{c-adv}^{(G)}(D^c, E_X^c, X) + \lambda_{c-adv}\mathcal{L}_{c-adv}^{(G)}(D^c, E_Y^c, Y) \\
&+\lambda_{ccc}\mathcal{L}_{ccc}(E_X^c, E_Y^c, E_X^s, E_Y^s, G_X, G_Y, X, Y) \\
&+\lambda_{d-adv}\mathcal{L}_{d-adv}^{(G)}(D_Y, E_X^c, E_Y^s, G_Y, X) + \lambda_{d-adv}\mathcal{L}_{d-adv}^{(G)}(D_X, E_Y^c, E_X^s, G_X, Y) \\
&+\lambda_{rec}\mathcal{L}_{rec}(E_X^c, E_X^s, G_X, X) + \lambda_{rec}\mathcal{L}_{rec}(E_Y^c, E_Y^s, G_Y, Y) \\
&+\lambda_{KL}\mathcal{L}_{KL}(E_X^c, E_X^s, X) + \lambda_{KL}\mathcal{L}_{KL}(E_Y^c, E_Y^s, Y) \\
&+\lambda_{lat}\mathcal{L}_{lat}(E_X^c, E_X^s, G_X, X) + \lambda_{lat}\mathcal{L}_{lat}(E_Y^c, E_Y^s, G_Y, Y) \\
&+\lambda_{ms}\mathcal{L}_{ms}(E_X^c, G_Y, X) + \lambda_{ms}\mathcal{L}_{ms}(E_Y^c, G_X, Y),
\end{aligned} \tag{2.35}$$

where in [59] the default weights controlling the relative importance of each loss are $\lambda_{c-adv} = 1$, $\lambda_{ccc} = 10$ $\lambda_{d-adv} = 1$, $\lambda_{rec} = 10$, $\lambda_{KL} = 0.01$, $\lambda_{lat} = 10$, and $\lambda_{ms} = 1$.

2.4.5 CUT

In the task of unpaired image translation, the content of the input image and the translated images is typically ensured with cycle–consistency used by for example CycleGAN (Section 2.4.3), and other architectures [64, 65]. Cycle–consistency assumes there is a bijection between the source and the target domain, which may be too restrictive [66]. For example, a sunny day image can be translated into a night image, but the reconstructed day image from the night image could not be necessarily sunny. In CUT, cycle–consistency is replaced with the mutual information maximization between corresponding source and target image patches [66]. This simplifies the training of the unpaired image translation to the training with a single generator (and its discriminator).

The information maximization between the input and output is done as in contrastive predictive coding [67] to associate output patch (query) with the corresponding input patch (positive) more than other input image patches (negatives) [66]. The query $\boldsymbol{v}$, positive $\boldsymbol{v}^+$ and N negatives $\boldsymbol{v}^-$ are mapped to a K–dimensional vector space, so that $\boldsymbol{v} \in \mathbb{R}^K$, $\boldsymbol{v}^+ \in \mathbb{R}^K$, and $\boldsymbol{v}^- \in \mathbb{R}^{N\times K}$ [66]. The constrastive loss is cast as a $(N+1)$–way classification problem, where the probability of the positive patch being selected over the negative patches is represented with cross–entropy:

$$l(\boldsymbol{v}, \boldsymbol{v}^+, \boldsymbol{v}^-) = -\log\left[\frac{\exp(\boldsymbol{v}^\top \boldsymbol{v}^+/\tau)}{\exp(\boldsymbol{v}^\top \boldsymbol{v}^+/\tau) + \sum_{n=1}^{N}\exp(\boldsymbol{v}^\top \boldsymbol{v}_n^-/\tau)}\right], \tag{2.36}$$

where $\tau = 0.07$ is a temperature scaling the distance between the query and other examples [66].

The same way as it is used in SimCLR [68], the projection head H is a two–layer perceptron that maps feature maps to the space, where the contrastive loss is applied. H is used to produce a stack of features $\{\boldsymbol{z}_l\}_L = \{H_l(G_l(\boldsymbol{x}))\}_L$

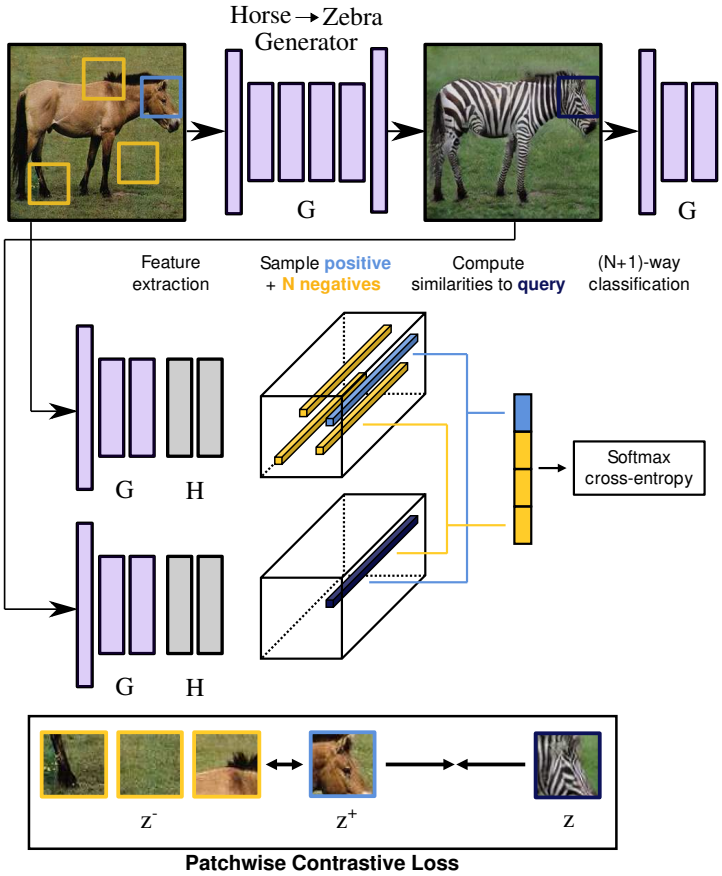


Figure 2.5: Patchwise contrastive loss for one–sided image translation with CUT [66]. The Horse→Zebra Generator (G) takes an input horse image and translates it into a fake zebra image. After this pass, the content in both images is ensured with patchwise contrastive loss minimalization, where feature maps of input and output images are extracted with selected generator layers and mapped with the projection head (H), then a query patch ($\boldsymbol{z}$) is sampled and compared with the positive patch $\boldsymbol{z}^+$ and negative patches ($\boldsymbol{z}^-$). G and H are optimized in a way that positive pair ($\boldsymbol{z}$, $\boldsymbol{z}^+$) is more similar than negative pairs $(\boldsymbol{z}, \boldsymbol{z}_n^-)_{n=1,\dots,N}$. Images were obtained and modified from [66].

for the input image features, and $\{\hat{\boldsymbol{z}}_l\}_L = \{H_l(G_l(G(\boldsymbol{x})))\}_L$ for the output image features, where $l \in \{1, 2, \dots, L\}$ is the index of the layer chosen for the consistency, L is the number of chosen layers, and G_l, H_l is the output of l–th chosen layer for the generator, projection layer, respectively, and additionally, $s \in \{1, \dots, S_l\}$ is the index of a spatial location at layer l, S_l is the number of spatial locations at layer l, and C_l is the number of channels at the specified layer [66]. The layers chosen for the contrastive loss are in the encoder part of the generator. The corresponding feature is referred as $\boldsymbol{z}_l^s \in \mathbb{R}^{C_l}$, other features as $\boldsymbol{z}_l^{S\setminus s} \in \mathbb{R}^{(S_l-1)\times C_l}$ [66]. Finally, the patchwise contrastive loss

$\mathcal{L}_{PatchNCE}$ can be defined as:

$$\mathcal{L}_{PatchNCE}(G, H, X) = \mathbb{E}_{\boldsymbol{x}\sim P_X} \sum_{l=1}^{L} \sum_{s=1}^{S_l} l(\hat{\boldsymbol{z}}_l^s, \boldsymbol{z}_l^s, \boldsymbol{z}_l^{S\setminus s}), \tag{2.37}$$

see Figure 2.5. The relative importance of individual regularizations can be controlled with λ_X and λ_Y in Equation 2.38. The default setting of these weights for the CUT is $\lambda_X = 1, \lambda_Y = 1$ [66]. Selected layers for $\mathcal{L}_{PatchNCE}$ are each convolutional layer in the encoder part of the generator, and it was shown by [66] when only the last layer is selected, GAN training often collapses.

The CUT optimization is similar to the pix2pix [56] formulation (covered in the Section 2.4.2) with two main differences:

- since CUT solves the unsupervised image translation task, the discriminator does not obtain the source domain image,
- image content is preserved with multilayer patchwise contrastive loss instead of L1 loss.

This results in an optimization for generator G, discriminator D, and projection head H formulated similarly to Park *et al.* [66] as follows:

$$\begin{aligned} &\min_D \mathcal{L}_{adv}^{(D)}(D, G, X, Y) \\ &\min_G \mathcal{L}_{adv}^{(G)}(D, G, X) + \lambda_X \mathcal{L}_{PatchNCE}(G, H, X) + \lambda_Y \mathcal{L}_{PatchNCE}(G, H, Y), \end{aligned} \tag{2.38}$$

where the adversial loss for discriminator $\mathcal{L}_{adv}^{(D)}$, and the adversial loss generator $\mathcal{L}_{adv}^{(G)}$ are:

$$\begin{aligned} \mathcal{L}_{adv}^{(D)}(D, G, X, Y) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim P_X}[(D(G(\boldsymbol{x})))^2] + \frac{1}{2}\mathbb{E}_{\boldsymbol{y}\sim P_Y}[(D(\boldsymbol{y}) - 1)^2] \\ \mathcal{L}_{adv}^{(G)}(D, G, X) &= \mathbb{E}_{\boldsymbol{x}\sim P_X}[(D(G(\boldsymbol{x})) - 1)^2]. \end{aligned} \tag{2.39}$$

2.5 Edge Detection

The task of edge detection is qualitatively close to the task of searching all pixel regions with sharp changes of intensity, color, or texture in the image [69]. Finding edges in the image lowers image data complexity and simplifies the subsequent tasks, such as image segmentation, or computing image descriptors, *etc.* [70, 52]. Edges are fairly invariant to illumination changes [52], which is exploited in this thesis in the novel GAN architecture. To extract edge maps from an image, the common approach is to use an **edge detector**. Two edge detectors are introduced: the first is a simple and effective Sobel operator, and the second one is a CNN–based HED edge detector.

2.5.1 Sobel Operator

Assuming, the edge point and its neighborhood have the highest difference in pixel intensities, one way how to detect edges is with maxima of gradients

(first derivatives) approximation. Gradients can be approximated using a small convolutional kernel. One of the popular convolution–based techniques is Sobel operator [71], which extracts image edge features $\boldsymbol{y}$ by convolving the input image $\boldsymbol{x}$ in the horizontal and vertical directions as:

$$\boldsymbol{G}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \star \boldsymbol{x}, \quad \boldsymbol{G}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \star \boldsymbol{x}, \quad \boldsymbol{y} = \sqrt{\boldsymbol{G}_x^2 + \boldsymbol{G}_y^2}, \tag{2.40}$$

where $\boldsymbol{G}_x$ and $\boldsymbol{G}_y$ are image responses in the right and down direction, respectively. To ensure the dimensions of $\boldsymbol{G}_x$ and $\boldsymbol{G}_y$ are the same as the input, the image $\boldsymbol{x}$ is padded with one pixel replicated at each side before convolving. Also, if the input image is a color, it is first transformed into a grayscale image. The result of the Sobel operator is $\boldsymbol{y}$, which expresses the magnitudes of edges at each pixel of the input image.

2.5.2 HED

Detecting image edges as taking image derivatives highlights high frequencies, which then produces noisy edge points that would not be considered as edges by humans [72]. Hollistically–nested edge detector (HED) [73] is CNN–based edge detector that addresses the edge ambiguity with learning on Berkeley Segmentation Dataset and Benchmark (BSDS 500) [74, 70] to match the side output of each convolutional layer with the ground–truth image through side outputs weighted fusion [73]. In other words, each side output is passed to the next layer as in CNN, and additionally, all side outputs are used to fuse the output edge map.

Let X be the image domain, and let Y be the edge map domain. At inference time, given an input image $\boldsymbol{x} \in X$, an edge map $\hat{\boldsymbol{y}} \in Y$ is extracted with HED detector E as:

$$\hat{\boldsymbol{y}} = \text{average}(\hat{\boldsymbol{y}}_{fuse}, \hat{\boldsymbol{y}}_{side}^{(1)}, \ldots, \hat{\boldsymbol{y}}_{side}^{(M)}) = E(\boldsymbol{x}), \tag{2.41}$$

where $\hat{\boldsymbol{y}}_{fuse}$ is the edge responses obtained after the $\hat{\boldsymbol{y}}_{side}^{(1)}, \ldots, \hat{\boldsymbol{y}}_{side}^{(M)}$ side outputs fusion, and M is the number of side outputs. To achieve this, E is optimized as in supervised learning task using training pairs $(\boldsymbol{x}, \boldsymbol{y})$ of input image $\boldsymbol{x}$ and ground–truth image edge maps $\boldsymbol{y} \in Y, \boldsymbol{y} \in \{0, 1\}^{H \times W}$ with the following two losses:

Side Outputs Loss. The goal of the side outputs is to produce an intermediate edge map approaching the ground–truth edge map since the first layers of the detector [73]. Let $E^{(m)}$ denote the m–th side output layer of the detector. Each of the side output layer is optimized with class–balanced binary cross–entropy loss, which is chosen by [73] to balance the bias between the rare edge pixels and large amounts of non–edge pixels. Let Y_{train}^{+} and Y_{train}^{-} be the sets of all training ground–truth non–edge and edge labels, respectively. The class–balancing weights are defined as $\beta = |Y_{train}^{-}|/|Y_{train}|$ and $(1 - \beta) = |Y_{train}^{+}|/|Y_{train}|$, where $Y_{train} = Y_{train}^{-} \cup Y_{train}^{+}$ [73]. Using these

weights, each side output layer is optimized with an image–level loss function for side outputs as:

$$\ell_{side}(m, E, X, Y) = -\mathbb{E}_{\boldsymbol{x}\sim P_X, \boldsymbol{y}\sim P_Y}[\beta\, \boldsymbol{y} \log(\hat{\boldsymbol{y}}^{(m)}_{side}) + (1-\beta)(1-\boldsymbol{y})\log(1-\hat{\boldsymbol{y}}^{(m)}_{side})], \tag{2.42}$$

where $\hat{\boldsymbol{y}}^{(m)}_{side} = (E^{(1)} \circ E^{(2)} \circ \cdots \circ E^{(m)})(\boldsymbol{x})$ is the m–th side output [73]. Using $\ell_{side}(m, E, X, Y)$, the side outputs loss is:

$$\mathcal{L}_{side}(E, X, Y) = \sum_{m=1}^{M} \ell_{side}(m, E, X, Y). \tag{2.43}$$

Fusion Loss. To directly use all side outputs, [73] add a (last) fusion layer E^{fuse} that attempts to learn weights per each side output $1, \ldots, M$; specifically $\hat{\boldsymbol{y}}_{fuse} = E^{fuse}(\hat{\boldsymbol{y}}_{fuse}) = E^{fuse}(\hat{\boldsymbol{y}}^{(1)}_{side}, \ldots, \hat{\boldsymbol{y}}^{(M)}_{side})$ is trained via binary cross–entropy loss:

$$\mathcal{L}_{fuse}(E, X, Y) = -\mathbb{E}_{\boldsymbol{x}\sim P_X, \boldsymbol{y}\sim P_Y}[\boldsymbol{y} \log(\hat{\boldsymbol{y}}_{fuse}) + (1-\boldsymbol{y})\log(1-\hat{\boldsymbol{y}}_{fuse})]. \tag{2.44}$$

Overall, HED optimization of the detector E is optimized by [73] as:

$$\min_{E} \mathcal{L}_{side}(E, X, Y) + \mathcal{L}_{fuse}(E, X, Y). \tag{2.45}$$

2.6 Related Work

First, in Section 2.6.1 and Section 2.6.2, prior work relevant to this thesis is briefly covered. In the last Section 2.6.3, image translation methods and their use–cases related to this thesis are briefly described.

2.6.1 Day–Night Image Retrieval

The prior work of Jenicek and Chum [51] is the closest to this thesis. They tackle the image illumination changes by aligning the day and night visual domains in two steps:

1. a photometric normalization is proposed to bring the appearance of day and night images more together through handcrafted CLAHE [72], or learned U-Net network,
2. embedding network is trained on matching day and night pairs collected from 3D reconstructions from [75], which was introduced as SfM N/D dataset [51].

In this thesis, day–night image retrieval is also evaluated with CLAHE because it helps the retrieval performance. On the other hand, data augmentation with day→night image translation in this thesis removes the requirement to obtain matching day–night image pairs, and also increases the diversity of night image data.

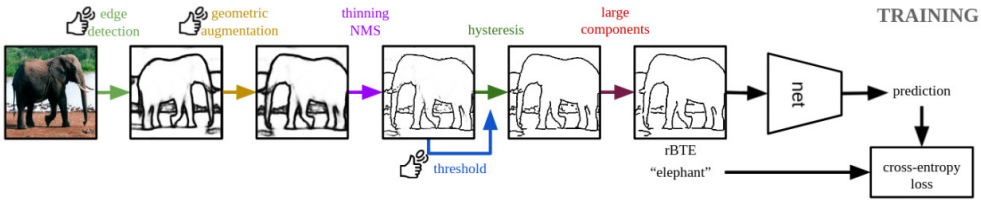


Figure 2.6: Overview of the training pipeline with rBTE. A photo is transformed into rBTE using multiple edge detectors (green) followed by geometric augmentations (yellow), and then, edge maps are thinned using non–maxima–suppression (purple), hysteresis with a random threshold method (green and blue) ended with large connected–components preservation (red). The transformed image together with the annotation of the former photo is then used to train a network classifier (net) with the cross–entropy loss. The whole Figure is taken from [32].

2.6.2 Sketch Recognition

Sketch Classification

The prior work of Efthymiadis *et al.* [32] aims to classify sketches at the inference time without access to sketches at the training time. To bridge the domain gap, they proposed to transform natural images into a pseudo domain called randomized Binary Thin Edges (rBTEs) [32] that visually approximates the sketch images, see Figure 2.6.

Edge–based Retrieval

The method of Radenovic *et al.* [52] experimented with metric learning only on edge maps using EdgeMAC descriptor. They observed that edges well survive the day–night transition, which results in no need to obtain more expensive night images, but [52, 51] observed that EdgeMAC performs poorly on standard image retrieval datasets due to the information loss during the transformation of image into the edge map.

Lengyel *et al.* [76] proposed CIConv, which is a method for zero–shot domain adaptation. Based on the observation that there are large distribution shifts in CNN feature maps between day and night image inputs, [76] proposed an edge detector implemented as a color invariant convolution layer that is added before the backbone network. CIConv has very promising results in night retrieval without providing any night training images, but the retrieval performance is worse on standard retrieval benchmarks.

2.6.3 Image–to–image Translation

GAN

In this thesis, pix2pix [56] (Section 2.4.2), CycleGAN [57] (Section 2.4.3), DRIT++ [58, 59], and CUT [66] (Section 2.4.5) were covered; yet there are

many more GAN–based networks for image translation [77].

StarGANv2 [78] might be considered as the state–of–the–art by many because it outperforms MUNIT [79], DRIT [58], and MSGAN [63]; however the quantitative comparisons were done only on CelebA-HQ [21] and AFHQ [78], which are datasets containing only human or animal faces making the use–case less related to image retrieval or visual localization.

With the knowledge that synthesized images are used in tasks, such as landmark retrieval or visual localization, one could use more advanced object–aware image translation to better enhance the objects of interest for the downstream task. For instance, INIT [80], DUNIT [81], and DE-CycleGAN [82] are instance–aware GAN architectures leveraging object detection models, and Multimodal AugGAN [83] integrates semantic segmentation models of both domains with a multimodal generator network. From my personal experience, manual deployment of these kinds of networks is too complex, and it takes a long time to train them from scratch. Instead, a novel lightweight HED$^{\text{N}}$GAN is introduced in this thesis, which takes orders of magnitude less time to train it and it is as good as CycleGAN [57] in day→night image translation for image retrieval.

VAE

Variational autoencoders (VAEs) [84] are similar to the GANs, as they both are tasked to generate a distribution $p(\boldsymbol{x})$, but VAEs do not have a discriminator that is trying to be tricked with a generator. The architecture of VAE is composed of an encoder that maps data $\boldsymbol{x}$ into a latent vector $\boldsymbol{z}$ approximating $q(\boldsymbol{z}|\boldsymbol{x})$ and a decoder that maps the latent $\boldsymbol{z}$ back to the data $\hat{\boldsymbol{x}}$ approximating $p(\boldsymbol{x}|\boldsymbol{z})$. VAEs are more stable to train than GANs [85], but image quality synthesized with VAE is worse compared to GAN–based reconstruction [77]. Nonetheless, UNIT [62], BicycleGAN [61], DRIT [58, 59] are examples of both GAN and VAE–based models because they both use a latent space for the image translation and also tries to generate image indistinguishable from real images by the discriminator. Also, by using VAE, GAN architecture can be adapted for multimodal image translation [61, 58, 59].

Translation for Domain Adaptation

Image translation was used in many computer vision tasks, which is similar to this thesis. The most similar to the day–night image retrieval are mentioned as follows:

Arruda *et al.* [86] use CycleGAN to translate day images into night images in car detection, while having image annotations only for day images.

Annosleh *et al.* [87] proposed ToDayGAN for visual localization to translate night image into a day image at inference time using the same generator as in CycleGAN [57], but trained with three discriminators based on different real and fake image features: the first discriminator for images in grayscale, the second discriminator for RGB blurred images, and the third discriminator for image gradients.

Mueller *et al.* [88] performed image translation from a rendered domain obtained from 3D recostructions into photo–realistic image domain showing that feature matching, image retrieval, and visual localization benefits from synthetic data, and additionally, training on merely synthetic images in the visual localization achieved similar results as training only on manually captured images.

Lou *et al.* [89] designed a Feature Distance Adversarial Network (FDA-Net) that is similar to GAN to online generate hard negatives for a car reidentification task.

Chapter 3

Method

3.1 Day–Night Image Retrieval

In this Section, two new GAN generators for unpaired image translation are introduced. Those generators are more lightweight than the vanilla generators covered in the previous background Chapter 2. Finally, a resolution of how a trained generator aids the day–night image retrieval in metric learning is described.

3.1.1 Day–Night Translation Notation

The goal of day–night image translation is to transform day images into night images. Let X be the set of day domain images, where the day image $\boldsymbol{x}$ is from the day visual domain if $\boldsymbol{x} \in X$. Let P_X denote the data distribution of day images. Image $\boldsymbol{x}$ follows distribution P_X, when $\boldsymbol{x} \sim P_X$. Night domain Y, night image $\boldsymbol{y}$, and night domain data distribution P_Y are defined similarly. A day→night generator $G : X \to Y$ is trained to translate day image $\boldsymbol{x}$ into synthetic night image $G(\boldsymbol{x})$.

3.1.2 Edge Consistency Generators

The task is unpaired day→night image translation performed with a GAN–based generator. The approach to the novel generators is based on the observation that edges tend to remain preserved when illumination conditions change [51, 75]. Therefore, **edge consistency** is used as a regularization in the generator, which forces the generator to generate images that have edge maps similar to the edge maps of the input image while still allowing the translation to produce a visually different image. Other GAN models for unpaired image translation, such as DRIT [58, 59] and CycleGAN [57], require a large amount of time to train from scratch. In particular, DRIT architecture shares latent vectors among decoders and discriminators forcing them to retain the computational graph during the training, which slows down the backpropagation, and in CycleGAN, the first generator relies on a second generator that is learned from scratch and presumably provides imprecise feedback to the first generator at the beginning of the training.

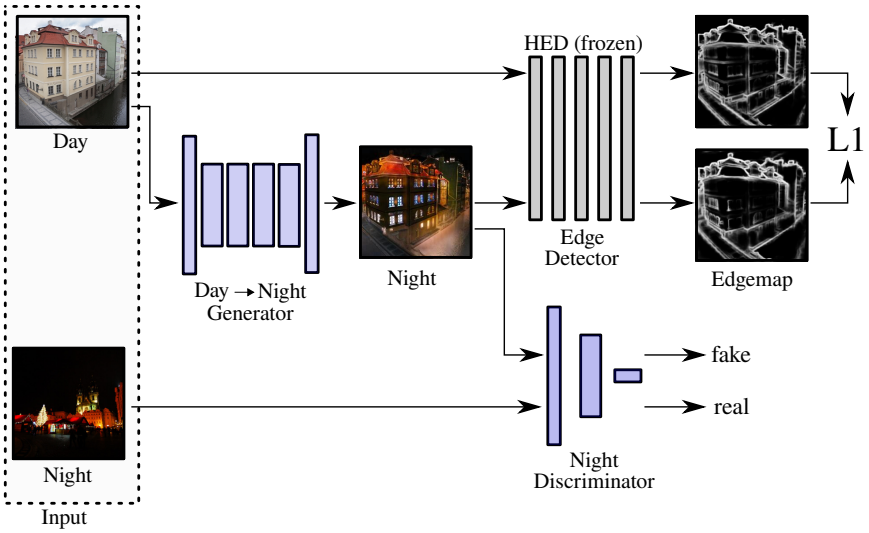


Figure 3.1: Optimization step of the HEDGAN training. The Day→Night Generator takes an input day image and translates it into a fake night image while maintaining consistency in the edge map using an L1 loss between the outputs of the HED detector. The Night Discriminator then determines whether the generated night image and the input night image are real or fake.

CUT [66] replaces the cycle consistency with feature map patches contrastive loss, which makes it more lightweight but still requires passing the image through the generator four times. In edge consistency–based generators, an input image passes through the generator only once, and then, the real and fake images pass through the edge detector also once.

HEDGAN

The architecture of HEDGAN consists of three models: a generator G, a discriminator D, and an edge detector E. The goal of generator and discriminator is the same as in other GAN–based training, while the goal of the edge detector is to provide edge maps of real and fake images. To obtain first order gradients in the backpropagation, the edge detector must be differentiable. For this purpose, HED [73] is employed as the edge detector. The full network training is displayed in Figure 3.1.

The generator and the discriminator weights are initialized at random, while HED weights are initialized with pretrained HED [90] and remain unchanged during the training. Each iteration, input image $\boldsymbol{x}$ from day visual domain is translated with the generator into a fake image $G(\boldsymbol{x})$ in the night target domain. Then, edge maps of input image $E(\boldsymbol{x})$ and output image $E(G(\boldsymbol{x}))$ are extracted and compared pixel–wise with L1 distance with the edge consistency loss $\mathcal{L}_{edge}$ forcing the edges between the input and fake image to be consistent:

$$\mathcal{L}_{edge}(G, E, X) = \mathbb{E}_{\boldsymbol{x} \sim P_X}[||E(\boldsymbol{x}) - E(G(\boldsymbol{x}))||_1]. \tag{3.1}$$

The discriminator is deployed in a standard unpaired image–to–image translation fashion and applied to the fake image $G(\boldsymbol{x})$ ensuring the fake night image is indistinguishable from real night image distribution by predicting $D(G(\boldsymbol{x}))$ is fake and $D(\boldsymbol{y})$ is real, training the generator to synthesize night images by tricking the discriminator to predict $D(G(\boldsymbol{x}))$ as real an image.[1] The adversial loss for the discriminator $\mathcal{L}_{adv}^{(D)}$, and the generator $\mathcal{L}_{adv}^{(G)}$ can be formulated as:

$$\begin{aligned}\mathcal{L}_{adv}^{(D)}(D,G,X,Y) &= \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim P_X}[(D(G(\boldsymbol{x})))^2] + \frac{1}{2}\mathbb{E}_{\boldsymbol{y}\sim P_Y}[(D(\boldsymbol{y})-1)^2] \\ \mathcal{L}_{adv}^{(G)}(D,G,X) &= \mathbb{E}_{\boldsymbol{x}\sim P_X}[(D(G(\boldsymbol{x}))-1)^2].\end{aligned} \tag{3.2}$$

Putting this together, this results in optimization of the unpaired image–to–image translation with LSGAN [13] (covered in Section 2.1.2) adversarial training, and with new edge consistency regularization, formulated as follows:

$$\begin{aligned}&\min_D \mathcal{L}_{adv}^{(D)}(D,G,X,Y) \\ &\min_G \mathcal{L}_{adv}^{(G)}(D,G,X) + \lambda \mathcal{L}_{edge}(G,E,X),\end{aligned} \tag{3.3}$$

The strength of the edge consistency is controlled with λ, an additional hyperparameter, which is $\lambda = 5$ in all experiments.

HED$^{\text{N}}$GAN

HED detector [73] can sometimes miss edges in the night images, which can result in imprecise feedback to the generator. This is because the HED detector is primarily trained on images taken during the day. To improve the edge detection in night images, a new edge detector called HED$^{\text{N}}$ is added and trained jointly with the generator and the discriminator so that HED$^{\text{N}}$ has similar responses on both real day and fake night images with the help of HED. This allows the edge consistency to be measured more accurately during the generator training.

The architecture of HED$^{\text{N}}$GAN consists of four models: a generator G, a discriminator D, a student edge detector E^N (HED$^{\text{N}}$), and a teacher edge detector E (HED). The generator and discriminator are trained similarly to HEDGAN training. The student is taught by the teacher to preserve edge maps between day–night and day–day image pairs. The role of the teacher is only to provide day image edge maps that are considered ground–truth in the training. The full network training is displayed in Figure 3.2.

More specifically, students' weights are initialized the same as the teacher's [90], and then, only students' weights are updated during the training. The adversarial loss of the discriminator and generator is the same, but in the edge consistency loss of the generator, HED$^{\text{N}}$ edge maps $E^N(G(\boldsymbol{x}))$ instead of HED edge maps $E(G(\boldsymbol{x}))$ of the fake image are compared with HED edge

[1] The discriminator output is not interval $[0,1]$, but it is unbounded. This is because the sigmoid is removed from the discriminator architecture and it is trained as in LSGAN [13] (covered in Section 2.1.2).

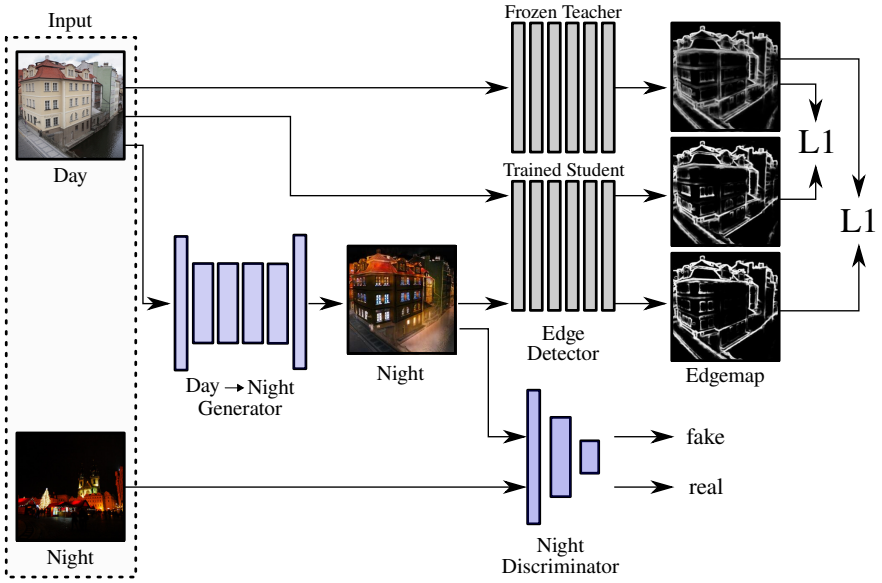


Figure 3.2: Optimization step of the HED$^{\mathrm{N}}$GAN training. The Day→Night Generator takes an input day image and translates it into a fake night image while maintaining consistency in the edge map using an L1 loss between the outputs of the HED and HED$^{\mathrm{N}}$ detectors. The Night Discriminator then determines whether the generated night image and the input night image are real or fake. The HED$^{\mathrm{N}}$ edge detector is trained by the HED edge detector to produce night image edge maps while still preserving the day image edge maps.

maps $E(\boldsymbol{x})$ of the input day image, which is reflected in the edge consistency loss for the generator:

$$\mathcal{L}_{edge}^{(G)}(G, E^N, E, X) = \mathbb{E}_{\boldsymbol{x}\sim P_X}[||E(\boldsymbol{x}) - E^N(G(\boldsymbol{x}))||_1]. \tag{3.4}$$

The student E^N is taught by the teacher E to detect more edges in the synthesized night images by being forced to output edge maps $E^N(G(\boldsymbol{x}))$ of the fake night image $G(\boldsymbol{x})$ that are the same as the ground–truth edge maps $E(\boldsymbol{x})$ of the input day image $\boldsymbol{x}$. To ensure that the student does not forget how to detect edges in day images, it is also trained to preserve the day image edge maps $E^N(\boldsymbol{x})$ with the edge maps $E(\boldsymbol{x})$ of the teacher. This is captured by the edge consistency loss for the student:

$$\begin{aligned}\mathcal{L}_{edge}^{(E)}(G, E^N, E, X) = \mathbb{E}_{\boldsymbol{x}\sim P_X}[||E(\boldsymbol{x}) - E^N(G(\boldsymbol{x}))||_1] \\ + \mathbb{E}_{\boldsymbol{x}\sim P_X}[||E(\boldsymbol{x}) - E^N(\boldsymbol{x})||_1].\end{aligned} \tag{3.5}$$

Then, the whole HED$^{\mathrm{N}}$GAN training can be formulated as follows:

$$\begin{aligned}&\min_D \mathcal{L}_{adv}^{(D)}(D, G, X, Y) \\ &\min_G \mathcal{L}_{adv}^{(G)}(D, G, X) + \lambda \mathcal{L}_{edge}^{(G)}(G, E^N, E, X) \\ &\min_{E^N} \mathcal{L}_{edge}^{(E)}(G, E^N, E, X).\end{aligned} \tag{3.6}$$

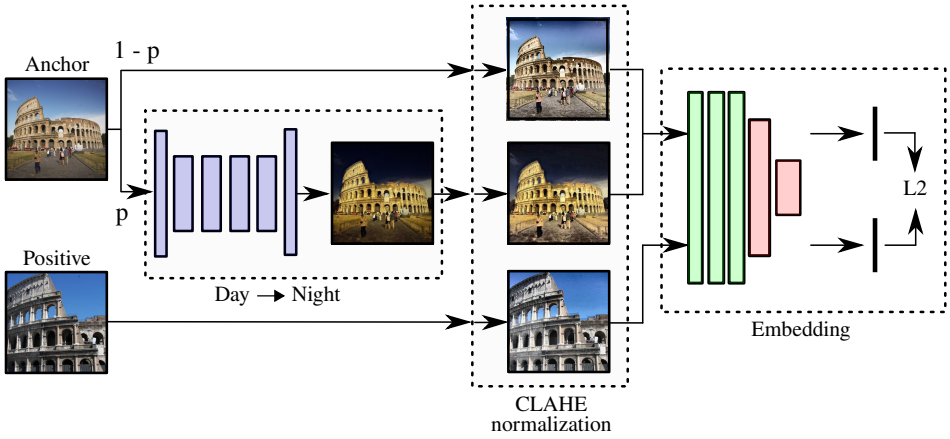


Figure 3.3: Data augmentation and photometric normalization in metric learning. An anchor is translated from day into night visual domain with probability p (Day→Night block). The resulting image together with a positive and negative (not illustrated) is photometrically normalized (CLAHE normalization block). Then, positive pair and negative pairs (not illustrated) forward through the embedding network. Finally, the L2 distance between positive pair global descriptors is minimized, while for negative pair global descriptors it is maximized up to a margin.

Adversarial losses $\mathcal{L}_{adv}^{(D)}$ and $\mathcal{L}_{adv}^{(G)}$ are same as in HEDGAN training (Equation 3.2). The strength of the edge consistency is controlled with λ, an additional hyperparameter, which is $\lambda = 5$ in all experiments.

3.1.3 Metric Learning

The learning of image global descriptor is approached with metric learning using a siamese network. The task is to make the embedding $f^{D/N} : \mathrm{X} \to \mathrm{Y}$ that represents an image $\mathbf{x}$ as a global descriptor $\mathbf{y}$ that is invariant to image visual appearance changes as much as possible. Unfortunately, not enough training data are available to provide full information to the embedding network about image retrieval under varying appearance conditions. One technique, how to make the embedding model generalize better is to train it on more data, including creating synthetic examples and adding them to the training dataset, which is called data augmentation in deep learning terminology [2]. Prior work of [15, 51] is followed, where the data augmentation in the embedding training is achieved with a trained generator, that synthesizes night training images from day training images. This process is displayed in Figure 3.3.

Training the embedding network is approached with metric learning, where the contribution of this work is only day→night image translation performed for the anchor at random. Before the training data mining and subsequent metric learning, first, each anchor is translated from day into night visual domain with a trained day→night generator, and second, CLAHE photometric

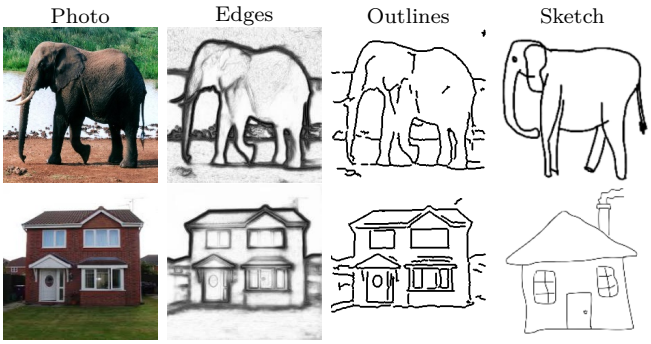


Figure 3.4: Examples of photo transformations towards sketches. From the natural image (first column) edges are extracted (second column). To bring the visual appearance closer to the true sketches (fourth column), the edges can be further thinned into outlines (third column). Outline images can be used as training data for sketch recognition. In this Figure, edges were obtained with HED [73] and outlines were obtained from [32].

normalization [72] is applied on each image. The same way as in [15, 51, 43] embedding network architecture is constructed from a model for classification, such as VGG-16 [6], ResNet-101 [7], *etc.* where the last linear layers are replaced with GeM layer [43] followed by L2 normalization layer; this is described in greater detail in the background of retrieval with CNNs in Section 2.3.2. At the start of the training, the embedding network is initialized with weights of ImageNet pretrained network [25]. Then, the embedding is finetuned for the metric learning task with the contrastive loss.

3.2 Sketch Recognition

This section focuses on the photo–sketch visual domain shift in image recognition. First, a generator is trained using a modified paired image translation. Second, the generator trained in the previous step is used together with an edge detector to produce approximate sketch images for the downstream task. Although image classification and image retrieval have dissimilar formulations, it was observed by [91] that those two tasks can be solved using the same algorithm. Therefore, to evaluate the effectiveness of the image translation in the downstream task, the task can be image classification instead of image retrieval. In addition, no available training sketches are considered, and therefore, natural photos are used to approximate sketch images. The intuition behind the approximation is displayed in Figure 3.4.

3.2.1 Thin-pix2pix

The goal is to construct the transformation that transforms photos to approximate sketch images. This is achieved using two models: an edge detector, followed by a generator that transforms edges into outlines. Thin edge outlines serve as the approximation of sketch images. The prior work of [32]

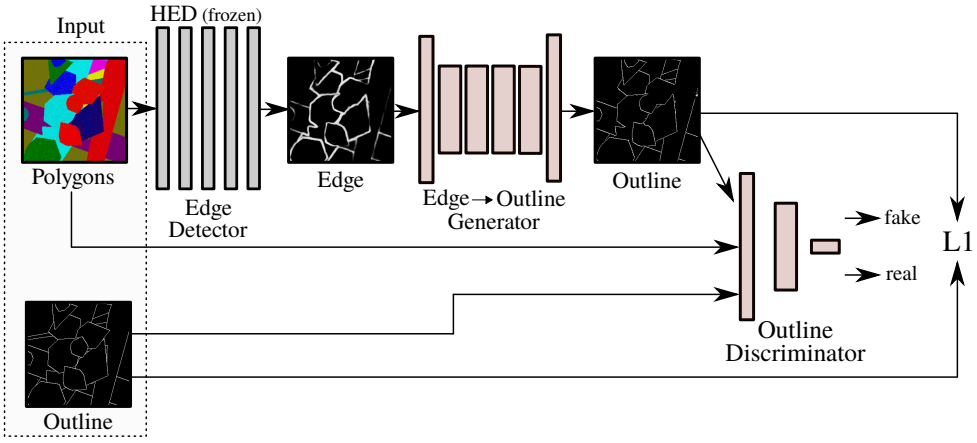


Figure 3.5: Optimization step of the Thin-pix2pix training for the thinning task. First, edges are extracted with HED [73] from input image polygons. Then, the Edge→Outline Generator translates these edges into a fake outlines image. After this forward pass, structure consistency is ensured with L1 loss minimalization. The Outline Discriminator then determines whether the translated image and the input outline image are real or fake with the input polygons image condition.

approximates sketches with handcrafted rBTE pseudo–domain. However, this approach has a drawback with many function parameters needing to be tuned. In this work, the transformation is composed of HED [73] edge detector and a generator trained for paired image translation.

The purpose of the generator is to synthesize thin image outlines from an input image. For this purpose, the generator is trained using self–supervised learning inspired by pix2pix [56]. This model is therefore called Thin-pix2pix. The training of the generator is displayed in Figure 3.5. The training of the Thin-pix2pix uses almost the same as the pix2pix optimization with three differences:

- the Thin-pix2pix generator is composed of the pretrained HED [73] detector followed by ResNet generator [57]; HED weights remain unchanged during the training,
- the training data consists of training pairs of generated random polygons and their one–pixel thin outlines,
- the relative importance of the discriminator is stronger compared to pix2pix ($\lambda = 10$ instead of former the 100 in Equation 2.17), which significantly reduces the noise in the synthesized outline images produced by the generator.

Each random polygon image consists of 20 randomly generated polygons of 14 different possible colors and each polygon is filled with a normally distributed random noise.

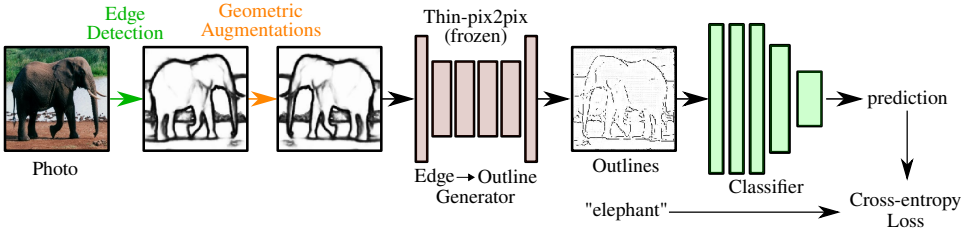


Figure 3.6: Overview of the training pipeline with Thin-pix2pix. A photo is transformed into outlines using multiple edge detectors (green) followed by geometric augmentations (yellow). Edge maps are transformed with Thin-pix2pix generator into outlines. The transformed image (outlines) together with the annotation of the former photo is then used to train a network classifier with the cross–entropy loss. This Figure is based on [32].

3.2.2 Sketch Classification

The downstream task is class–level sketch classification. Let X be the set of images, let S be the set of sketches, and let Y be the set of labels. The classification is performed with the classifier $f^{Sketch} : \mathrm{S} \to \mathrm{Y}$ that classifies a sketch $\boldsymbol{s}$ with a class–level label y. Following the prior work of [32], the classifier is obtained using supervised learning that trains f^{Sketch} with training pairs $(\boldsymbol{s}, \mathrm{y})$. To obtain any of these pairs, a natural image $\boldsymbol{x}$ is transformed into image outline $T(\boldsymbol{x})$, where $T : \mathrm{X} \to \mathrm{S}$. The classifier is trained using the transformation T providing $(T(\boldsymbol{x}), \mathrm{y})$ training pairs. During the classifier training iteration, $T(\boldsymbol{x})$ passes through the classifier, and its prediction of the class is compared with the true class y using cross–entropy loss.

The transformation T is a composition of multiple transformations. The composition, together with the classifier training is displayed in Figure 3.6. At the training time, given an input image, edge maps are extracted, then random geometric augmentations are performed, and finally, edge maps are thinned with the Thin-pix2pix generator. At the inference time, the input sketch is only thinned with the generator.

Chapter 4

Implementation

4.1 Datasets

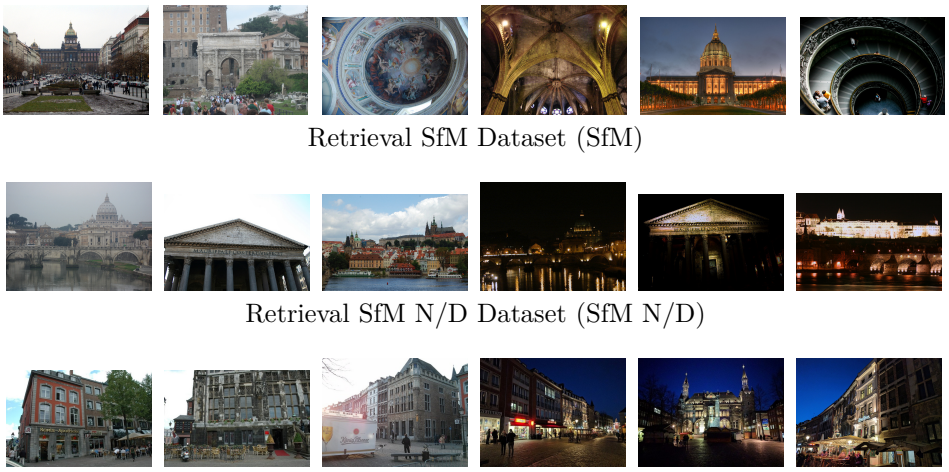
In this Section, all datasets needed to perform the experiments are described. In day–night image retrieval experiments, two datasets are used for the generator training, and two datasets are used for the image retrieval, while for the evaluation, three different datasets are used. In sketch classification experiments, two datasets are used and split for training and evaluation. The evaluation protocol for the day–night image retrieval is mean average precision (mAP), and the sketch classification is evaluated with classification accuracy.

4.1.1 Training Datasets

Retrieval SfM Dataset (*SfM*) is based on the dataset of Schronberger *et al.* [92] consisting of 7.4 million images of popular landmarks and cities across the world downloaded from Flickr, from which were then reconstructed 3D models with retrieval–SfM pipeline allowing fully automatic annotation of positives and negatives without any human interaction [43]. This dataset is used for the baseline retrieval evaluation as well as for the generator training and data augmentation. For all image retrieval experiments, SfM contains 98045 images from 3D reconstructions. For the day→night generator training, annotations of [1] were used for day and night images and images having any resolution lower than 512 px were removed to allow further random scale–crop augmentation, which results in 86385 day images and 10039 night images.

Retrieval SfM N/D Dataset (*SfM N/D*) was introduced by [51], who constructed additional positive pair with different illumination conditions. This dataset is an enrichment of SfM with the day–night positive pairs gathered from 3D reconstructions. Notice that training the day→night augmentation does not require the use of day–night pairs, since all evaluated augmentations are unpaired image–to–image translations. Nonetheless, night images 3D reconstructions of [1] hint, that night images have low diversity.

Aachen Day–Night Dataset (*Aachen*) contains images of the old inner city of Aachen in Germany [93, 94]. The dataset consists of 5152 day images and 191 night images, where night images are taken with mobile phones with



Retrieval SfM Dataset (SfM)

Retrieval SfM N/D Dataset (SfM N/D)

Aachen Day-Night Dataset (Aachen)

Figure 4.1: Training data image samples for the image-to-image translation and image retrieval. Row order from top to bottom corresponds to SfM [92], SfM N/D [51], and Aachen [93, 94] datasets. In each row, the first three left images correspond to the day visual domain and the last three right images correspond to the night visual domain. For the SfM N/D dataset, day and night images form positive pairs (2nd row, 1st and 4th pair, 2nd and 5th pair, and 3rd and 6th pair). SfM and SfM N/D datasets were used in image retrieval training, while SfM and Aachen datasets were used in the image–to–image translation training.

HDR setting. From the dataset, only day-night annotations are used to train the CycleGAN on day and night image sets.

4.1.2 Evaluation Datasets

Three datasets are used in the final image retrieval evaluation and two datasets for the sketch classification evaluation.

Revisited Oxford and Paris Datasets ($\mathcal{R}$Oxf and $\mathcal{R}$Par) are standard image retrieval datasets revisited by [95] consisting of 4993 and 6322 day images respectively, capturing buildings. All retrieval performance scores are reported on medium difficulty evaluation. The purpose of these datasets is to measure, whether the retrieval in regular conditions does not drop below the baseline.

24/7 Tokyo Dataset (Tokyo 24/7) [96] is a collection of smartphone pictures taken in different light conditions, such as day, night, and sunset consisting of 1125 images capturing 375 scenes from 125 distinct locations. The evaluation of day–night retrieval was proposed by [51], who uses each image as a query, images from the same scene but different lighting conditions as positives, and images from different locations as negative. In this thesis, the day–night retrieval is evaluated with the same protocol as in [51].

PACS [34] is a dataset suitable for domain generalization tasks consisting of four visual domains: *photo* (1670 images), *art painting* (2048 images),

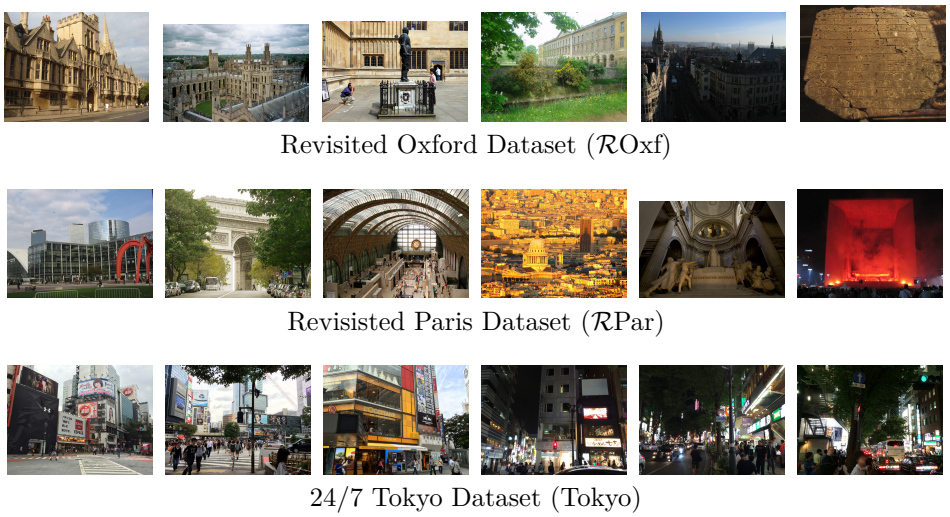


Figure 4.2: Evaluation data image samples for the image retrieval. Row order from top to bottom corresponds to $\mathcal{R}$Oxf [95], $\mathcal{R}$Par [95], and Tokyo [96] datasets. For the Tokyo dataset, the first three left images correspond to the day visual domain and the last three right images correspond to the night visual domain. All three datasets are used for image retrieval performance evaluation.

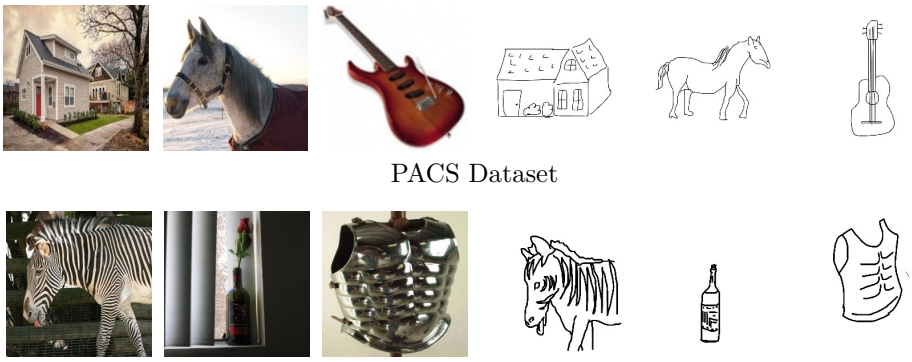
cartoon (2344 images), and *sketch* (3929 images), where each of these domains contains 7 classes: *dog*, *elephant*, *giraffe*, *guitar*, *house*, *horse*, and *person* [34]. In all sketch classification experiments, the practice of [32] is followed, that is, only images in photo class are used for training, where 1499 are used for the training and 171 for the validation, and all 3929 sketches are used in the testing part.

Sketchy [97] is a large–scale dataset obtained with crowd workers that have been asked to sketch a specific object. The dataset counts a total of 75471 sketches of 12500 objects each belonging to one of 125 classes. Following the practice of [32], sketch recognition training use 11500 images and the corresponding 68418 sketches, where 80 % of those images are used for the training and the remaining images for validation; the testing part uses 1250 images and the corresponding 7063 sketches as in [97].

4.2 Day–Night Image Retrieval

In the first step, a generator needed for data augmentation is trained. In the second step, image retrieval is learned using metric learning, where the generator from the previous step is used to translate day images into night images producing positive night–day pairs.[1]

[1] Image retrieval codebase and implementation progress is available online at CTU Gitlab at https://gitlab.fel.cvut.cz/jenicto2/mdir



PACS Dataset

Sketchy Dataset

Figure 4.3: Training and evaluation data image samples for the sketch classification task. Row order from top to bottom corresponds to PACS [34], and Sketchy [97] datasets. In each row, the first three left images correspond to the natural photo visual domain and the last three right images correspond to the sketch visual domain. Both these datasets are used for sketch classification training and evaluation.

4.2.1 Generator Training

In all experiments, the training data for the GAN training are preprocessed the same. Each input image is randomly downscaled by the factor from 0.8 up to 1 and then randomly cropped to the final size of 256×256 pixels. Each training epoch has 10000 iterations, where in each iteration a pair of images in source and target visual domains are used.

Vanilla Generators

For CycleGAN [57], CUT [66], and DRIT [59], their original settings are used. For clarity, the number of training epochs for models fully converge is 100 epochs for CycleGAN, 50 epochs for DRIT, and 300 epochs for DRIT. During the training of all three models, the learning rate linearly decays to zero over the last half of the training epochs.

For CUT [66], changes from the original implementation are the following:

- all weights are initialized with He initialization [25] instead of weight initialization from $\mathbf{N}(0, 0.02)$,
- downsampling and upsampling convolutional layers have trainable weights (same as in CycleGAN) instead of fixed weights,
- first layer output is omitted in the contrastive loss (applied on layers 4,7,10,14 instead of former 0,4,8,12,16)
- weight of identity loss λ_Y is set to 10 instead of 1.

The first three modifications aid the retrieval performance. However, the third modification with omitting the first layer might be questionable, since experiments [66] show using only the output of the last layer in contrastive loss destabilizes the training. I observed similar behavior and stabilized the training with stronger identity loss as the fourth modification.

Edge Consistecy Generators

Describing only HED$^{\mathrm{N}}$GAN is enough since HEDGAN is trained the same as HED$^{\mathrm{N}}$GAN (without finetuning HED).

HED$^{\mathrm{N}}$GAN architecture is composed of ResNet generator [98, 57], PatchGAN discriminator [56], and two HED edge detectors [73], where one is frozen and second is trained. HED$^{\mathrm{N}}$GAN optimization step is the same as the CycleGAN [57] for one generator, with the differences of cycle consistency being replaced by edge consistency for the generator, and additional edge detector optimization step. In contrast with CycleGAN [57], the generator and the discriminator use batch normalization instead of instance normalization. The generator and the discriminator are initialized with He initialization [25], while both HED detectors, student and teacher, are initialized with the same pretrained weights obtained from the pretrained PyTorch reimplementation[2] of [90]. The generator and the discriminator are trained from scratch similarly to [57], whereas the teacher weights are not updated during the training, and the student is finetuned with Adam optimizer [99] with learning rate 10^{-6}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay 0.0002. All three networks are trained jointly with batch size 10. Concerning the L1 losses between edge maps, in the detector optimization step, the L1 loss is applied between outputs before the sigmoid function, while in the generator optimization step, the loss is applied after the sigmoid function.

Also, the Sobel operator was evaluated as an edge detector in the edge consistency–based GAN training. This method is called SobelGAN. In the SobelGAN training, only the edge detector changed, all other hyperparameters remain the same as in the HEDGAN training.

4.2.2 Metric Learning

Metric learning follows the practice of [51] with the differences of the same training data obtaining annotations from 3D reconstructions are used as in the baseline [43], diverse anchor images are mined, and data augmentation with the generator is performed. For the CNN embedding backbones, VGG-16 [6], ResNet-18 [7], ResNet-50 [7], or ResNet-101 [7] are used. The backbone is initialized with weights of ImageNet pretrained network [100], and then the embedding network is finetuned on the SfM dataset [48, 51]. In the preprocessing steps, each input image is downscaled to the size of 362 px per dimension, then data augmentation takes place (described in Section 4.2.2), and finally, CLAHE [72] is performed on a grid 8×8 with clip limit 1. The embedding network is finetuned for 40 epochs, where each training epoch has 2000 iterations. In each iteration, 7 images are provided: *anchor*, *positive*, and 5 *negatives*, where for each anchor, negatives are mined from different clusters (different 3D reconstructions) such that the distance of anchor descriptor and positive descriptor is minimal. For clarity, *anchor* means anchor image, terms *anchor translation*, and *translation* mean to perform image–to–image

[2] https://github.com/sniklaus/pytorch-hed

translation with the anchor during the finetuning.

Diverse Anchors Mining

In the prior work of [43, 51], positive pairs are selected at random from all possible positive pairs. This could sometimes lead to a repeated selection of anchors with the same or similar scene and viewpoint causing lower data diversity within the epoch iterations. This issue is fixed with diverse anchors selected from a random pool of anchors, specifically, 2000 anchors are selected from the pool of size 10000. The selection process of the anchors from the pool is the following: the first anchor is selected at random; then before the next anchor is selected, the remainder of the pool is sorted by minimal L2 distance of descriptors extracted from the currently finetuned embedding network; and then the next anchor is selected from the ordered pool between 20th and 80th percentile. The reasoning behind dropping the first 20% is to increase the diversity of anchors and dropping the last 20% is to prevent the selection of anchors with outlying descriptors.

Data Augmentation

During the CNN finetuning, the generator trained in the previous step is used for the anchor translation from the source into the target domain (day→night, or photo→edges) as a form of data augmentation. Whenever an anchor would be inputted into the embedding network, it is translated with probability 0.25. The generator is not finetuned nor updated during the training. In negatives mining before each epoch, the anchor translation is performed before the mining.

4.3 Sketch Recognition

In the first step, a generator needed for image outlines synthesis is trained. In the second step, the downstream task of sketch classification is trained, where the generator is used with an edge detector to provide the training data to the classifier.[3]

4.3.1 Thin-pix2pix Generator

The architecture of Thin-pix2pix is composed of ResNet generator [98, 57], PatchGAN discriminator [56], and HED edge detector [73]. The optimization step is almost the same as in the pix2pix training [56]. The main difference is that the input image comes through the HED detector and the extracted edge maps are then put into the generator. In contrast with pix2pix training, the ResNet generator [57] is used instead of the U-Net generator [101, 56], $\lambda = 10$ instead of 100, and the generator and the discriminator are initialized with

[3]Sketch recognition codebase and its implementation progress is available online at Github at https://github.com/8ToThePowerOfMol/im2rbte

He initialization [25]. HED detector is initialized with pretrained weights the same way as in HED^{N}GAN and weights of the detector remain unchanged during the training. Thin-pix2pix is trained only for 20 epochs with a constant learning rate of 0.0002.

4.3.2 Sketch Classification

The sketch classifier is trained similarly to the prior work of [32]. The rBTE [32] domain is constructed from a natural image in 5 steps:

1. edge detection,
2. geometric augmentations,
3. non–maxima–suppression thinning,
4. hysteresis thresholding, and
5. large connected–components preservation.

In the Thin-pix2pix method, the first two steps remain because the generator is learned to thin edge maps obtained from the HED detector [73], and geometric augmentations are standard data augmentation techniques used in many machine learning applications. After these two steps, outlines obtained with the generator are used to train the classifier. To compare Thin-pix2pix more precisely with the rBTE baseline, edge maps are provided in two options: first is using HED detector alone and the second is using three edge detectors together, specifically Structured Edges (SE) [102], HED [73], and Bi–Directional Cascade Network (BDCN) [103]. Then, the sketch classifier is trained for 30 training epochs, where one epoch has the number of iterations equal to the size of the training dataset.

Thin-pix2pix generator is unimodal, but rBTE is multimodal in the image translation sense because the thresholding approach is chosen randomly from a candidate pool of 5 thresholding methods. To better compare the influence of randomization in the thresholding, thresholding is only performed with Otsu method [104]. In the experiments, this kind of domain construction is denoted as BTE.

Chapter 5

Results

5.1 Generator Training

The first Section 5.1.1 shows a comparison of different generator model training statistics, the second Section 5.1.2 aims to show the relation between the generator training and image retrieval performance, Section 5.1.3 shows quantitative results of trained generators, and the last Section 5.1.4 shows comparison HED and new HED$^{\mathrm{N}}$ edge detectors.

5.1.1 Architecture Comparison

Vanilla generators, such as CycleGAN [57], CUT [66], and DRIT [59], together with newly introduced HEDGAN and HED$^{\mathrm{N}}$GAN are compared in terms of training time, convergence, and the number of trainable parameters.

More technically, all models were trained with NVIDIA Tesla P100 16GB GPU on the same machine. The experiments on the server were not strictly isolated, which can affect epoch time measurement.

Results

Generator training comparison in Table 5.1 shows HEDGAN and HED$^{\mathrm{N}}$GAN require an order of magnitude less time to train compared to CycleGAN and DRIT.

Looking closer to single epoch estimated time, DRIT requires less time in contrast with CycleGAN, because in their original implementation[1], generators and other discriminators are updated every third iteration, and therefore, DRIT requires 300 training epochs instead of 100 epochs. CUT, HEDGAN, HED$^{\mathrm{N}}$GAN train only one generator resulting in lower epoch time.

Discussion

Concerning the total number of training epochs, while CycleGAN and DRIT utilize cycle consistency, which is inaccurate at the start of the training,

[1] Line 47 in https://github.com/HsinYingLee/DRIT/blob/master/src/train.py

Training	Train (h)	Epoch (h)	Epochs	Params (M)
DRIT	196	0:39	300	75.5
CycleGAN	102	1:01	100	51.0
CUT	35	0:42	50	26.0
HEDGAN	18	0:21	50	25.5
HED$^{\mathrm{N}}$GAN	19	0:23	50	40.2

Table 5.1: Generators training time and parameters comparison. In the first two columns, training times are reported in hours. In the third column, the total number of training epochs necessary for the model to converge is reported. Each epoch consists of 10000 optimization iterations. In the last column, the number of trainable parameters in millions is reported.

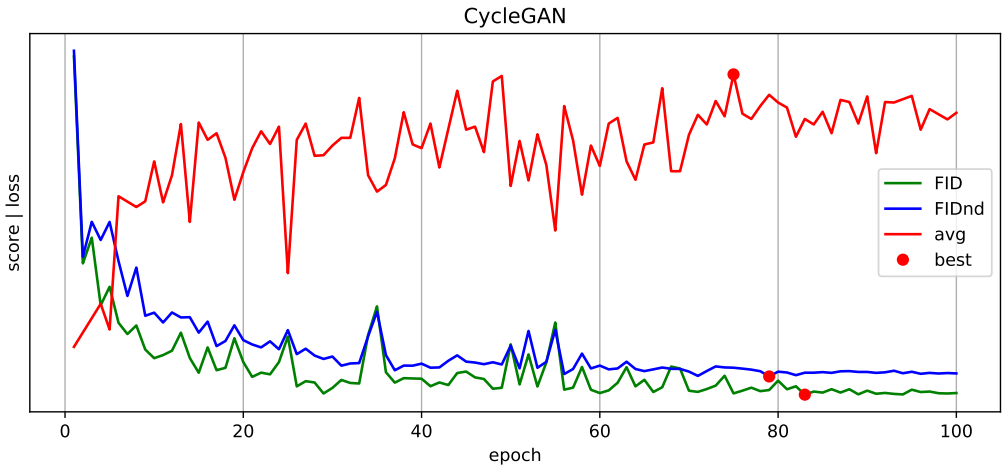


Figure 5.1: The generator and embedding network evaluation during generator training. Mean average precision (mAP) of average of Tokyo 24/7, $\mathcal{R}$Oxf, and $\mathcal{R}$Par (red) are compared with FID [18, 17] (green) and FIDnd (described in Section 5.1.2) (blue). An epoch, where the evaluation is the best is emphasized with a red dot. All measures were linearly scaled for easier visual comparison. No exponential moving average was used for smoothing.

HEDGAN and HED$^{\mathrm{N}}$GAN preserve the structure of images with edge consistency, which provides more feedback to the generator since the start of the training. Therefore, HEDGAN and HED$^{\mathrm{N}}$GAN need twice as many or fewer epochs than CycleGAN and DRIT. CUT also needs less number of epochs, which is studied in more detail by Park *et al.* [66].

5.1.2 Optimization Towards Retrieval Performance

It is well–known that GAN training is unstable and difficult process [16]. Therefore, it would be beneficial to be able to select the best model obtained during the GAN training, because the last iteration not necessarily leaves the best–performing model for the desired task of data augmentation in metric learning. For simplicity, this Section focuses only on day→night.

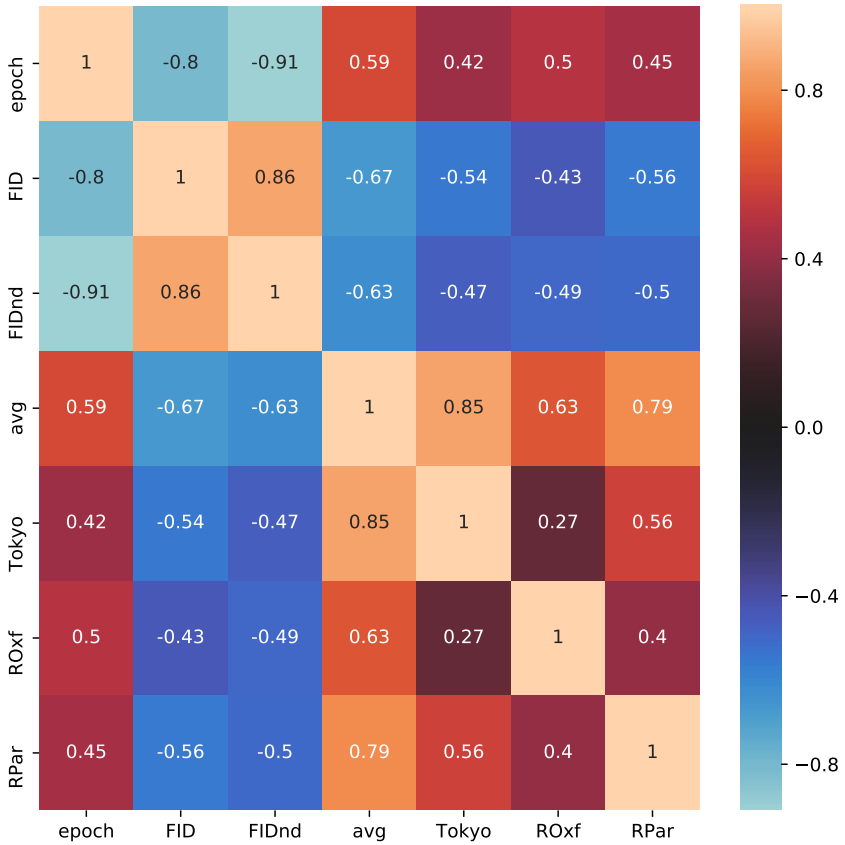


Figure 5.2: Heatmap of the correlation matrix of generator and embedding evaluation measures obtained during generator training. The correlation between each pair is measured with Spearman's rank correlation coefficients [105]. FID [18, 17] and FIDnd (described in Section 5.1.2) are evaluation measures of the generator model, while Tokyo, $\mathcal{R}$Oxf, and $\mathcal{R}$Par are evaluation measures of the embedding model. Avg is the average of Tokyo, $\mathcal{R}$Oxf, and $\mathcal{R}$Par. The correlations were measured on the same training of CycleGAN as in 5.1.

An extensive evaluation of the best–performing model – CycleGAN trained on SfM – learning process was conducted. After each epoch, average mAP of Tokyo 24/7, $\mathcal{R}$Oxf, $\mathcal{R}$Par, FID [18, 17], and FIDnd is measured between positive pairs.

FID. The most standard way how to measure FID is between the input data distribution and output data distribution generated directly from the input. A set of real day images is translated into fake night images, and then FID is measured between these two sets, using Inception-V3 [8] embedding network.

FIDnd. The second option how to measure FID is between real night and fake night image distributions. To measure FID between the night–night pairs, pixel–aligned day–night pairs would be required. I have not found such pairs in the landmark domain. Therefore, SfM N/D dataset is used for this purpose, where from night–day positive pairs, the day image is translated

from day to night and the distance is measured between real–night and fake–night pairs. Image pairs do not have necessarily the same structure, but depict the same object. Moreover, both image distributions are night, for which the Inception-V3 embedding network [8] could be imprecise since it was not trained with enough night images. Therefore, Inception-V3 network is replaced with finetuned VGG16 on SfM N/D for metric learning from [51]. Also, another way how to measure the distance would be simply to take the average L2 distance between all real–night and fake–night pairs. From my experiments, this average L2 distance correlates with FIDnd.

After all the measures are obtained, a matrix of non–parametric Spearman's rank correlation coefficients [105] are calculated between all measured variables. These calculations conclude, which of FID, FIDnd, or the last epoch should be used to pick the best generator model.

Results

Figure 5.1 shows retrieval performance is very varying during the generator training. After approximately 10 epochs, the generator is enough trained to operate in day→night data augmentation, increasing retrieval performance. Whenever FID and FIDnd increase, retrieval performance drops (see the spikes in Figure 5.1), which may be beneficial in the situation, when the generator in its last k epochs would be considered to be picked. However, none of FID or FIDnd can indicate, which epoch is the best for the final metric learning.

Figure 5.2 indicates, FID would be the best criterion to pick for the best epoch selection. However, correlation ranks in their absolute values do not differ much from the current epoch number correlation rank.

Discussion

The large deviations in retrieval performance per generator epoch can be explained by the unstable nature of GAN training. The best way how to obtain the best performance would be to perform metric learning per generator's current weights generator, which is costly, especially with higher prices of electricity in late 2022. Different GAN architectures can cause different deviations and their FID and FIDnd can have different correlations with final retrieval performance during the generator training. To provide a more general conclusion about whether FID, FIDnd, or the last epoch correlates the best, these experiments would be better to conduct with other GAN models than just CycleGAN. To remain consistent in other experiments, the last epoch of the generator training is always used.

Figure 5.3: Examples of day→night translation on SfM dataset [43] with different generators. Source image (left column) is translated with (from left to right): HED^{N}GAN, CycleGAN [57], DRIT [59], and CUT. All models are trained on the SfM dataset.

5.1.3 Inference Comparison

Example outputs for all generative models are provided. Figure 5.3 shows day→night image outputs on the SfM training dataset. Figure 5.4 shows outputs obtained from the learned thinning on different kinds of input data. Appendix A contains image outputs compared in more detail.

Results

Almost all outputted images can be distinguished from a real image by a human observer. Nonetheless, this does not necessarily decrease the final retrieval performance.

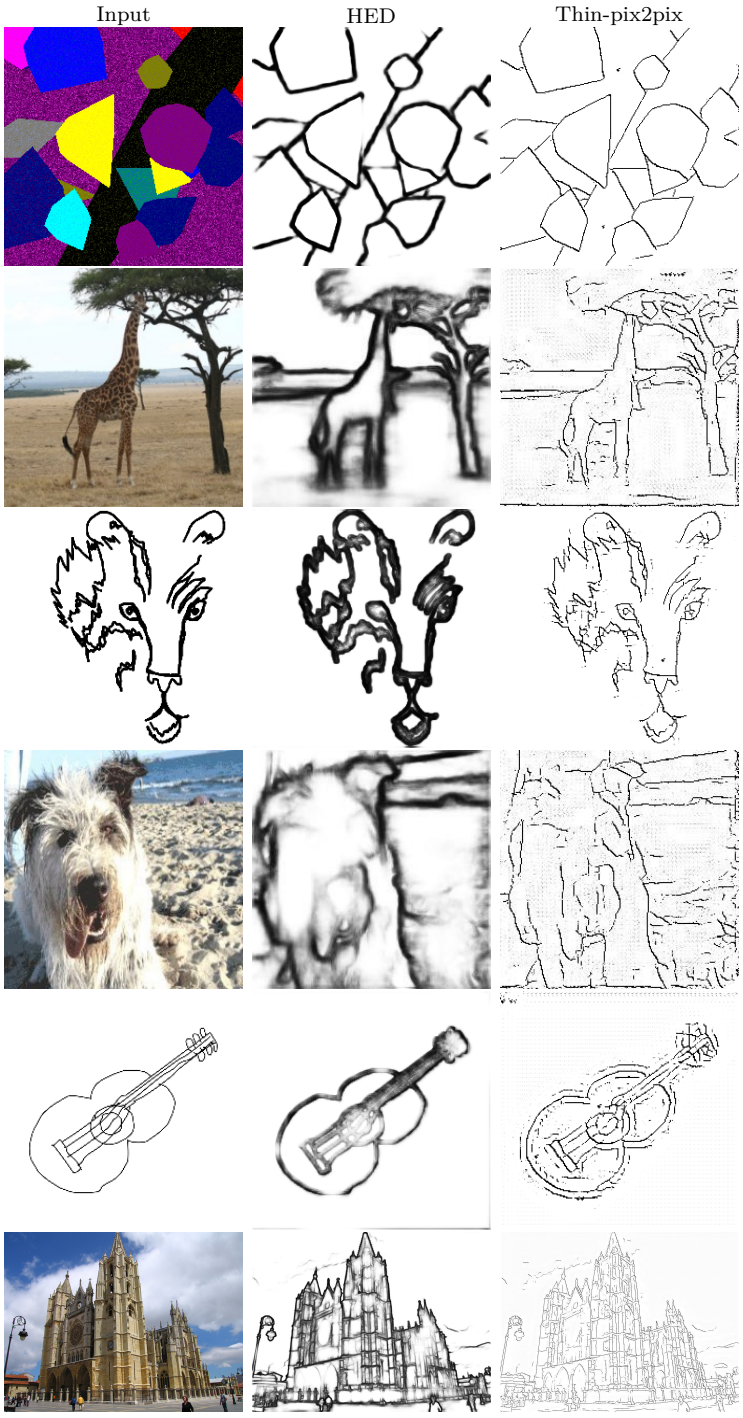


Figure 5.4: Examples of learned thinning among different datasets. From the input image (left column) edge maps (middle column) are obtained with HED [73] edge detector followed by the proposed Thin-pix2pix, which is learned to thin edge maps into outlines (last column). Rows consist of images of different datasets: generated training polygons, Sketchy [97] with an image input, Sketchy with a sketch input, PACS [34] with an image input, PACS with a sketch input, and SfM [92] with a large image input. In the 3rd and 5th row, outlines were obtained from the input sketch instead of the HED edge map. Edge maps and outlines are inverted for better visibility on white paper.

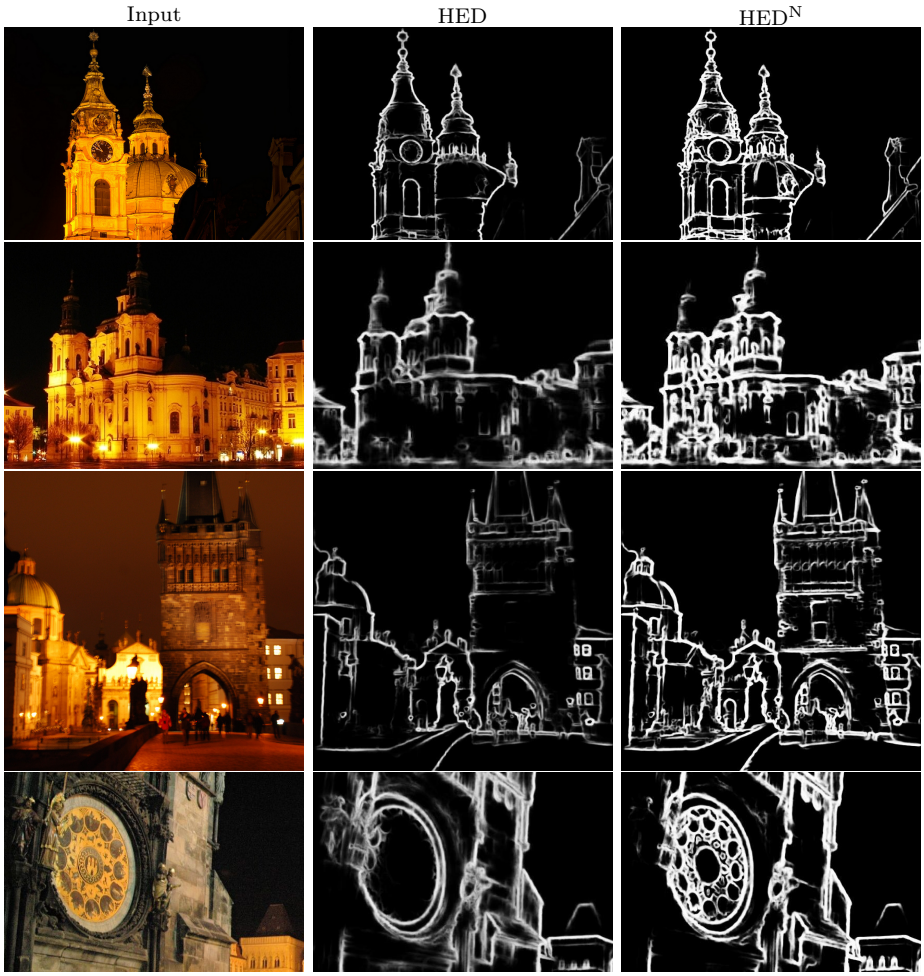


Figure 5.5: Examples of extracted edges from HED [73], and HED^{N} edge detectors. From the input image (left column), the edge maps were extracted with HED (middle column), and with HED^{N} (right column), which was trained jointly with the generator and the discriminator in HED^{N}GAN training. Input images were selected from SfM dataset [43].

5.1.4 Edge Detection Comparison

In Figure 5.5, comparison between selected night images edge extraction with HED [73] and HED^{N}, which is finetuned jointly with the generator in HED^{N}GAN architecture.

5.2 Day–Night Image Retrieval

In the first Section 5.2.1, overall retrieval results are summarized. The last Section 5.2.2 contains a comparison of different night image data augmentations. In Appendix B, an ablation that shows how diverse anchors and CLAHE influence retrieval training can be found together with a second ablation that compares the performance of the new HED^{N} detector on edge–based image retrieval with HED detector [73].

5.2.1 Concluding Results

The effect of data augmentation is compared to baselines:

- GeM [43], vanilla method with annotations obtained automatically from 3D reconstructions,
- CLAHE [51] uses the same data as GeM and each image is photometrically normalized,
- GeM N/D [51] uses SfM N/D with day–night training positive pairs,
- CLAHE N/D [51] uses SfM N/D with CLAHE normalization,
- CIConv [76].

In the tables, whenever a method is reported with citation reference, its performance values were obtained from the results the original authors report, otherwise, baseline models were evaluated. For CIConv [76], their trained model with ResNet-101 backbone was used to evaluate it on $\mathcal{R}$Oxf and $\mathcal{R}$Par. The SfM N/D dataset has a 0.25 ratio of night anchors in all experiments. Details on the data augmentation are covered in Section 4.2.2. Methods examined in this thesis are referred to the name of the generator used in the night data synthesis (CycleGAN, HED^{N}GAN). When CLAHE photometric normalization [72] is used, it is marked with $\mathcal{C}$; when diverse anchors mining (described in Section 4.2.2) is used, it is marked with $\mathcal{D}$.

Results

In Table 5.2, results show data augmentation with image–to–image translation clearly surpasses all baselines. Notice, metric learning using paired day–night training pairs were outperformed by a method, which does not need day–night pairs, it only needs two sets of day and night training images.

Discussion

The reason why the data augmentation with synthetic images beats the method using true day–night pairs from SfM N/D can be explained by the diversity of night training data: real day images of SfM dataset are diverse, while real night images are not, and therefore, fake night images are diverse since they are synthesized from diverse day images.

VGG-16 backbone

Method	Avg	Tokyo	$\mathcal{R}$Oxf	$\mathcal{R}$Par
GeM [43]	69.9	79.4	**60.9**	69.3
GeM N/D [51] !	71.1	83.5	60.0	69.8
CIConv [76]	-	83.3	-	-
CLAHE $\mathcal{C}$ [51]	71.6	84.1	60.8	69.8
CLAHE N/D $\mathcal{C}$ [51] !	72.4	87.0	60.2	70.0
SobelGAN $\mathcal{CD}$	**73.5**	**89.3**	**60.9**	**70.5**
HED$^{\text{N}}$GAN $\mathcal{CD}$	73.4	88.9	**61.1**	70.3
CycleGAN $\mathcal{CD}$	**74.0**	**90.2**	60.7	**71.0**

ResNet-18 backbone

Method	Avg	Tokyo	$\mathcal{R}$Oxf	$\mathcal{R}$Par
GeM	65.4	76.1	50.5	69.5
GeM N/D !	67.0	79.6	51.3	**70.0**
CLAHE $\mathcal{C}$	67.3	80.6	52.6	68.6
CLAHE N/D $\mathcal{C}$!	68.0	82.5	52.5	69.0
HED$^{\text{N}}$GAN $\mathcal{CD}$ (ours)	**69.6**	**85.1**	**53.6**	**70.0**
CycleGAN $\mathcal{CD}$ (ours)	**69.7**	**84.4**	**55.1**	69.6

ResNet-50 backbone

Method	Avg	Tokyo	$\mathcal{R}$Oxf	$\mathcal{R}$Par
GeM	74.6	85.4	63.4	**75.1**
GeM N/D !	75.7	88.3	63.1	**75.6**
CLAHE $\mathcal{C}$	74.7	87.4	62.5	74.2
CLAHE N/D $\mathcal{C}$!	75.3	89.0	62.3	74.5
HED$^{\text{N}}$GAN $\mathcal{CD}$ (ours)	**77.0**	**91.7**	**64.4**	74.9
CycleGAN $\mathcal{CD}$ (ours)	**77.0**	**92.3**	**64.0**	74.7

ResNet-101 backbone

Method	Avg	Tokyo	$\mathcal{R}$Oxf	$\mathcal{R}$Par
GeM [43]	75.7	85.0	65.3	76.7
GeM N/D !	77.0	88.6	65.7	76.8
CIConv [76]	75.0	88.3	62.0	74.7
CLAHE $\mathcal{C}$	76.9	88.1	66.1	76.6
CLAHE N/D $\mathcal{C}$!	77.4	89.5	66.1	76.5
CUT $\mathcal{CD}$	77.9	90.2	65.7	**77.7**
CUT (tuned) $\mathcal{CD}$	78.0	90.9	65.7	**77.3**
SobelGAN $\mathcal{CD}$ (ours)	78.1	91.5	**66.3**	76.6
HED$^{\text{N}}$GAN $\mathcal{CD}$ (ours)	**78.4**	**92.2**	**66.3**	76.6
CycleGAN $\mathcal{CD}$ (ours)	**78.4**	**92.0**	**66.8**	76.4

Table 5.2: Day–night image retrieval performance comparison. All scores are reported in mean average precision (mAP). In the second column, the average of Tokyo 24/7, $\mathcal{R}$Oxf, and $\mathcal{R}$Par is reported for easier comparison. Methods marked by ! use day–night image pairs. Methods marked with a reference were not trained, their results were obtained from their reference. The best score for each backbone is in red bold, second best is in bold.

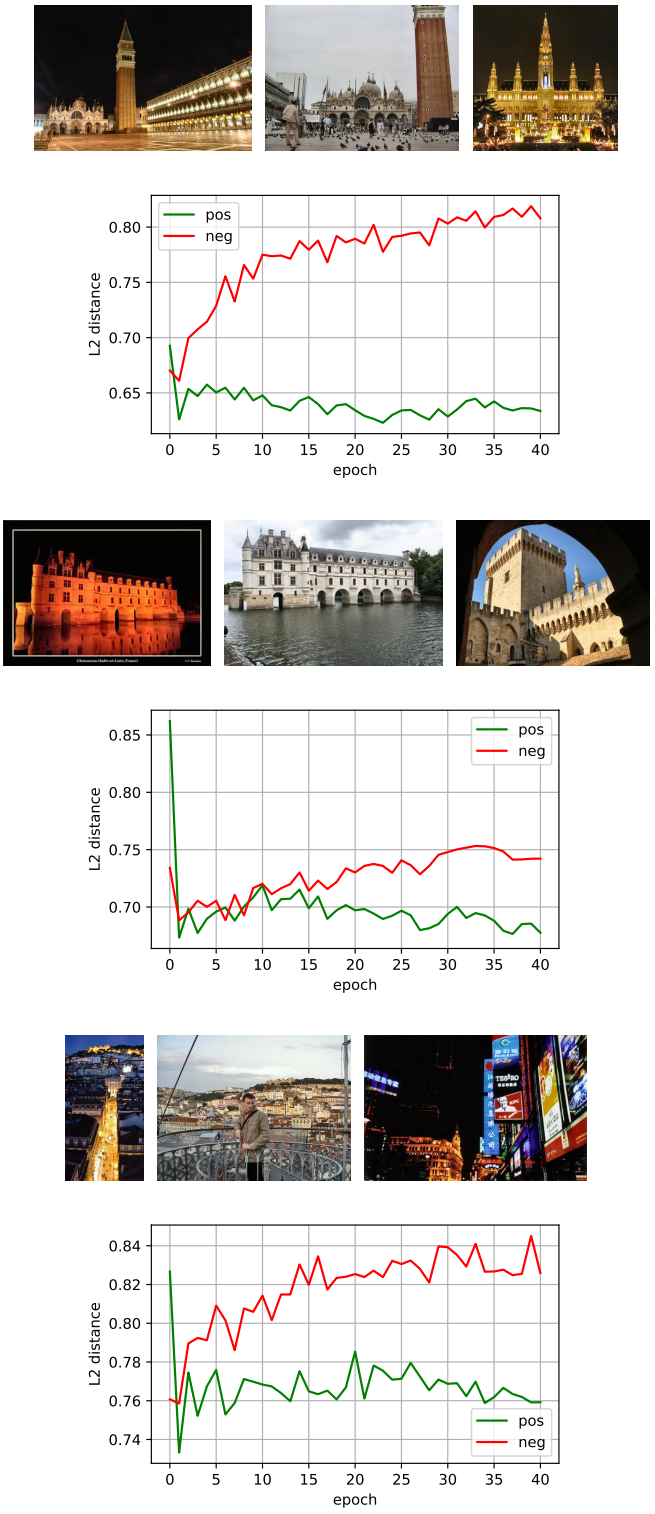


Figure 5.6: Examples of descriptor distances of positives and negatives during metric learning. Image triplets are (from left to right): night anchor, day positive, and mined negative. Under each triplet, the L2 distance between anchor–positive (green), and anchor–negative (red) is plotted against each training epoch. Epoch 0 corresponds to the embedding network initialization.

5.2.2 Impact of Data Augmentation

The ratio of synthesized training images and the choice of the generator influence metric learning. Two ratios: 25% and 50% of night data are evaluated. Five different day→night generators are compared:

- CycleGAN [57], first popular and simple GAN for the task unpaired image–to–image translation,
- DRIT [59], generator trained for the task of multimodal unpaired image–to–image translation,
- CUT [66], lightweight version of CycleGAN, which replaced cycle consistency with encoder feature map contrastive loss,
- HEDGAN, generator trained similarly to CycleGAN, where cycle consistency is replaced by edge consistency,
- HED$^{\mathrm{N}}$GAN, edge detector is trained jointly with generator and discriminator, so night edge detection is improved during the training.

Also, pairs of best–performing generators (CycleGAN and HED$^{\mathrm{N}}$GAN) with a ratio of 1:1 were evaluated. All experiments for the embedding network training were conducted with VGG-16 backbone.

Results

For the amount of night data, Table 5.3 shows using 50% instead of 25% has a minor performance increase in day–night retrieval (Tokyo 24/7), otherwise, this does not make a significant difference. To stay comparable with [51], 25% of night data is still used in other experiments.

Concerning the choice of the generator, Table 5.3 shows CycleGAN is the best choice of the generator. Surprisingly, the multimodal unpaired image translation method (DRIT) did not end up in the first place. Nonetheless, all generators are outperforming training with the real day–night image pairs, while the differences among the generator choices are little.

Discussion

Better performance achieved with synthetic night–day pairs compared to real night–day pairs can be explained from the observation that photo–realistic appearance of fake night images is not crucial for retrieval performance, and thus taking the most powerful generator nor adding additional true night images is not necessarily the best option how to increase night retrieval performance. Similar observation can be confirmed by [88], who managed to learn a network for visual localization merely on synthesized images.

Lastly, despite similar scores achieved with GAN–based generators, their training time differs greatly, which was shown in Section 5.1.1.

Night Data	Avg	Tokyo	$\mathcal{R}$Oxf	$\mathcal{R}$Par
DRIT $\mathcal{CD}$	73.5	90.2	59.8	70.5
CUT $\mathcal{CD}$	73.0	87.7	60.3	**71.1**
CUT (tuned) $\mathcal{CD}$	73.4	88.6	60.8	70.8
SobelGAN $\mathcal{CD}$	73.5	89.3	60.9	70.5
HEDGAN $\mathcal{CD}$	73.2	88.1	61.0	70.5
HED$^{\text{N}}$GAN $\mathcal{CD}$	73.4	88.9	**61.1**	70.3
HED$^{\text{N}}$GAN 50% $\mathcal{CD}$	73.4	90.3	60.0	70.0
CycleGAN $\mathcal{CD}$	**74.0**	90.2	60.7	71.0
CycleGAN 50% $\mathcal{CD}$	73.9	**91.4**	60.0	70.4
CycleGAN Aachen $\mathcal{CD}$	73.8	89.9	60.7	70.8
CycleGAN + N/D $\mathcal{CD}$!	73.5	88.6	60.8	**71.1**
CycleGAN + N/D 50% $\mathcal{CD}$!	73.9	90.1	**61.1**	70.6
HED$^{\text{N}}$GAN + N/D $\mathcal{CD}$!	73.3	87.9	**61.1**	70.8
HED$^{\text{N}}$GAN + N/D 50% $\mathcal{CD}$!	73.6	89.1	**61.1**	70.6
HED$^{\text{N}}$GAN + HEDGAN $\mathcal{CD}$	73.4	88.0	60.7	70.6
HED$^{\text{N}}$GAN + HEDGAN 50% $\mathcal{CD}$	73.4	90.0	60.5	69.8
CycleGAN + CycleGAN Aachen $\mathcal{CD}$	74.0	90.4	60.5	**71.1**
CycleGAN + CycleGAN Aachen 50% $\mathcal{CD}$	74.0	**91.4**	60.3	70.4
HED$^{\text{N}}$GAN + CycleGAN $\mathcal{CD}$	74.0	90.0	61.0	70.9
HED$^{\text{N}}$GAN + CycleGAN 50% $\mathcal{CD}$	**74.1**	**91.4**	60.5	70.5

Table 5.3: The impact of retrieval training data comparison. In the top block, various generator architectures CycleGAN [57], CUT [66] (original and tuned version), DRIT [59], HEDGAN, and HED$^{\text{N}}$GAN (trained HED) are compared. CycleGAN was also trained on Aachen dataset (CycleGAN Aachen) instead of SfM dataset. In the bottom block, CycleGAN, or HED$^{\text{N}}$GAN is combined with the SfM N/D dataset or with a different generator with a ratio 1:1, and with 25% (default in experiments), or 50% of night images in the training data. In all experiments, VGG-16 backbone is used. Methods marked by ! use day–night image pairs. The best score for each block is in bold.

5.3 Sketch Recognition

In this Section, the effectiveness of natural image into outlines translation used to provide training data for sketch recognition with no training sketches is evaluated.

5.3.1 Sketch Classification without Sketches

The prior work [32] performed an ablation on Sketchy dataset [97] and compared with different domain generalization methods on PACS dataset [34]. To provide as much information as possible for the comparison, methods on both Sketchy and PACS datasets are evaluated. Evaluations are performed the same way as in [32], specifically for PACS (Sketchy) dataset, ResNet-18 (ResNet-101) [7] backbone is used and the reported classification accuracy is averaged over 20 (5) randomized runs. Since the relative sketch size can vary with respect to the image, single–scale and multi–scale evaluations are performed, where the sketch in its original size is taken in the single–scale case, while in the multi–scale case, the sketch is cropped to its bounding box, padded to 1:1 aspect ratio and downscaled to 90%, 65%, and 45% of the 224 input size [32].

Results

Table 5.4 shows, the rBTE [32] outperforms the proposed Thin-pix2pix by a larger margin in both single–scale and multi–scale evaluations, and with both datasets. BTE outperforms Thin-pix2pix by a smaller margin. The same observation holds when both edge maps of 3 detectors are used together as well as only with edge maps from HED [73] detector. Surprisingly, the performance gap between Thin-pix2pix and BTE is smaller, when multiple edge maps are used.

Discussion

There are three reasons why Thin-pix2pix performs worse than rBTE [32]. First, the performance gap between Thin-pix2pix and BTE using only HED [73] edge maps indicates that the trained Thin-pix2pix generator has shortcomings to approximate thin edges as accurately as a handcrafted method. In Figure 5.4, it can be observed that Thin-pix2pix tries to fill gaps between the edge maps because during the Thin-pix2pix training, HED missed some edges between polygons with similar color, and therefore, the generator tries to fill the outlines up to the ground–truth outlines (Figure 5.4, first row). As a result, the generator can hallucinate edges or produce artifacts (Figure 5.4, pre–last row). Second, during the training, rBTE can consist of different thinned edges per a single input edge map, which is not the case of Thin-pix2pix, which outputs the same unimodal outlines per input edge map. Using architecture for multimodal image translation, such as BicycleGAN [61]

Single–scale; SE, HED, and BDCN

Method	PACS	Sketchy
rBTE	70.7 ± 2.0	49.7 ± 0.5
BTE	66.3 ± 2.5	47.2 ± 0.5
Thin-pix2pix	62.8 ± 2.2	43.7 ± 0.4

Multi–scale; SE, HED, and BDCN

Method	PACS	Sketchy
rBTE	70.7 ± 2.0	52.3 ± 0.5
BTE	65.1 ± 2.9	50.5 ± 0.7
Thin-pix2pix	60.0 ± 3.3	45.5 ± 0.6

Single–scale; HED

Method	PACS	Sketchy
rBTE	66.8 ± 2.1	47.9 ± 0.6
BTE	65.2 ± 2.6	46.3 ± 0.6
Thin-pix2pix	57.2 ± 3.0	41.0 ± 0.4

Multi–scale; HED

Method	PACS	Sketchy
rBTE	66.7 ± 2.6	50.9 ± 0.4
BTE	64.2 ± 3.3	49.1 ± 0.5
Thin-pix2pix	52.7 ± 4.6	42.9 ± 7.6

Table 5.4: Sketch classification performance comparison. All scores are reported in accuracy. In the first two blocks, Structured Edges (SE) [102], Hollistically–Nested Edge Detection (HED) [73], and Bi–Directional Cascade Network (BDCN) [103] all together provide edge maps, while in the last two blocks, only HED provides edge maps. For rBTE, results for Sketchy dataset were obtained from the ablations of [32].

together with the removal of small parts consistently in the input and outlines during the generator training would help. Third, Thin-pix2pix was trained only on HED [73] edge maps, but in the classification, different types of edge maps that Thin-pix2pix never seen are provided.

Chapter 6

Conclusions

In this thesis, the challenges of the day–night, and photo–sketch appearance changes were addressed with image synthesis to approximate these shifts. To achieve this, GAN–based generators were studied and widely employed for image synthesis of the scarce visual domain in two tasks. In day–night image retrieval task, synthetically generated night images from day images were used as a form of data augmentation in metric learning, which relaxes the necessity to obtain training day–night positive image pairs. Also, an ablation with 4 different GAN architectures was conducted, including the new HED$^{\mathrm{N}}$GAN architecture, which uses edge consistency to preserve the image content. In sketch recognition, the task considered that no training sketch images were available at all, which is solved by natural image transformation into image outlines using an edge detector together with a trained Thin-pix2pix generator. This generator aids the downstream task by thinning edge maps into outlines. Thin-pix2pix training does not require gathering any additional data.

In the two tasks, GAN–based image synthesis improved the performance of day–night image retrieval, but in sketch recognition task, it performed worse than the handcrafted baseline. In day–night image retrieval, experiment results demonstrated the effectiveness of this approach with three main conclusions:

1. A larger diversity of synthesized night images is more powerful than smaller diversity of real night images.
2. Day–night retrieval performance with synthesized night images does not improve, when real night images are added.
3. HED$^{\mathrm{N}}$ edge detector trained with HED$^{\mathrm{N}}$GAN has superior performance over the HED detector, which was shown quantitatively and qualitatively.

Experiment results also showed the choice of the GAN generator is negligible for the final retrieval performance.

Bibliography

[1] Filip Radenovic, Johannes L Schonberger, Dinghuang Ji, Jan-Michael Frahm, Ondrej Chum, and Jiri Matas. From dusk till dawn: Modeling in the dark. In *CVPR*, 2016.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[3] Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, 1989.

[4] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *NIPS*. Curran Associates, Inc., 2012.

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[8] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*. Curran Associates, Inc., 2014.

[10] Ian J Goodfellow. On distinguishability criteria for estimating generative models. *ICLR*, 2015.

[11] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR*, 2016.

[12] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *NeurIPS*, 2016.

[13] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *ICCV*, 2017.

[14] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[15] Albert Mohwald. Data augmentation by image-to-image translation for image retrieval. Bachelor's thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, 2020.

[16] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016.

[17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

[18] Maurice Fréchet. Sur la distance de deux lois de probabilité. *COMPTES RENDUS HEBDOMADAIRES DES SEANCES DE L ACADEMIE DES SCIENCES*, 1957.

[19] Leonid Nisonovich Vaserstein. Markov processes over denumerable products of spaces, describing large systems of automata. *Problemy Peredachi Informatsii*, 1969.

[20] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *ICLR*, 2018.

[21] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *ICLR*, 2018.

[22] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.

[23] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 2000.

[24] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning (ICML)*. PMLR, 2013.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[26] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010.

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. PMLR, 2015.

[28] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[29] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *NeurIPS*, 2014.

[30] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 2016.

[31] Gabriela Csurka. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017.

[32] Nikos Efthymiadis, Giorgos Tolias, and Ondřej Chum. Edge augmentation for large-scale sketch recognition without sketches. In *2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE, 2022.

[33] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *NeurIPS*, 2006.

[34] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, 2017.

[35] Wengang Zhou, Houqiang Li, and Qi Tian. Recent advance in content-based image retrieval: A literature survey. *arXiv preprint arXiv:1706.06064*, 2017.

[36] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[37] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *ECCV*. Prague, 2004.

[38] Anastasiia Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. *NeurIPS*, 2017.

[39] Yurun Tian, Xin Yu, Bin Fan, Fuchao Wu, Huub Heijnen, and Vassileios Balntas. Sosnet: Second order similarity regularization for local descriptor learning. In *CVPR*, 2019.

[40] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *IEEE*, 2010.

[41] Giorgos Tolias, Yannis Avrithis, and Hervé Jégou. To aggregate or not to aggregate: Selective match kernels for image search. In *ICCV*, 2013.

[42] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, 2016.

[43] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *IEEE TPAMI*, 2018.

[44] Artem Babenko and Victor Lempitsky. Aggregating local deep features for image retrieval. In *ICCV*, 2015.

[45] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of cnn activations. In *ICLR*, 2015.

[46] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

[47] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[48] Filip Radenović, Giorgos Tolias, and Ondřej Chum. CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples. In *ECCV*, 2016.

[49] Mu Zhu. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2004.

[50] Andrew Turpin and Falk Scholer. User performance versus precision measures for simple search tasks. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006.

[51] Tomas Jenicek and Ondrej Chum. No fear of the dark: Image retrieval under varying illumination conditions. In *ICCV*, 2019.

[52] Filip Radenovic, Giorgos Tolias, and Ondřej Chum. Deep shape matching. In *ECCV*, 2018.

[53] Yansheng Li, Jiayi Ma, and Yongjun Zhang. Image retrieval from remote sensing big data: A survey. *Information Fusion*, 2021.

[54] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[55] Giorgos Tolias, Filip Radenovic, and Ondrej Chum. Targeted mismatch adversarial attack: Query with a flower to retrieve the tower. In *ICCV*, 2019.

[56] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[57] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[58] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Kumar Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *ECCV*, 2018.

[59] Hsin-Ying Lee, Hung-Yu Tseng, Qi Mao, Jia-Bin Huang, Yu-Ding Lu, Maneesh Singh, and Ming-Hsuan Yang. Drit++: Diverse image-to-image translation via disentangled representations. *IJCV*, 2020.

[60] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*. PMLR, 2017.

[61] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NeurIPS*, 2017.

[62] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NeurIPS*, 2017.

[63] Qi Mao, Hsin-Ying Lee, Hung-Yu Tseng, Siwei Ma, and Ming-Hsuan Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *CVPR*, 2019.

[64] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *CVPR*, 2017.

[65] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *International conference on machine learning (ICML)*. PMLR, 2017.

[66] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *ECCV*. Springer, 2020.

[67] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *NeurIPS*, 2018.

[68] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning (ICML)*. PMLR, 2020.

[69] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. 2004.

[70] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. In *IEEE TPAMI*, 2010.

[71] William K Pratt. *Digital image processing*. Wiley, 1978.

[72] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. Springer London, 2010.

[73] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *ICCV*, 2015.

[74] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*. IEEE, 2001.

[75] Filip Radenović, Giorgos Tolias, and Ondřej Chum. CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples. In *ECCV*, 2016.

[76] Attila Lengyel, Sourav Garg, Michael Milford, and Jan C. van Gemert. Zero-shot day-night domain adaptation with a physics prior. In *ICCV*, 2021.

[77] Yingxue Pang, Jianxin Lin, Tao Qin, and Zhibo Chen. Image-to-image translation: Methods and applications. *IEEE Transactions on Multimedia*, 2021.

[78] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *CVPR*, 2020.

[79] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018.

[80] Zhiqiang Shen, Mingyang Huang, Jianping Shi, Xiangyang Xue, and Thomas S. Huang. Towards instance-level image-to-image translation. In *CVPR*, 2019.

[81] Deblina Bhattacharjee, Seungryong Kim, Guillaume Vizier, and Mathieu Salzmann. Dunit: Detection-based unsupervised image-to-image translation. In *CVPR*, 2020.

[82] Peng Gao, Tian Tian, Linfeng Li, Jiayi Ma, and Jinwen Tian. Decyclegan: An object enhancement network for weak vehicle detection in satellite images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2021.

[83] Che-Tsung Lin, Yen-Yi Wu, Po-Hao Hsu, and Shang-Hong Lai. Multimodal structure-consistent image-to-image translation. In *AAAI*, 2020.

[84] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.

[85] Partha Ghosh, Mehdi SM Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. *arXiv preprint arXiv:1903.12436*, 2019.

[86] Vinicius F Arruda, Thiago M Paixão, Rodrigo F Berriel, Alberto F De Souza, Claudine Badue, Nicu Sebe, and Thiago Oliveira-Santos. Cross-domain car detection using unsupervised image-to-image translation: From day to night. In *IJCNN*, 2019.

[87] Asha Anoosheh, Torsten Sattler, Radu Timofte, Marc Pollefeys, and Luc Van Gool. Night-to-day image translation for retrieval-based localization. In *ICRA*, 2019.

[88] Markus S Mueller, Torsten Sattler, Marc Pollefeys, and Boris Jutzi. Image-to-image translation for enhanced feature matching, image retrieval and visual localization. *ISPRS*, 2019.

[89] Yihang Lou, Yan Bai, Jun Liu, Shiqi Wang, and Lingyu Duan. Veri-wild: A large dataset and a new method for vehicle re-identification in the wild. In *CVPR*, 2019.

[90] Simon Niklaus. A reimplementation of HED using PyTorch. https://github.com/sniklaus/pytorch-hed, 2018.

[91] Lingxi Xie, Richang Hong, Bo Zhang, and Qi Tian. Image classification and retrieval are one. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, 2015.

[92] Johannes L Schonberger, Filip Radenovic, Ondrej Chum, and Jan-Michael Frahm. From single image query to detailed 3d reconstruction. In *CVPR*, 2015.

[93] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl, and Tomas Pajdla. Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions. In *CVPR*, 2018.

[94] Torsten Sattler, Tobias Weyand, Bastian Leibe, and Leif Kobbelt. Image Retrieval for Image-Based Localization Revisited. In *BMVC*, 2012.

[95] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Revisiting Oxford and Paris: Large-scale image retrieval benchmarking. In *CVPR*, 2018.

[96] Akihiko Torii, Relja Arandjelović, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 place recognition by view synthesis. In *CVPR*, 2015.

[97] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (TOG)*, 2016.

[98] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*. Springer, 2016.

[99] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[100] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[101] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015.

[102] Piotr Dollár and C Lawrence Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013.

[103] Jianzhong He, Shiliang Zhang, Ming Yang, Yanhu Shan, and Tiejun Huang. Bi-directional cascade network for perceptual edge detection. In *CVPR*, 2019.

[104] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 1979.

[105] Charles Spearman. The proof and measurement of association between two things. 1961.

[106] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 2017.

Appendix A

Generative Model Outputs

Figure A.1, and Figure A.2 show day→night image outputs on the SfM training dataset compared among similar models in more detail. The effect of day→night and night→day translation on retrieval training and evaluation datasets is examined in Figure A.3, and Figure A.4.

In Figure A.4, since ToDayGAN [87] is trained only on RobotCar dataset [106], it is difficult for ToDayGAN to synthesize visually appealing images on different than only RobotCar dataset, although a hallucinated car appears in the middle of the image.

Figure A.1: Examples of day→night translation on SfM dataset [43] with similar generators. Source image (left column) is translated with (from left to right): CycleGAN [57] trained on the SfM dataset, CycleGAN [57] trained on the Aachen dataset, CUT with further hyperparameter tuning, and original CUT [66].

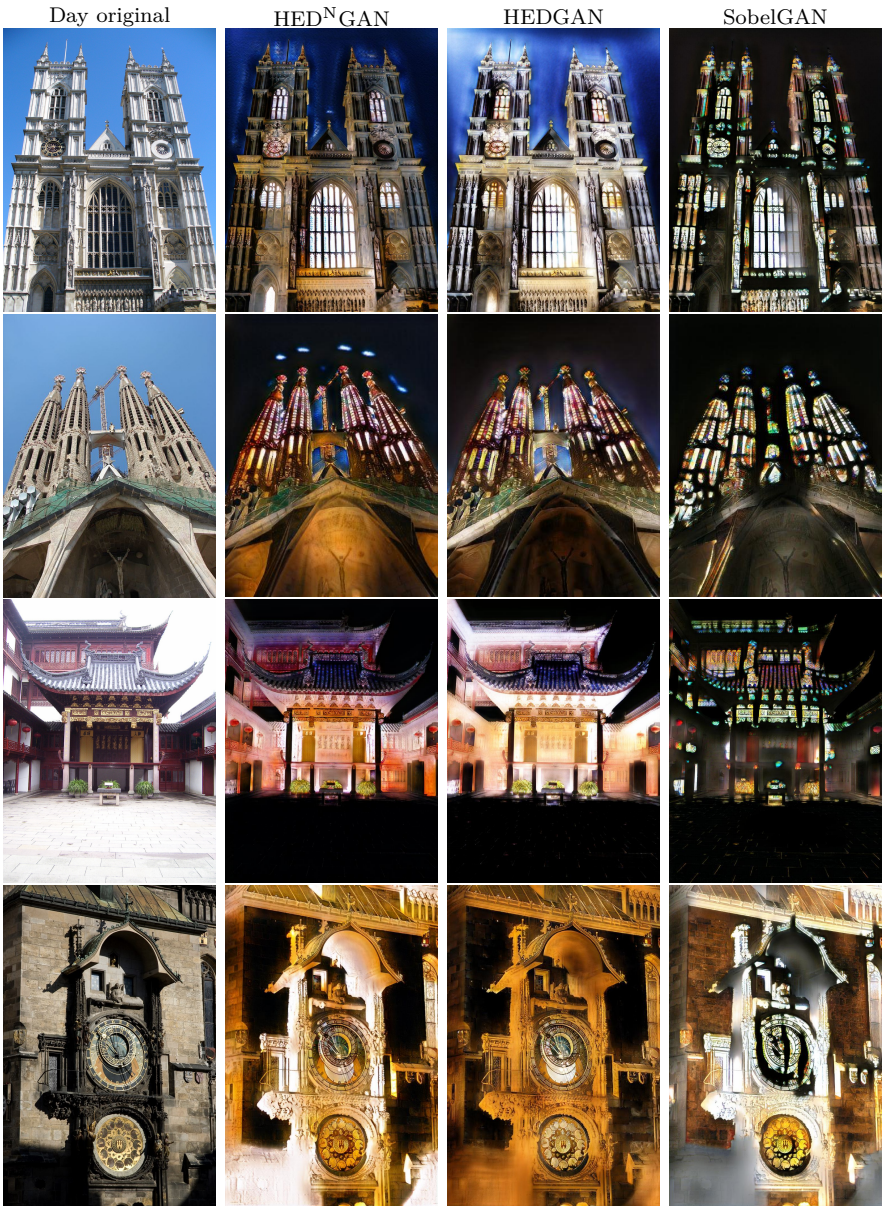


Figure A.2: Examples of day→night translation on SfM dataset [43] with generators based on edge–consistency. Source image (left column) is translated with (from left to right): HED$^{\text{N}}$GAN, HEDGAN, and SobelGAN. All models are trained on the SfM dataset.

Figure A.3: Examples of day→night translation on different datasets with different generators. The input image (left column) is translated with (from left to right) HED$^{\text{N}}$GAN, CycleGAN [57], CUT, DRIT [59], and ToDayGAN [87] generator. The rows consists of example images from different datasets: SfM [43], Aachen [93, 94], Tokyo [96], Oxford [95], and RobotCar [106], respectively.

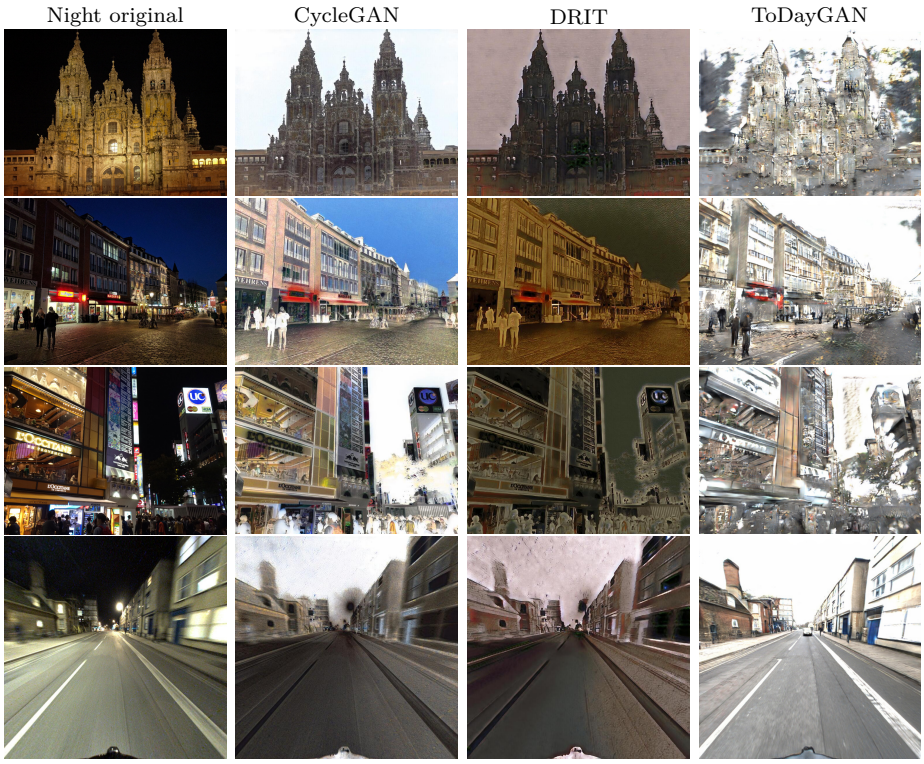


Figure A.4: Examples of night→day translation on different datasets with different generators. The input image (left column) is translated with (from left to right) CycleGAN [57], DRIT [59], and ToDayGAN [87] generator. The rows consists of example images from different datasets SfM N/D [51], Aachen [93, 94], Tokyo [96], and RobotCar [106], respectively.

Appendix B
Image Retrieval Ablations

B.1 Divere Anchors and CLAHE

After the GeM baseline [43], the work of [51] proposed to use the CLAHE photometric normalization [72] to bring the visual appearance of day and night images together. Moreover, diverse anchors mining (covered in Section 4.2.2) is used, which could make comparison with baselines questionable, since it was not used in previous works. Therefore, ablations with these two data preprocessing techniques are conducted.

Results in Table B.1 show both diverse anchors mining ($\mathcal{D}$) or CLAHE ($\mathcal{C}$) consistently improve day–night image retrieval (Tokyo 24/7) on both baselines and data augmentation–based methods, while the performance on the standard day–time retrieval ($\mathcal{R}$Oxf and $\mathcal{R}$Par) remains mostly unchanged. Slight improvement with the day–time retrieval is achieved, when both diverse anchors mining and CLAHE are used. These improvements can be concluded with the observation that diverse anchors mining further complements training image enhancements, such as day→night image translation or CLAHE.

B.2 Edge–based Image Retrieval

To qualitatively compare the performance of the HED and HED$^{\mathrm{N}}$ edge detectors, they are compared with the prior EdgeMAC [52] descriptor. All three embeddings are trained on edge maps obtained by the tested method from natural images. To be comparable with previous retrieval experiments, the image size is increased to 362 and edge maps are not binarized.

Table B.2 shows embedding based on HED$^{\mathrm{N}}$ outperforms embedding based on HED in edge–based retrieval. Using ensembles of edge map and image embeddings, HED$^{\mathrm{N}}$–based embedding also outperforms HED embedding in all evaluation cases. However, it should be noted, that the higher retrieval performance with the ensembles has the cost of double descriptor dimensionality.

Method	Avg	Tokyo	$\mathcal{R}$Oxf	$\mathcal{R}$Par
GeM	70.0	80.4	59.9	69.8
GeM $\mathcal{D}$	70.3	81.2	59.8	69.9
CLAHE $\mathcal{C}$	71.9	85.4	60.0	70.1
CLAHE $\mathcal{CD}$	**72.2**	**85.9**	**60.3**	**70.5**
GeM N/D !	71.5	84.0	60.4	70.0
GeM N/D $\mathcal{D}$!	71.5	84.1	60.3	70.1
CLAHE N/D $\mathcal{C}$!	72.5	87.5	59.9	70.1
CLAHE N/D $\mathcal{CD}$!	**73.0**	**87.7**	**60.8**	**70.7**
HED$^{\text{N}}$GAN	72.7	88.0	60.2	70.0
HED$^{\text{N}}$GAN $\mathcal{D}$	73.0	88.7	60.2	70.1
HED$^{\text{N}}$GAN $\mathcal{C}$	73.2	88.7	60.5	70.4
HED$^{\text{N}}$GAN $\mathcal{CD}$	**73.4**	**88.8**	**60.7**	**70.6**
CycleGAN	73.0	88.8	59.6	70.5
CycleGAN $\mathcal{D}$	73.3	89.1	59.9	70.7
CycleGAN $\mathcal{C}$	73.6	89.6	60.5	70.9
CycleGAN $\mathcal{CD}$	**74.0**	**90.2**	**60.7**	**71.0**

Table B.1: The effect of diverse anchors mining ($\mathcal{D}$) or CLAHE ($\mathcal{C}$) as a data preprocessing steps. In the top block, GeM baseline is compared with adding diverse anchors or CLAHE. In the second, third, and last block, the same experiments are performed with SfM N/D [51], HED$^{\text{N}}$GAN, and CycleGAN, respectively. The best score for each dataset is in bold.

Method	Avg	Tokyo	$\mathcal{R}$Oxf	$\mathcal{R}$Par
EdgeMAC [52]	45.6	75.9	17.3	43.5
HEDMAC	56.8	79.5	38.3	52.5
HED$^{\text{N}}$MAC	**59.2**	**81.9**	**38.4**	**57.2**
HEDMAC+GeM ‡	72.0	84.8	60.9	70.3
HED$^{\text{N}}$MAC+GeM ‡	72.6	85.7	**61.1**	70.9
HEDMAC+HED$^{\text{N}}$GAN $\mathcal{CD}$‡	73.8	90.9	60.1	70.5
HED$^{\text{N}}$MAC+HED$^{\text{N}}$GAN $\mathcal{CD}$‡	74.4	91.4	60.6	71.3
HEDMAC+CycleGAN $\mathcal{CD}$‡	74.2	91.5	60.0	71.2
HED$^{\text{N}}$MAC+CycleGAN $\mathcal{CD}$‡	**74.7**	**91.8**	60.4	**71.9**

Table B.2: The effect of trained HED$^{\text{N}}$ detector (from HED$^{\text{N}}$GAN) on the EdgeMAC descriptor [52]. In the top block, variants of EdgeMAC method, specifically HEDMAC and HED$^{\text{N}}$MAC are compared with EdgeMAC. HEDMAC or HED$^{\text{N}}$MAC use HED [73] edge detector with either original weights or weights taken after HED$^{\text{N}}$GAN training, respectively. In the bottom block, ensembles of EdgeMAC variants with chosen methods of image descriptor from Table 5.2 are reported. GeM is from [43]. Ensembles have double the dimensionality (1024) and are marked with ‡.