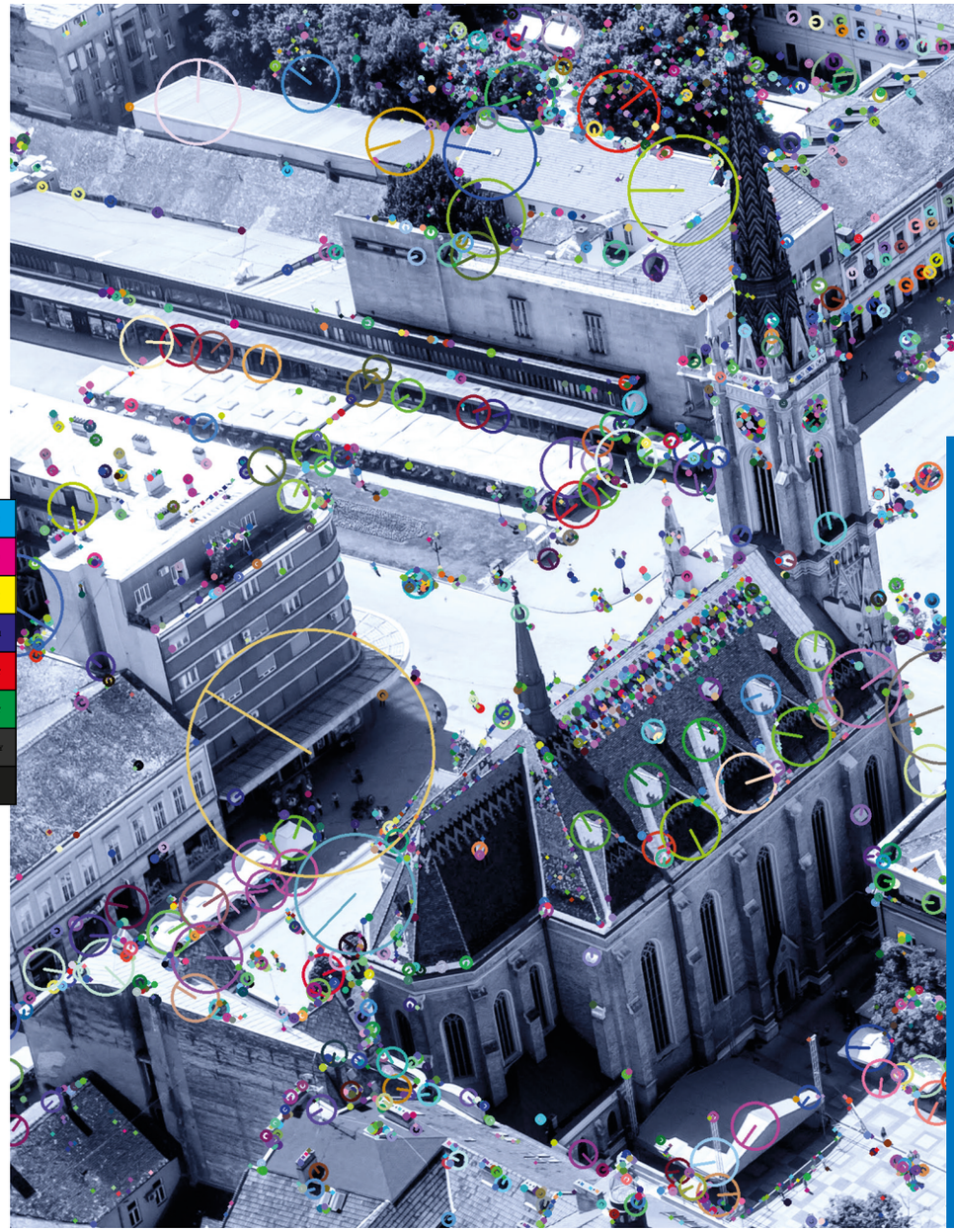
FACULTY OF ENGINEERING AND ARCHITECTURE



A photo of Novi Sad (Serbia), showing keypoint features as identified by the SIFT local image descriptor.

Nina Žižakić - doctoral dissertation

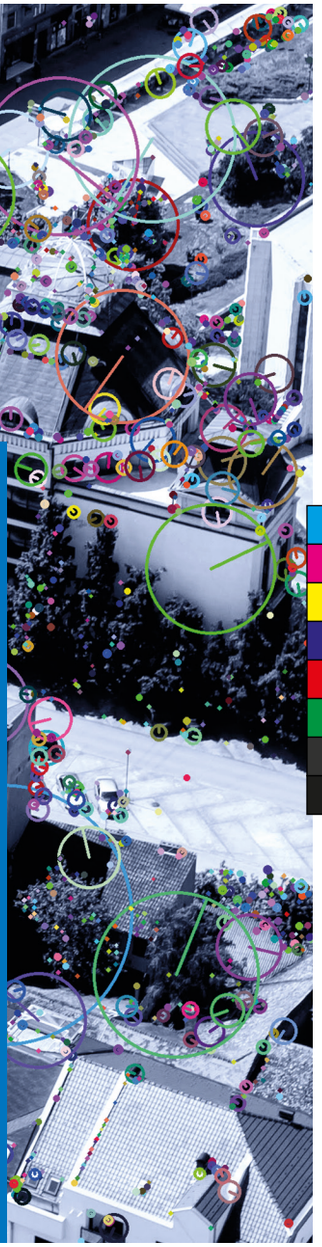
Autoencoder-Based Image Dimensionality Reduction Methods

Nina Žižakić

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisor
Prof. Aleksandra Pizurica, PhD
Department of Telecommunications and Information Processing
Faculty of Engineering and Architecture, Ghent University

November 2022



2022

GHENT UNIVERSITY

FACULTY OF ENGINEERING
AND ARCHITECTURE

Autoencoder-Based Image Dimensionality Reduction Methods

Nina Žižakić

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisor
Prof. Aleksandra Pizurica, PhD
Department of Telecommunications and Information Processing
Faculty of Engineering and Architecture, Ghent University

November 2022

GHENT
UNIVERSITY

ISBN 978-94-6355-642-2
NUR 984, 958
Wettelijk depot: D/2022/10.500/83

Members of the Examination Board

Chair

Prof. Patrick De Baets, PhD, Ghent University

Other members entitled to vote

Prof. Alin Achim, PhD, University of Bristol, United Kingdom
Prof. Hiep Luong, PhD, Ghent University
Prof. Miloš Radovanović, PhD, University of Novi Sad, Serbia
Prof. Yvan Saeys, PhD, Ghent University
Prof. Steven Verstockt, PhD, Ghent University

Supervisor

Prof. Aleksandra Pizurica, PhD, Ghent University

Acknowledgements

– You have my sword.
– And my bow.
– And my axe.
—Aragorn, Legolas, and Gimli (respectively), from *The Lord of the Rings: The Fellowship of the Ring*

Like Frodo in *The Lord of the Rings*, I, too, had a fellowship of people who have helped me in the journey to accomplishing my task (in my case, the task of finishing a PhD). I would like to explicitly thank them here.

First and foremost, I would like to express my deepest gratitude to my PhD advisor, prof. dr. ir. Aleksandra Pižurica. Thank you for giving me the opportunity to do my PhD here in Belgium, for believing in me when I didn't, for guiding me throughout my PhD with useful comments and discussions, and for your kindness.

I am also grateful to the members of my PhD examination board for taking the time to read my thesis, ask great questions, raise valid points and fair criticisms – all of which have led to an improved version of my thesis.

Next, I would like to thank my colleagues and friends from Telin who have made my PhD experience a great one! Thank you to Ivana, Martin, Ana, Zuhaib, Maarten (whose fault it is if there are any errors in the Dutch summary), David, Dimitri, Apoorv, Sarah, Anoek, Zaira, Brian, Sanne, Gianni, Michiel, Mathieu, Rafał, Jan, Hiep, Bart and others for fun (and often very random) lunch discussions – you have all made the mediocre *De Brug* food much more enjoyable! I also want to thank my GAIM colleagues whom I have really enjoyed working with – thank you (and xièxiè!) to Laurens, Shaoguang, Nicolas, Srđan, Marin, Izumi, Meizhu, Xianlu, Lingchen, Ting, Yoann and Roman. Thank you also to Philippe and Davy for their IT (and moral!) support and to Patrick and Sylvia for their help with any administrative or bureaucratic issues that I had.

I would also like to thank my other friends in Belgium that have made me feel like home here – the (ex) "Ghent friends" and the Iridians – both of whom have accepted me in their group and made me feel like one of them.

Next, I would like to thank my friends from high school ('specijalci') and my friends from uni, and especially the friends at the intersection of these two sets – the (active) "Gelender" members Dića, Trujić and Nikola. Hanging out with you guys is pure fun and seeing messages in our group chat always makes me happy, even when they are about Dota or NBA. I would also like to thank other 'specijalci' – Živko, Valentin, Uroš, Mire, Sergej and everyone else – thank you for accepting me in my true nerdy self. Thank you also to my friends from uni

in Novi Sad: Slobica, Raške, Jović, Nataša and Nemanja; and my friends from Paderborn: Milica, Demir and Matteo.

I would also like to thank my oldest friends (in terms of how long we've known each other – I am making no comments about their age): my best friend Milena (who is the light of every party, the most fun, kind, thoughtful and joyous person, and the best best friend I could have ever hoped to have (we have the same opinion!)), my 'žena' Antonina (who will have beaten me by two days in getting a PhD!), my neighbour Miloš and my 'mrvica' Milica! For the last 20 years I have a feeling I can share anything with you guys and get nothing but your full support!

I would like to thank Michael (who gets his own paragraph) for being a fun Covid-times office-mate, a kind person, and an immense support throughout my PhD. Thank you for putting up with my PhD-related and non-PhD–related stress and with my spam! Finally, I will not be less qualified than you!

Lastly, but most importantly, I want to thank my parents Mitar and Dušanka. By definition, I would not have achieved any of this if it wasn't for you. But, you two have done so much for me. Thank you for investing in me, for always being there for me, for supporting me and loving me unconditionally! I realise that I am extremely lucky to have you as my parents. I love you and I dedicate this thesis to you.

Ghent, October 2022
Nina Žižakić

Funding acknowledgement. This research received funding from the Flemish Government (AI Research Program).

Contents

Acknowledgements **i**

List of Abbreviations **xiii**

Samenvatting **xv**

Summary **xix**

1 Introduction **1**
1.1 AI, deep learning and autoencoders 1
1.2 Image dimensionality reduction 3
1.2.1 Local image descriptors 5
1.2.2 Image hashing 8
1.3 Research problems 11
1.4 Main contributions 12
1.5 Publications 15
1.5.1 Publications in international journals 15
1.5.2 Publications in international conferences 16
1.6 Outline of the thesis 16

2 Autoencoders **19**
2.1 Deep learning for image processing – preliminaries 19
2.1.1 Layers 20
2.1.2 Activation functions 23
2.1.3 Loss functions 26
2.2 A brief history of autoencoders 27
2.3 Applications of autoencoders 29
2.4 Types of autoencoders 30
2.4.1 Classical autoencoders 30
2.4.2 Sparse autoencoders 31
2.4.3 Denoising autoencoders 32
2.4.4 Variational autoencoders 32
2.5 Conclusion and summary 36

3 Learning local image descriptors with autoencoders **41**
3.1 Introduction and overview of local image descriptors 41
3.2 Benchmarks for local image descriptor evaluation 45
3.3 AE versus VAE and a hyperparameter study 46
3.3.1 Overview of the hyperparameters 46

3.3.2 The experimental setup . . . 50
3.3.3 Empirical results for hyperparameter selection . . . 51
3.4 Approximate evaluation of autoencoders for local image descriptors 54
3.4.1 Image similarity metrics . . . 55
3.4.2 Proposed approximate evaluation method . . . 57
3.4.2.1 The experimental setup . . . 57
3.4.2.2 Results and discussion . . . 61
3.5 Invertible local image descriptors . . . 63
3.5.1 Experimental results . . . 65
3.5.1.1 Evaluation on patch retrieval . . . 65
3.5.1.2 Evaluation of invertibility . . . 67
3.6 Conclusion . . . 69

4 Memory-efficient autoencoder-based local image descriptors 71
4.1 Introduction . . . 71
4.2 Reducing computational memory with intermediate representation 73
4.3 Experimental results . . . 76
4.3.1 Evaluation on HPatches benchmark . . . 76
4.3.2 Robustness to noise . . . 77
4.3.3 Robustness to missing data . . . 81
4.4 A case study – inpainting . . . 81
4.5 Conclusion . . . 87

5 Transferring the knowledge from hand-crafted to learning-based descriptors 89
5.1 Introduction . . . 89
5.2 Framework for knowledge transfer . . . 91
5.3 A learned variant of BRIEF . . . 92
5.3.1 BRIEF descriptor . . . 92
5.3.2 Learned BRIEF . . . 94
5.4 Experimental results . . . 95
5.5 Conclusions . . . 97

6 Deep image hashing with autoencoders 103
6.1 Introduction . . . 103
6.2 Hashing for content-based image retrieval – an overview . . . 105
6.3 Variational Autoencoder Twin-Bottleneck Hashing . . . 107
6.3.1 Improving the binary bottleneck . . . 107
6.3.2 Expanding the continuous bottleneck . . . 109
6.4 Experimental results . . . 109
6.5 Search of self-similar structures in electron microscopy images . 115
6.6 Conclusion . . . 117

7 Conclusions and future work 119
7.1 Conclusions . . . 119
7.2 Future work . . . 122

Bibliography 122

List of Figures

1.1 Ray Solomonoff's notes from the 1956 Dartmouth workshop on *Thinking Machines* . . . 1
1.2 Applications of image processing in everyday life . . . 3
1.3 Image sampled uniformly at random . . . 4
1.4 Manifold hypothesis . . . 5
1.5 Applications of local image descriptors . . . 6
1.6 Mars Helicopter Ingenuity using local image descriptors . . . 7
1.7 Content-based image retrieval . . . 8
1.8 An example of content-based image retrieval . . . 9

2.1 A single neuron in a neural network . . . 21
2.2 Fully-connected layer . . . 21
2.3 Convolutional layer . . . 23
2.4 Max-pooling layer . . . 23
2.5 Sigmoid activation function . . . 24
2.6 Tanh activation function . . . 24
2.7 Rectified Linear Unit activation function . . . 25
2.8 Parametric Linear Unit activation function . . . 26
2.9 Exponential Linear Unit activation function . . . 26
2.10 Autoencoders then and now . . . 28
2.11 Dimensionality reduction – local image descriptors . . . 29
2.12 Image generation with β-VAEs . . . 30
2.13 Classical autoencoder architecture . . . 31
2.14 Sparse autoencoder architecture . . . 32
2.15 Denoising autoencoder architecture . . . 33
2.16 2D latent space of MNIST dataset learned by an AE and a VAE 34
2.17 Variational autoencoders as probabilistic graphical models . . . 34
2.18 Variational autoencoder as a probabilistic graphical model . . . 37
2.19 Variational autoencoder as a neural network . . . 37
2.20 Manifolds learned with VAEs . . . 38

3.1 The effect of different hyperparameter choices on descriptor's performance (on HPatches) . . . 52
3.2 Performance by HPatches tasks (matching, retrieval and verification), showing the effect of different hyperparameter choices . 53

3.3 Correlation between performance of a descriptor learned using an AE on HPatches benchmark, and the distance in terms of different IQA metrics between the input and output patches of that AE . . . 58
3.4 Correlation between performance of a descriptor learned using an AE on different HPatches tasks, and the distance in terms of different IQA metrics between the input and output patches of that AE . . . 59
3.5 Correlation between performance of a descriptor learned using an AE on different HPatches tasks, and the distance in terms of different IQA metrics between the input and output patches of that AE . . . 60
3.6 Reconstructing an image from its SIFT descriptors . . . 63
3.7 The selected variational autoencoder architecture that we used for learning the invertible local image descriptor . . . 64
3.8 Patch retrieval examples . . . 66
3.9 Comparison of patch retrieval performance and patch reconstruction performance for different β_{norm} values . . . 68
3.10 Examples of patch reconstruction based on the descriptor's encoding . . . 69

4.1 Exploiting the proposed IR of an image in algorithms that require many patch comparisons . . . 75
4.2 Encoder architecture in AEs: traditional and proposed . . . 75
4.3 Comparison of performance on HPatches between different descriptors . . . 78
4.4 Memory needed for storing patch encodings using different descriptors and IR . . . 78
4.5 Patch retrieval examples . . . 79
4.6 Patch retrieval examples . . . 80
4.7 Comparison of descriptors' robustness to noise and missing data 82
4.8 Noisy patch retrieval and patch retrieval where the query has missing parts . . . 83
4.9 Image inpainting results . . . 85
4.10 Image inpainting results . . . 86

5.1 General framework for knowledge transfer from hand-crafted to learning-based descriptors . . . 93
5.2 Visualisation of the proposed knowledge transfer and improvement framework from a BRIEF descriptor to an autoencoder-learned descriptor . . . 96
5.3 Performance comparison between on the patch retrieval task . 98
5.4 Performance comparison on the image matching task . . . 98
5.5 Patch retrieval examples . . . 99
5.6 Patch retrieval examples . . . 100

6.1 Schematic of the twin-bottleneck hashing method . . . 106

6.2 Schematic of the proposed variational autoencoder twin-bottleneck hashing 108
6.3 Example images from MS-COCO dataset 110
6.4 Example images from CIFAR-10 dataset 110
6.5 Comparison of precision-recall curves on the CIFAR-10 dataset for TBH, our two improvements separately and together 111
6.6 Comparison of precision-recall curves on the MS-COCO dataset for TBH and our method 112
6.7 mAP of our method and state-of-the-art self-supervised hashing methods on the CIFAR-10 dataset 113
6.8 Retrieval examples for both TBH and our method 114
6.9 Content-based patch retrieval 116

List of Tables

3.1 Summary of the analysed hyperparameters and our findings regarding their influence on the quality of local image descriptors learned with (V)AEs 49
3.2 Patch retrieval performance comparison 67
3.3 Patch reconstruction performance comparison 67

List of Abbreviations

AAE	Adversarial Autoencoder
AE	Autoencoder
AI	Artificial Intelligence
ANN	Artificial Neural Network
BRIEF	Binary Robust Independent Elementary Features
BCE	Binary Cross-Entropy
CBIR	Content-based image retrieval
CNN	Convolutional Neural Network
CVAE	Conditional Variational Autoencoder
DL	Deep Learning
ELBO	Evidence Lower Bound
ELU	Exponential Linear Unit
EM	Expectation-Maximisation (algorithm)
FAST	Features from Accelerated Segment Test (descr.)
FC	Fully-Connected (Layer)
GAN	Generative Adversarial Network
i.i.d.	Independent and identically distributed
IR	Intermediate Representation
KLD	Kullback-Leibler Divergence
LSH	Locality Sensitive Hashing
mAP	Mean Average Precision
ML	Machine Learning
MP	Max Pooling (Layer)
MSE	Mean Squared Error
MS-SSIM	Multiscale Structural Similarity
NLP	Natural Language Processing
ORB	Oriented FAST and Rotated BRIEF (descriptor)

PCA	Principal Component Analysis
PGM	Probabilistic Graphical Model
ReLU	Rectified Linear Unit
SIFT	Scale-Invariant Feature Transform (descriptor)
SSD	Sum of Square Differences
SSIM	Structural Similarity Index
SURF	Speeded-Up Robust Features (descriptor)
SVM	Support Vector Machine
TBH	Twin Bottleneck Hashing
t-SNE	t-distributed stochastic neighbour embedding
VAE	Variational Autoencoder
VI	Variational Inference

Samenvatting

Naar schatting worden elke seconde zo'n 45.000 beelden vastgelegd. Dat is een enorme hoeveelheid beelden die zich opstapelt tot zo'n 1,5 triljoen beelden per jaar. Ondanks de ongelooflijke vooruitgang van de computertechnologie in de laatste 50 jaar, is het nog steeds moeilijk om de grote hoeveelheden beelden die elke seconde worden vastgelegd en hun hoge dimensionaliteit bij te houden. In het bijzonder is er een toenemende vraag naar zowel de extractie van toepassingsspecifieke beeldkenmerken als de mogelijkheid om beelden te representeren in een lagere dimensionale ruimte zodat ze kunnen worden gebruikt door computervisie-algoritmen. Met andere woorden, methoden voor beelddimensionaliteitsreductie zijn relevanter dan ooit.

Twee belangrijke families van methoden voor beelddimensionaliteitsreductie zijn lokale beelddescriptoren en methoden voor het hashen van beelden. Lokale beelddescriptoren laten ons toe verschillende beeldpatches te matchen op basis van hun gelijkenis en spelen dus een cruciale rol in vele beeldverwerkingstaken, zoals het tracken, herkennen en doorzoeken van objecten. Het hashen van beelden is een fundamenteel instrument voor inhoudelijke samenvatting van beelden, waarbij de meest gelijkende beelden worden opgehaald voor een gegeven zoekbeeld.

Lange tijd waren deze methoden voor het verminderen van de beelddimensies gebaseerd op met de hand vervaardigde kenmerken die werden gebruikt om een beeld of beeldfragment om te zetten in een laagdimensionale codering, die vervolgens kon worden gebruikt in computervisie-algoritmen. Bekende voorbeelden van manuele vervaardigde methoden voor beelddimensionaliteitsreductie zijn localiteitsgevoelige hashing en de lokale beelddescriptoren SIFT en HOG. In het afgelopen decennium zijn de trends in beeldverwerking verschoven van handcrafted methodes naar op leergebaseerde methoden, als gevolg van het enorme succes van deep learning. Deze trend heeft geleid tot een toename van de populariteit van op leergebaseerde methoden voor beelddimensionaliteitsreductie, die al indrukwekkende resultaten hebben opgeleverd. De belangstelling is echter vooral uitgegaan naar methoden met supervisie, waarvoor gelabelde gegevens nodig zijn. Deze vereiste is problematisch voor toepassingen waarbij de interessante patronen relatief zeldzaam kunnen zijn en/of het labelproces kennis van deskundigen vereist (bijvoorbeeld bij elektronenmicroscopiebeeldvorming van biologische weefsels). Een mogelijke oplossing voor dit probleem is het overwegen van methoden die geen gelabelde gegevens vereisen, die ook veelbelovende eerste resultaten hebben laten zien in dit onderzoeksgebied, maar meer onderzoek is nodig om te bepalen of deze methoden een levensvatbare aanpak zijn voor dergelijke toepassingen.

Het doel van dit proefschrift is het ontwikkelen van zelfgesuperviseerde leer-

methoden gebaseerd op autoencoders (AE) en variationele autoencoders (VAE) voor beelddimensionaliteitsreductie van lokale beelddescriptoren en het hashen van beelden. Er zijn verschillende onopgeloste uitdagingen die we hebben geïdentificeerd in dit onderzoeksgebied. Het is onduidelijk hoe autoencoders zich verhouden tot variationele autoencoders in deze context en wat de voor- en nadelen zijn van beide methoden met betrekking tot dit probleem. Verder is er geen literatuur over de invloed van hyperparameterselectie voor het trainen van autoencoders en variationele autoencoderd voor beelddimensionaliteitsreductie – dit wordt meestal ad hoc uitgevoerd zonder een grondig onderzoek en rapportage over de invloed van hyperparameters op de prestaties van deze methoden. Bovendien is de evaluatie van deze zelfgesuperviseerde methoden ook een uitdagend probleem omdat we geen toegang hebben tot de gelabelde gegevens. Hoewel het mogelijk is om de evaluatie uit te voeren op de benchmarks, is deze evaluatie vaak computationeel intensief en niet haalbaar als een metriek die tijdens de training kan worden gebruikt.

In deze dissertatie pakken we deze onderzoeksuitdagingen aan en stellen we methoden voor die gebaseerd zijn op zelfgesuperviseerd leren, voor zowel het leren van lokale beelddescriptoren als voor het hashen van beelden. De belangrijkste bijdragen van dit proefschrift zijn als volgt:

- *Een lokale beelddescriptor die het mogelijk maakt patch-coderingen geheugenefficiënt op te slaan.* Een veelvoorkomende uitdaging bij beeldverwerkingstaken die veel patch-vergelijkingen binnen een beeld vereisen, is dat ze ofwel (i) alle patch descriptoren moeten opslaan (wat veel geheugen vereist) of (ii) ze telkens opnieuw moeten berekend worden wanneer de patches nodig zijn voor de taak (wat veel rekentijd vereist). Wij stellen een originele architectuur van een autoencoder voor die efficiënt omgaat met lokale beelddescriptoren. Deze methode vergemakkelijkt toepassingen met veel patch-vergelijkingen binnen een beeld, die typisch voorkomen in beeldverwerkingstaken zoals denoising en inpainting. Onze voorgestelde architectuur produceert een speciale beeldrepresentatie, die een compacte manier mogelijk maakt om de descriptoren van alle patches van een beeld op te slaan, omdat de descriptoren van overlappende patches zichzelf overlappen, terwijl ook de descriptorprestaties niet in gevaar worden gebracht. Bij wijze van proof-of-concept integreren we onze descriptor in een inpainting-algoritme en evalueren we kwalitatief de prestaties ervan bij toepassing op de virtuele restauratie van meesterschilderijen.

- *Belangrijke inzichten in autoencoders voor het leren van lokale beelddescriptoren.* Ondanks het feit dat autoencoders en variationele autoencoders populaire zelfgesuperviseerd leermethoden zijn, zijn er verschillende hiaten in de literatuur over hoe ze effectief getraind moeten worden en hoe ze correct geëvalueerd kunnen worden. Bovendien ontbreekt het in de literatuur aan grondige vergelijkingen tussen autoencoders en variationele autoencoders voor het leren van lokale beelddescriptoren. In deze dissertatie gaan we in op deze problemen en presenteren we onze bevindingen met betrekking tot de mogelijke benaderingen voor het trainen

en evalueren van autoencoders. We voeren een grondige vergelijkende analyse uit van deze twee neurale netwerkarchitecturen samen met een diepgaande analyse van de meest relevante hyperparameters om hun optimale selectie te begeleiden. Naast deze analyse geven we inzicht in de uitdagingen en het belang van het selecteren van de juiste evaluatietechnieken tijdens het zelfgesuperviseerd leren van de lokale beelddescriptoren en stellen we een snelle approximatieve evaluatiemethode voor van descriptoren geleerd met autoencoders die sterk gecorreleerd is met veelgebruikte benchmarks.

- *Een raamwerk voor het overbrengen van de kennis van handcrafted naar leergebaseerde lokale beelddescriptoren.* Een interessant probleem dat nog niet eerder is onderzocht (voor zover wij weten) is het probleem van het gebruik van met de hand gekozen kenmerken als startpunt van waaruit een neuraal netwerk verder kan worden getraind, om de training ervan een voorsprong te geven met als potentieel voordeel dat de modellen beter verklaarbaar worden. Daartoe hebben we een raamwerk ontwikkeld om kennisoverdracht van handcrafted naar leergebaseerde descriptoren mogelijk te maken. Dit raamwerk beoogt de prestaties van leergebaseerde descriptoren te bereiken met behoud van veel van de voordelen van handcrafted descriptoren. De descriptor die met behulp van dit raamwerk wordt aangeleerd is beter uit te leggen (geërfd van de handcrafted descriptor) en kan nauwkeurig worden afgestemd op specifieke beeldverwerkingstoepassingen zonder dat daarvoor een gelabelde dataset nodig is. Deze karakteristieken zijn niet waar voor de handgemaakte noch voor de supergeviseerde leergebaseerde descriptoren. We demonstreren het gebruik van dit raamwerk door het creëren van de *geleerde BRIEF* descriptor (gebaseerd op de BRIEF handcrafted descriptor). Om dit te bereiken stellen we een elegante implementatie van BRIEF voor als een convolutioneel neuraal netwerk.

- Een lokale beelddescriptor die het mogelijk maakt om patch-coderingen terug te draaien naar de originele patches. Het inverteren van lokale beelddescriptoren is een actief onderzoeksgebied geweest in de afgelopen tien jaar, maar het is nog nooit uitgevoerd met behulp van autoencoders. In deze dissertatie dragen we een efficiënte methode aan voor het leren van lokale beelddescriptoren en hun inversiefunctie met behulp van een β-variationele autoencoder. We onderzoeken verschillende waarden van β in de verliesfunctie van de β-VAE om de optimale balans te vinden tussen het stimuleren van de overeenkomsten tussen input patches om behouden te blijven in de linbeddingsruimte, en het verzekeren van een goede reconstructie van de patches vanuit hun coderingen in de inbeddingsruimte. Onze voorgestelde descriptor toont patch retrieval vergelijkbaar met de referentie autoencoder-gebaseerde lokale beelddescriptor en toont verbeterde reconstructie van patches van hun coderingen.

- *Een zelfgesuperviseerd deep image hashing methode die gebruik maakt van variationele autoencoders om de compressie- en verklaarbaarheidaspecten*

van image hashing te verbeteren. Onze voorgestelde deep hashing methode is gebaseerd op twin-bottleneck hashing die beide bottlenecks verbetert door gebruik te maken van recente inzichten op het gebied van variationele autoencoders. In de binaire bottleneck veranderen we de generatie van hash codes om gebaseerd te zijn op variationele autoencoder, waardoor het leren van ontwarde variabelen wordt bevorderd en we de regularisator kunnen weglaten, waardoor het model wordt vereenvoudigd. In het continue knelpunt gebruiken we een variationele autoencoder die getraind wordt met behulp van een beperkte optimalisatieopzet, om de trade-off tussen compressie en reconstructiekwaliteit van de gegenereerde monsters beter te beheersen. Beide verbeteringen resulteren afzonderlijk in betere hashingprestaties, en een nog beter resultaat wanneer ze samen worden toegepast. De resultaten zijn grondig gevalideerd op alle hashcode groottes (16-bit, 32-bit en 64-bit) en benchmark datasets CIFAR-10 (60k beelden) en MS-COCO (330k beelden), en tonen aan dat onze methode beter presteert dan de state-of-the-art in deep hashing zonder gelabelde gegevens.

- *Case study over het efficiënt zoeken van gelijkvormige structuren in elektronenmicroscopiebeelden.* We demonstreren het potentieel van de toepassing van onze ontwikkelde beelddimensionaliteitsreductiemethoden door te zoeken naar regio's van beelden die biologische structuren bevatten die vergelijkbaar zijn met de structuur van een zoekpatch. We gebruiken lokale beelddescriptoren om de patches te coderen en het zoekproces te versnellen. De eerste stappen die voor deze toepassing zijn gezet, laten veelbelovende resultaten zien.

Het in dit proefschrift gepresenteerde werk heeft geleid tot twee tijdschriftpublicaties als eerste auteur (één opgenomen in het Web of Science en één in een peer-reviewed internationaal tijdschrift), één tijdschriftpublicatie in voorbereiding, en zeven publicaties in de proceedings van internationale conferenties, waarvan vijf als eerste auteur (één momenteel in review).

Summary

It is estimated that around 45 thousand images are captured every second, a vast amount of images that accumulates to around 1.5 trillion images every year. Despite incredible advances in computer technology over the last 50 years, it is still difficult to keep up with the large quantities of images captured every second and their high dimensionality. In particular, there is increasing demand for both the extraction of application-specific image features and the ability to represent images in lower-dimensional space so that they can be utilised by computer vision algorithms. In other words, image dimensionality reduction methods have become more relevant than ever.

Two important families of image dimensionality reduction methods are local image descriptors and methods for image hashing. Local image descriptors allow us to match different image patches based on their similarity and thus play a crucial role in many image processing tasks, such as object tracking and recognition, panorama image stitching, and image retrieval. Image hashing is an fundamental tool for content-based image retrieval, where the most similar images are retrieved for a given query image.

For a long time, these image dimensionality reduction methods relied on hand-crafted features which were used to transform an image or image patch into a lower-dimensional encoding, which could then be used in computer vision algorithms. Well-known examples of hand-crafted image dimensionality reduction methods include locality-sensitive hashing and local image descriptors SIFT and HOG. In the past decade, due to the huge success of deep learning, the trends in image processing have shifted from hand-crafted to learning-based methods. This trend has led to an increase in the popularity of learning-based image dimensionality reduction methods which has already yielded impressive results. However, the interest has mostly been concentrated on supervised methods which require labelled data. This requirement is problematic for the applications where the patterns of interest may be relatively rare and/or the labelling process requires expert knowledge (for example, electron microscopy imaging of biological tissues). One potential solution to this problem is to consider methods that do not require labelled data, which have also shown promising initial results in this area of research, however, more research is needed to determine whether or not these methods are a viable approach for such applications.

The objective of this thesis is to develop self-supervised learning methods based on autoencoders (AEs) and variational autoencoders (VAEs) for image dimensionality reduction – local image descriptors and image hashing. There are several unaddressed challenges that we have identified in this area of research. It is unclear how autoencoders compare with variational autoencoders

in this context and what are advantages and disadvantages of both methods regarding this problem. Furthermore, there is no literature on the influence of hyperparameter selection for training AEs and VAEs for image dimensionality reduction – this is usually performed in an ad-hoc fashion without a thorough investigation and reporting on the influence of hyperparameters on the performance of the methods. Moreover, evaluating these self-supervised methods is also a challenging problem since we do not have access to the labelled data. While it is possible to perform the evaluation on the benchmarks, this evaluation is often computationally intensive and not feasible as a metric to be used during training.

In this thesis, we address these research challenges and propose self-supervised-learning–based methods for both learning local image descriptors and for image hashing. The main contributions of this thesis are as follows:

- *A local image descriptor that allows for memory-efficient storing of patch encodings.* A common challenge with image processing tasks that require many patch comparisons within an image is that they have to either (i) store all the patches' descriptors (which requires a lot of memory) or (ii) recalculate them every time the patches are needed for the task (which requires a lot of computational time). We propose an original autoencoder architecture that efficiently handles local image descriptors. This method facilitates applications with many patch comparisons within an image, which typically arise in image processing tasks such as denoising and inpainting. Our proposed network architecture produces a special image representation, which enables a compact way of storing the descriptors of all the patches of an image because the descriptors of overlapping patches overlap themselves, while also not compromising on descriptor's performance. As a proof of concept, we integrate our descriptor into an inpainting algorithm and qualitatively evaluate its performance when applied to the virtual restoration of master paintings.

- *Important insights into autoencoders for learning local image descriptors.* Despite the fact that autoencoders and variational autoencoders are popular self-supervised learning methods, there are several gaps in the literature on matters concerning how to effectively train and properly evaluate them. Moreover, the literature lacks thorough comparisons between autoencoders and variational autoencoders for learning local image descriptors. In this thesis, we address these knowledge gaps and present our findings regarding the possible approaches to the training and evaluation of autoencoders. We perform a thorough comparative analysis of these two neural network architectures along with an in-depth analysis of the most relevant hyperparameters to guide their optimal selection. In addition to this analysis, we provide insights into the challenges and the importance of selecting right evaluation techniques during the self-supervised learning of the local image descriptors and propose a rapid approximate evaluation method for descriptors learned with autoencoders which is strongly correlated with established benchmarks.

- *A framework for transferring the knowledge from hand-crafted to learning-*

based local image descriptors. An interesting problem that has not been explored before (to the best of our knowledge) is the problem of using the hand-crafted features as a starting point from which a neural network can further be trained, in order to give its training a 'head start' and with the potential benefit of allowing for higher explainability of the models. To that end, we developed a framework for enabling knowledge transfer from hand-crafted to learning-based descriptors. This framework aims to achieve the performance of learning-based descriptors while preserving many of the benefits of hand-crafted descriptors. The descriptor learned using this framework has improved explainability (inherited from the hand-crafted descriptor) and can be fine-tuned for specific image processing applications without the need for a labelled dataset. These characteristics are not true of neither the hand-crafted nor the supervised-learning–based descriptors. We demonstrate the use of this framework by creating the *learned BRIEF* descriptor (based on the BRIEF hand-crafted descriptor). To achieve this, we propose an elegant implementation of BRIEF as a convolutional neural network.

- *A local image descriptor that allows for inverting patch encodings back onto the original patches.* Inverting local image descriptors has been an active area of research in the past decade, however, it has never been performed using autoencoders. In this thesis, we contribute an efficient method for learning local image descriptor and its inversion function using a β-variational autoencoder. We examine different values of β in the loss function of the β-VAE to find the optimal balance between incentivising the similarities between input patches to be preserved in latent space, and ensuring good reconstruction of the patches from their encodings in latent space. Our proposed descriptor demonstrates patch retrieval comparable to the reference autoencoder-based local image descriptor and shows improved reconstruction of patches from their encodings.

- *A self-supervised deep image hashing method that utilises variational autoencoders to improve the compression and explainability aspects of image hashing.* Our proposed deep hashing method is based on twin-bottleneck hashing that improves both bottlenecks using recent insights in the field of variational autoencoders. In the binary bottleneck we change the generation of hash codes to be based on variational autoencoder, thus promoting learning of disentangled variables and allowing us to omit the regulariser, therefore simplifying the model. In the continuous bottleneck we employ a variational autoencoder that is trained using a constrained optimisation setup, in order to better control the trade-off between compression and reconstruction quality of generated samples. Both of these improvements result in better hashing performance separately, and an even better result when applied together. The results have been thoroughly validated on all hash-code sizes (16-bit, 32-bit and 64-bit) and benchmark datasets CIFAR-10 (60k images) and MS-COCO (330k images), showing that our method outperforms the state of the art in deep hashing without labelled data.

- *Case study on efficient search of self-similar structures in electron microscopy images.* We demonstrate the potential of the application of our developed image dimensionality reduction methods by searching for regions of images that contain biological structures similar to the structure on the query image patch. We are using local image descriptors to encode the patches and accelerate the search process. The first steps that have been taken for this application show promising results.

The work presented in this thesis has led to two journal publications as the first author (one included in the Web of Science and another in a peer-reviewed international journal), one journal publication in preparation, and seven publications in the proceedings of international conferences, five of which as the first author (one currently in review).

1

Introduction

All we have to decide is what to do with the time that is given to us.
—Galadriel, from *The Lord of the Rings: The Fellowship of the Ring*

1.1 AI, deep learning and autoencoders

Humans have been intrigued by the idea of non-human intelligence since the ancient times. The history of artificial intelligence (AI) is considered to begin with the Greek myths of Pygmalion, Daedalus and Hephaestus about artificial beings endowed with intelligence by their master craftsmen [Sparkes 96, Tandy 97, Ovid 04]. In mid-19th century, more than hundred years before the first computer was built, Charles Babbage and Ada Lovelace worked on programmable mechanical calculating machines [McCorduck 04], with Lovelace now being recognised as the first computer programmer for her algorithm intended to be performed by such a machine [Lovelace 42]. In 1950, Alan Turing published a seminal paper in which he contemplated the possibility of *machines that think* [Turing 50]. He defined the term 'thinking' by proposing the "imitation game as a test", now famously known as the Turing Test.

The term AI was coined in 1956 at a workshop at Dartmouth College [Kaplan 19]. The attendees of the workshop included famous names such as John McCarthy (the organiser), Claude Shannon, John Nash, Marvin Minsky and Ray Solomonoff [Solomonoff 56].

In the early days of AI, the researchers addressed problems that can be described with formal (mathematical) rules, such as first-order logic. These

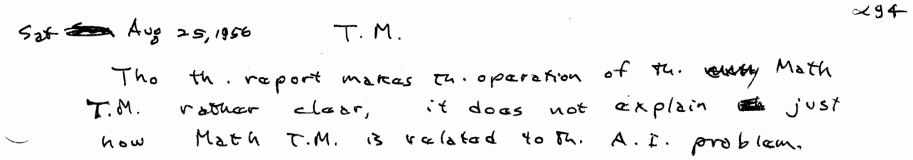


Figure 1.1: Ray Solomonoff's notes from the 1956 Dartmouth College workshop on *Thinking Machines* [Solomonoff 56], where the term AI was coined.

problems were typically difficult for humans but were simple to implement on the computers (e.g. finding a shortest path between two vertices in a weighted graph). The real challenge for AI, however, proved to be solving problems that are straightforward and intuitive for humans. Such problems include recognising objects in an image, speech recognition, and understanding text.

One initial approach to solving such problems is known as **knowledge-based systems**, which involved hard-coding the knowledge about the world using formal languages. The idea was that a computer can reason about the statements in the formal languages by using the logic inference rules. This approach, however, proved difficult as it was nearly impossible to create a knowledge base large enough to encompass, for example, all the rules in a natural language [Jacobs 85, Lenat 89].

The shortfalls of knowledge-based systems led researchers to believe that one needs to build a system that can *learn* the rules instead of having them hard-coded – thus **machine learning** was born. The idea of machine learning was that algorithms can improve automatically by extracting patterns from data. While the original ML algorithms (such as logistic regression) showed good performance on simple datasets (e.g. the famous Kaggle challenge of predicting survival on the Titanic [Kaggle 10]), it became clear that these algorithms are not capable of addressing more challenging problems such as computer vision and natural language processing.

After a few stagnate periods in the development of such systems, the field of AI started to gain traction again around the year 2010. This increase in interest was caused by several factors, namely, faster and cheaper computers, algorithmic improvements, and access to large amounts of data (*big data*). Researchers realised that making the neural networks deeper (where depth of a network refers to the number of layers) and training them on larger amounts of data would result in the improvement of their performance on image processing tasks such as object classification [Krizhevsky 09]. Incorporating machine learning techniques to use these deep neural networks has given rise to **deep learning**, which is still a thriving field of research today.

The extent to which the field of deep learning has been successful cannot be overstated. In the beginning, a lot of the success was in supervised deep learning, where models learn to associate some input (data) to some output (labels), with the applications such as image processing and computer vision (e.g. image or object classification) [Krizhevsky 09, Simonyan 14, He 16] and speech recognition [Deng 13, Abdel-Hamid 14, Sainath 15].

In some applications, however, supervised learning is not a viable approach due to the requirement of providing labelled datasets. Modern datasets are often made up of millions of samples that need to be labelled if they are to be used by supervised learning techniques. Labelling this many samples, however, is costly and in many cases simply not practical. Moreover, sometimes the labelling can only be performed by experts in the field (e.g. medical doctors) whose time is very valuable. Most companies and research groups do not have the resources for this type of labelling and in these cases, supervised learning is not a good fit.

In parallel to the development of supervised learning techniques, there has

also been development of self-supervised learning techniques, which do not rely on labelled data. These techniques exploit characteristics and patterns in the data to learn the required behaviour.

One example of a highly successful self-supervised learning technique is an autoencoder (AE) [Hinton 06], where a network is trained to reproduce its input via a lower-dimensional intermediate layer. More recently, research into autoencoders has led to variational autoencoders (VAEs), which are based on probabilistic graphical models [Kingma 13]. In the past decade, AEs and VAEs have been used for dimensionality reduction, learning features, and generative modelling. In this thesis, we will study how AEs and VAEs can be trained and used for image dimensionality reduction.

1.2 Image dimensionality reduction

Vast amounts of images are captured every second – many quick selfies we send our friends, numerous photographs capturing important moments of our lives, various medical scans the doctors perform to examine our health, abundant satellite images monitoring the current climate and geopolitical issues. Despite the fact that our computers – from the smartphones in our pockets to the massive clusters of supercomputers in specialised data centres – have seen incredible advances and can perform operations unfathomable to us only a decade ago, it is still difficult to keep up with the vast quantity of images captured every second and their high dimensionality. In particular, there is increasing demand for both the extraction of application-specific image features and the ability to represent images in lower-dimensional space so that they can be utilised by computer vision algorithms.

Figure 1.2: Applications of image processing in everyday life.

The information contained in images is highly redundant [Tošić 11]. In natural images, we typically have many nearly-flat or slowly-changing areas and many repetitive patterns, which is why we can efficiently compress the images, i.e. transform them onto a lower-dimensional space. This property of natural images is outlined in the *manifold hypothesis* [Cayton 05], which states that natural images (and other types of data such as sounds and text) lie on **lower-dimensional manifolds**. The supporting evidence for the manifold

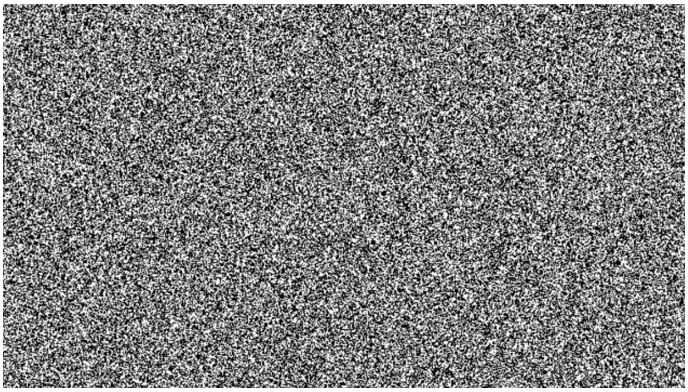


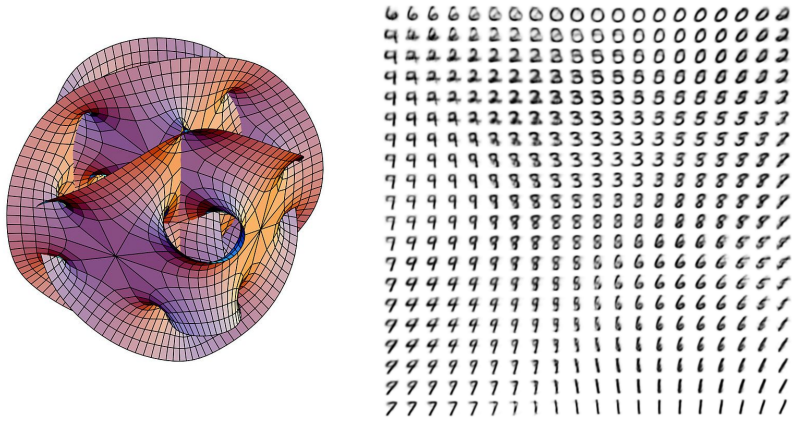
Figure 1.3: Image sampled uniformly at random (each pixel value is sampled from a uniform distribution). Despite the fact that there is a non-zero probability of generating an image that looks like a 'natural image' (e.g. of a house, a dog), we do not observe this in practice, suggesting that natural images occupy a negligible portion of the space of all possible images.

hypothesis can be summarised as follows. Firstly, by sampling uniformly from the space of all possible images, it is extremely unlikely that a natural image would ever be found. For example, Figure 1.3 shows a uniformly sampled image, which resembles meaningless noise. Secondly, the natural image data space seems to be continuous. We can imagine many possible transformations that would allow one to trace a manifold through this space of all possible images: one could gradually move the position and orientation of the camera, gradually move or rotate objects in the camera's frame, or gradually adjust the brightness of the lights in the room. This characteristic of the image data space is evident in Figure 1.4 (b), which shows how the subspace of handwritten digits is interconnected. Moreover, the manifold hypothesis has been supported through rigorous experiments [Brand 03, Belkin 03, Donoho 03, Cayton 05, Weinberger 06, Narayanan 10, Fefferman 16].

The realisation that natural images lie on low-dimensional manifolds has led to the research on manifold learning, and more generally, dimensionality reduction.

The terms 'manifold learning' and 'dimensionality reduction' are used loosely and sometimes interchangeably in the literature, but they are not exactly the same. Dimensionality reduction generally refers to any kind of transformation of the data from a high-dimensional space into a lower-dimensional space. While this dimensionality reduction method can be linear, as it is with, for example, Principal Component Analysis (PCA), newly-developed methods are usually non-linear and aim to learn a lower-dimensional manifold (manifold learning).

Image dimensionality reduction can be performed in many applications and for a variety of reasons such as: allowing for faster and more efficient image search (content-based image retrieval), extracting the most important features of an image in order to classify it, lowering the computational time of different



(a) (b)

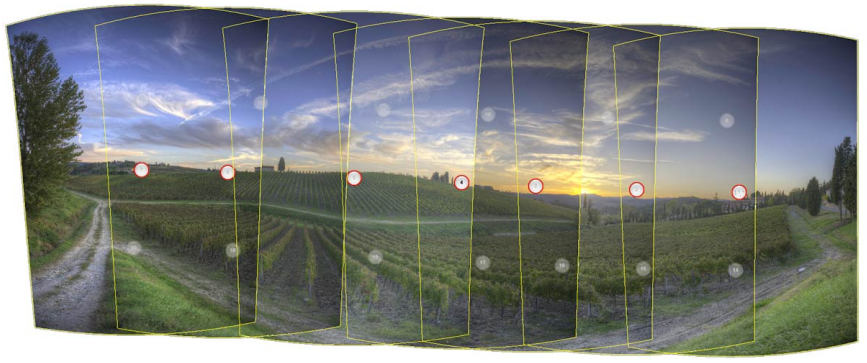
Figure 1.4: (a) A visualisation of a manifold – a connected region which locally appears to be a Euclidean space from any given point. (b) A two-dimensional manifold learned with a variational autoencoder on the MNIST dataset of handwritten digits [LeCun 10].

image processing tasks, facilitating explainability by providing insights into a machine learning pipeline, or tackling any other problems that arise from the curse of dimensionality. In this thesis, we focus on two types of image dimensionality reduction methods: local image descriptors and image hashing.

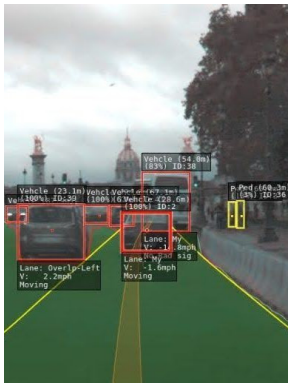
1.2.1 Local image descriptors

Local image descriptors are functions that transform a small part of an image (an image patch) into a compact representation – a patch encoding. A local image descriptor should preserve similarities between patches – two similar patches should also have similar encodings. Depending on the type of application where the local image descriptor is employed, it can be invariant to rotation, translation or brightness.

Local image descriptors are a crucial building block of various image processing tasks including: image inpainting, denoising, stitching, object tracking, motion estimation, and saliency detection. For example, in image stitching, matching parts of images are identified by comparing patches' descriptors [Brown 07]. In object tracking and motion estimation, local image descriptors are used to identify corresponding objects in subsequent frames and to associate those objects between frames [Yilmaz 06, Poddar 19]. Similarly to image stitching, this is done by comparing descriptors of patches between frames. In image inpainting, the goal is to fill in the missing part of an image in a visually plausible way. Patch-based inpainting methods replace (partially) missing image patches with the matching ones taken from the undamaged parts of the image. Local image descriptors can be used here to improve and speed up the search for matching



Panorama image stitching



Object tracking

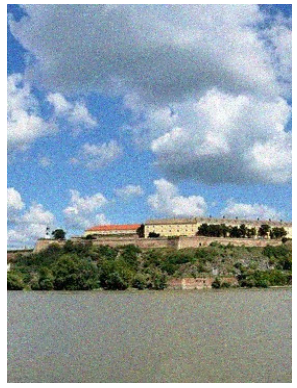


Image denoising

Image inpainting

Figure 1.5: Applications of local image descriptors

patches.

Local image descriptors are even used on Mars! Mars helicopter Ingenuity, the first powered controlled extraterrestrial aircraft, is able to fly thanks to local image descriptors. A modified local image descriptor FAST [Rosten 06] is detecting and encoding terrain features in each frame of the video feed that is being recorded by the navigation camera (Figure 1.6). Helicopter's velocity is then calculated based on the difference in positions of tracked features between subsequent frames [Bayard 19].

The traditional approach to designing local image descriptors involved using hand-crafted features, with SIFT [Lowe 99], HOG [Dalal 05], GLOH [Mikolajczyk 05], SURF [Bay 08] and DAISY [Tola 09] being the most famous descriptors. In recent years, deep learning has become a popular approach to solving many image processing challenges, including the design of local image descriptors. The first learned local image descriptors arose from the necessity to find a way of choosing parameters in the hand-crafted methods that

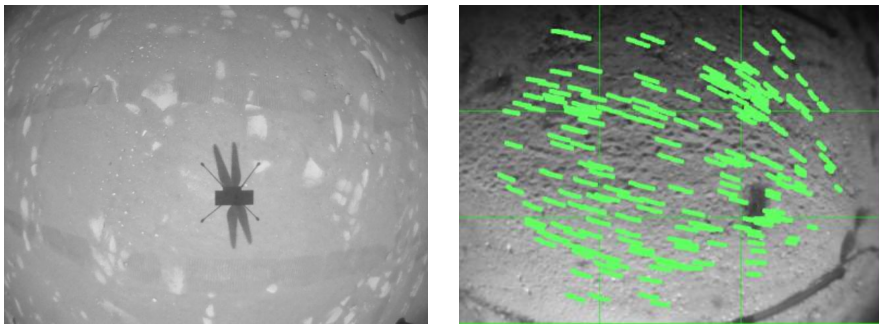


Figure 1.6: The Ingenuity Mars Helicopter's navigation camera uses local image descriptors for autonomous navigation. Left: Helicopter's shadow captured on the surface of Jezero Crater during Ingenuity's second experimental test flight on April 22, 2021. Credits: NASA/JPL-Caltech. Right: Example of feature tracking using Ingenuity's camera, showing features tracked over 10 consecutive frames. Image taken from [Bayard 19].

does not involve hand-tuning [Jin 05, Winder 07]. As deep learning methods developed further a growing number of learning-based descriptors started emerging. Nowadays, learned descriptors are mostly supervised methods that learn useful features on pairs of similar and dissimilar patches, striving for a high correlation between the similarity of the patches and the similarity of their descriptors. Most recent methods involve the use of convolutional neural networks [Fischer 14a, Han 15, Ono 18, Wang 20], some of which are siamese networks [Zagoruyko 15, Simo-Serra 15], or triplets [Balntas 16, Tian 19].

Some learned descriptors have been shown to outperform hand-crafted ones on benchmarks [Fischer 14a, Balntas 17]. Nevertheless, despite many advancements in the learning approach and their superior performance on benchmarks, hand-crafted descriptors still perform comparably or better than the learned descriptors in practical applications [Schonberger 17]. He et al. [He 18] argue that descriptor learning should not be approached as a standalone problem, but rather as a component of a broader image processing task, which must also be taken into consideration. For example, descriptors learned on general datasets such as ImageNet may not necessarily work well when part of an image processing task that involves some specific medical imaging dataset such as X-ray images. The ability to fine-tune them on such specific datasets could help improve their performance.

Self-supervised deep learning methods such as autoencoders do not suffer from dependence on labelled data. Chen et al. were the first to apply autoencoders to learn local image descriptors [Chen 15]. Their method shows promising results, however, the network architecture they used is very small (containing only one hidden layer) and thus lacks the capacity to learn and encode some useful features of patches.

In this thesis, we propose autoencoder-based and variational-autoencoder–

based architectures suitable for learning to capture relevant patch features for modern-day image processing applications. More generally, we research learning local image descriptors using autoencoders and variational autoencoders and provide insights on different aspects of their training process. We compare the performance of classical autoencoders with variational autoencoders and study how to select their hyperparameter in an optimal manner. We also propose an efficient method for approximate evaluation of autoencoders which enables their faster training. Furthermore, we investigate how to optimise for the invertibility of descriptors learned with autoencoders and present an invertible local image descriptor. We also propose a special autoencoder architecture that is optimised for image processing tasks that require many patch comparisons within a single image and, more specifically, for inpainting. Finally, we propose a framework for transferring knowledge from hand-crafted to self-supervised-learning–based descriptors in order to exploit the best characteristics of both approaches.

1.2.2 Image hashing

Image hashing is another image dimensionality reduction method wherein whole images are mapped onto low-dimensional hash codes (usually 16-bit, 32-bit or 64-bit). Image hashing is mainly used for content-based image retrieval (CBIR), where we want to retrieve the most similar images for a given query image. This similarity does not have to be visual but can also be semantic similarity, e.g. reflecting the types of object(s) depicted in images.

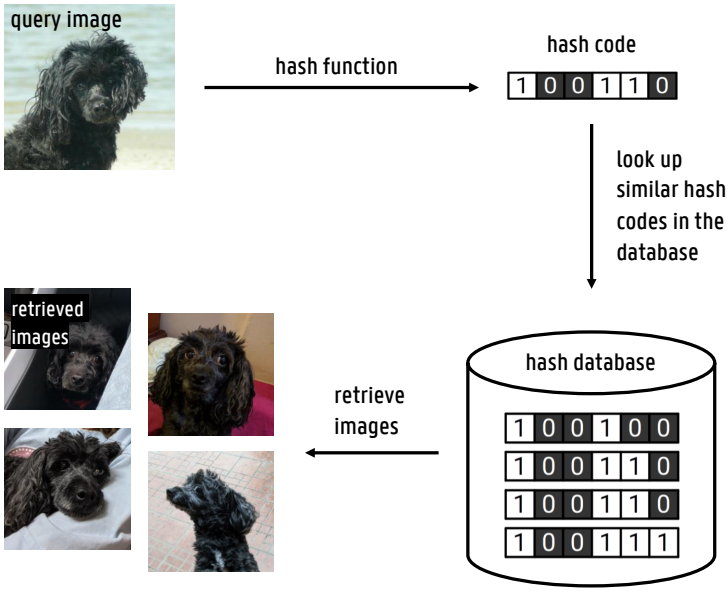


Figure 1.7: Content-based image retrieval

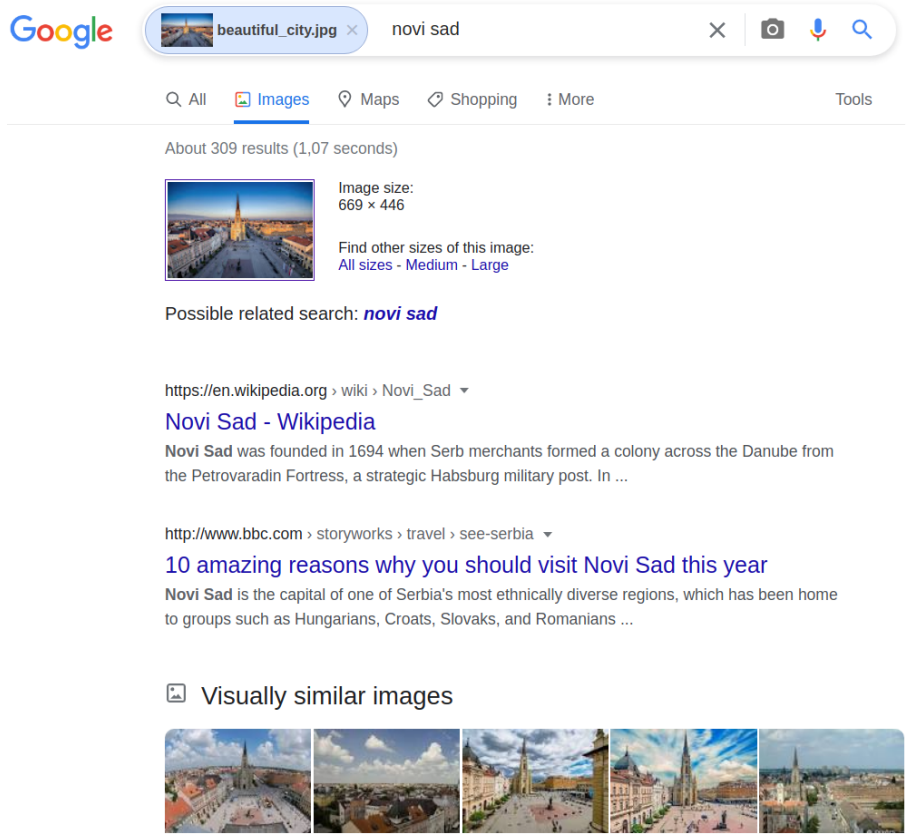


Figure 1.8: An example of content-based image retrieval – Google image search. An image is uploaded and visually similar images are retrieved (in addition to some information about the image).

While there is overlap between image hashing and local image descriptors, there are also important differences. For example, image hashing aims to represent specific information about the whole image (e.g., what is this a picture of) while local image descriptors work with image patches and aim to enable finding parts of the image that are similar.

Similarly to local image descriptors, we can classify image hashing methods into two categories: traditional (data-independent) methods and learning-based methods (data-dependent).

Traditional hashing methods rely on hand-crafted features, and can thus only capture visual similarity between images, rather than a semantic similarity. These methods are often referred to as locality sensitive hashing (LSH) methods, named after a family of hash functions that maps similar inputs to the same hash code. Following on the two pioneering works [Charikar 02, Andoni 06], many variations have been proposed using different distance metrics [Indyk 98,Dasgupta 11,Kulis 11,Ji 12] or changing the search method [Pan-

igrahy 05, Bawa 05, Pan 12]. The issue with these traditional hashing methods is the fact that they have no way of encoding the semantic similarities between images, only the visual similarities.

Learning-based methods, on the other hand, use deep learning to achieve not only visual but also semantic similarity. They have shown superior performance over the traditional hand-crafted methods. Most of the work in this field has focused so far on supervised techniques, which exploit the data distribution together with its annotations. Over recent years, many different architectures have been proposed for supervised learning to hash. The most straightforward techniques make use of a deep encoder to directly encode inputs to hash codes [Gong 13, Erin Liong 15, Shen 15, Wang 16b]. Yang et al. presented a fine-tuning process to convert such an encoder of a classifier into a deep hashing network [Yang 17]. Another popular supervised technique trains on predicting a pairwise loss function [Lin 13, Xia 14, Liu 16, Shi 16]. A natural extension to this are triplet-based losses, where a query data point is compared to both a similar and a dissimilar data point [Wang 16a]. Supervised generative adversarial networks have proven to be viable hashing methods [Qiu 17, Cao 18, Wang 18], as well as supervised autoencoder-based networks that exploit the models' bottleneck to extract hash codes [Cao 16, Dadaneh 20]. While supervised-learning–based methods show promising retrieval results, they require annotated datasets. This is a severe limitation, because, for many real-world datasets, it is not feasible or it is simply too costly to make annotations. There is a high demand for methods that can learn useful hash functions solely based on images, i.e. without relying on their labels or other types of additional information on what the images depict.

There is a wide variety of image hashing methods that do not rely on labelled data. Generative adversarial networks again show state-of-the-art performance in this field [Cao 18, Dizaji 18, Song 18, Zieba 18]. Similarly, autoencoders also remain a relevant technique [Carreira-Perpinán 15, En 17, Hansen 20]. Dai et al. build on the idea of a variational autoencoder, applying it to construct hash codes directly from the bottleneck [Dai 17]. Hu et al. propose a self-supervised technique which assigns pseudo labels to the data using precomputed features and shows promising results [Hu 17]. The approach optimises its hash function to maximally compress the dataset and is a generative approach since it can be used to regenerate the inputs. Recent work by Shen et al. [Shen 20] introduced a method called twin-bottleneck hashing (TBH) that uses autoencoder-type architecture which implements elements from graph-based learning.

In this thesis, we focus on self-supervised-learning–based image hashing. We propose a novel image hashing method that is optimised for both learning a useful hash function (that retrieves relevant images when being used for content-based image retrieval) and for its explainability (which is achieved by promoting disentangled representations). Our method achieves state-of-the-art performance in image hashing, as validated on different benchmark datasets.

1.3 Research problems

Tremendous amounts of images are captured every second, resulting in an ever-increasing need to extract the most important information from images and to represent them in a lower-dimensional space. This need necessitates the demand for effective image dimensionality reduction methods.

Traditionally, image dimensionality reduction involved using hand-crafted features to transform an image (or an image patch) to its lower-dimensional encoding. Local image descriptors such as SIFT [Lowe 99] and HOG [Dalal 05] are perhaps the most famous examples of hand-crafted methods. In the past decade, due to the huge success of deep learning, the trends in image processing have shifted from hand-crafted to learning-based methods. This trend shift has also led to the growing popularity of learning-based image dimensionality reduction methods, which showed impressive results [Shen 15, Balntas 16, He 18, Tian 19]. In general, these methods are supervised and rely on large labelled datasets in order to achieve adequate performance. However, when designing image dimensionality reduction methods for specific image processing tasks (e.g. image dimensionality reduction of electron microscopy images), one will not always have access to the resources (time and/or money) needed to acquire such extensive labelled datasets. In addition to the supervised methods, some self-supervised image dimensionality reduction methods have also shown promising results [En 17, Song 18, Shen 20], however, this field of research still has a lot of room for further development and innovation.

The objective of this thesis is to develop self-supervised learning methods (autoencoders and variational autoencoders) for image dimensionality reduction – local image descriptors and image hashing. There are several unaddressed challenges that we have identified in this area of research. It is unclear how autoencoders compare with variational autoencoders in this context and what are advantages and disadvantages of both methods regarding this problem. Furthermore, there is no literature on the influence of hyperparameter selection for training AEs and VAEs for image dimensionality reduction – this is usually performed in an ad-hoc fashion without a thorough investigation and reporting on the influence of hyperparameters on the performance of the methods. Moreover, evaluating these self-supervised methods is also a challenging problem since we do not have access to the labelled data. While it is possible to perform the evaluation on the benchmarks, this evaluation is often computationally intensive and not feasible as a metric to be used during training.

The research objectives for this thesis are as follows:

- Explore the differences between autoencoders and variational autoencoders for learning local image descriptors and analyse the hyperparameters' impact on descriptor's performance.

- Enable more efficient training of local image descriptors using self-supervised methods by accelerating the evaluation of the descriptors during training.

- Enable more efficient storage of the patch encodings calculated using a local image descriptor. Storing patch encodings for overlapping patches is not efficient with local image descriptors, i.e. we have to store the encodings for all the patches separately, even if the patches are overlapping.

- Explore a model-based approach where the knowledge from a specified hand-crafted descriptor is transferred to a learned descriptor. This type of knowledge transfer would allow for retaining explainability of the descriptor while allowing its performance to improve with the training.

- Develop a framework for inverting the encodings from self-supervised-learning–based local image descriptors back onto their original image patch. This problem has been an active area of research in the past decade, starting from inverting the hand-crafted descriptors [Weinzaepfel 11, Vondrick 13] to, more recently, inverting supervised-learning–based descriptors [Mahendran 15]. However, the research on inverting self-supervised-learning–based descriptors lacks adequate literature.

- Develop self-supervised image hashing method optimised without compromising on its explainability. Most well-performing image hashing methods are learning-based methods, with the neural networks often being seen as a black box. This can be an issue, especially in some critical use-cases such as medicine (where it is also difficult to find labelled datasets, thus self-supervised learning is a better fit). Applying more explainable self-supervised neural networks is needed for this use-case.

- Explore the potential of using image dimensionality reduction methods for the efficient search of self-similar structures in biological datasets. An example application of this is as follows. When a biologist or a doctor notices a strange pattern in an image of tissue, it would be of great use if they could find other occurrences of similar-looking tissue. It is possible to perform brute-force search for finding similar-looking patch candidates, but this limits the search space that can be searched in reasonable time. Performing such search in lower-dimensional space would allow us to expand the search space and/or decrease the computational time.

1.4 Main contributions

The focus of this thesis is to develop a more in-depth understanding on how autoencoders can be applied to perform two kinds of prominent image dimensionality reduction: local image descriptors and image hashing. In addition to investigating and developing local image descriptors and deep image hashing methods, we also provide insights into some important aspects of deep learning. These include exploring how to efficiently evaluate autoencoders, investigating what are the most important hyperparameters when training them for image dimensionality reduction, studying how to optimise for the invertibility of data

encodings, and researching how to transfer knowledge from traditional (handcrafted) methods to the learning-based methods, thus reaping the benefits of both approaches. The main contributions of this thesis can be summarised as follows:

- *A local image descriptor that allows for memory-efficient storing of patch encodings.*

  A common challenge with image processing tasks that require many patch comparisons within an image is that they have to either (i) store all the patches' descriptors (which requires a lot of memory) or (ii) recalculate them every time the patches are needed for the task (which requires a lot of computational time). We propose modifying the architecture of an autoencoder in order to yield a descriptor design specifically tailored for applications with many patch comparisons within a single image. Our proposed network architecture produces a special image representation which enables a compact way of storing the descriptors of all the patches of an image because the descriptors of overlapping patches overlap themselves. We call this data structure *intermediate representation*, and extracting a descriptor from it is efficient and is achieved through a single max-pooling operation. We show that this architectural change does not impact the descriptor's performance when evaluated on a standard benchmark for local image descriptors and at the same time results in significant savings in memory complexity in single-image tasks. As a proof of concept, we integrate our descriptor into an inpainting algorithm and qualitatively evaluate its performance when applied to the virtual restoration of master paintings.

  This research resulted in three conference papers [Žižakić 19a,Žižakić 19b, Meeus 20].

- *Important insights into autoencoders for learning local image descriptors.*

  Despite the fact that autoencoders and variational autoencoders are popular self-supervised learning methods, there are several gaps in the literature on matters concerning how to effectively train and properly evaluate them. Moreover, the literature lacks thorough comparisons between autoencoders and variational autoencoders for learning local image descriptors. In this thesis, we address these knowledge gaps and present our findings regarding the possible approaches to the training and evaluation of autoencoders. We perform a thorough comparative analysis of these two neural network architectures along with an in-depth analysis of the most relevant hyperparameters to guide their optimal selection. In addition to this analysis, we provide insights into the challenges and the importance of selecting right evaluation techniques during the self-supervised learning of the local image descriptors and propose a rapid approximate evaluation method for descriptors learned with autoencoders which is strongly correlated with established benchmarks.

  This research resulted in a journal article [Žižakić 21a] and a conference paper currently under review [Žižakić 22].

- *A framework for transferring the knowledge from hand-crafted to learning-based local image descriptors.*

  An interesting problem that has not been explored before (to the best of our knowledge) is the problem of using the hand-crafted features as a starting point from which a neural network can further be trained, in order to give its training a 'head start' and with the potential benefit of allowing for higher explainability of the models.

  To that end, we developed a framework for enabling knowledge transfer from hand-crafted to learning-based descriptors. This framework aims to achieve the performance of learning-based descriptors while preserving some benefits of hand-crafted descriptors – the descriptor learned using this framework has improved explainability (inherited from the hand-crafted descriptor) and can be fine-tuned for specific image processing applications without the need for a labelled dataset. These characteristics are not true of neither the hand-crafted nor the supervised-learning–based descriptors. We demonstrate the use of this framework by creating the *learned BRIEF* descriptor (based on the BRIEF hand-crafted descriptor [Calonder 10]). To achieve this, we propose an elegant implementation of BRIEF as a convolutional neural network.

  This research resulted in one conference paper [Žižakić 20a].

- *A local image descriptor that allows for inverting patch encodings back onto the original patches.*

  Inverting local image descriptors has been an active area of research in the past decade [Weinzaepfel 11, d'Angelo 12, Vondrick 13, Mahendran 15]. In this thesis, we contribute an efficient method for learning local image descriptor and its inversion function using a β-variational autoencoder. We examine different values of β in the loss function of the β-VAE to find the optimal balance between incentivising the similarities between input patches to be preserved in latent space, and ensuring good reconstruction of the patches from their encodings in latent space. Our proposed descriptor demonstrates patch retrieval comparable to the reference autoencoder-based local image descriptor and shows improved reconstruction of patches from their encodings.

  This research resulted in a conference paper [Žižakić 20b] and a journal article [Žižakić 21b].

- *A self-supervised deep image hashing method that utilises variational autoencoders to improve the compression and explainability aspects of image hashing.*

  Our proposed deep hashing method is based on twin-bottleneck hashing that improves both bottlenecks using recent insights in the field of variational autoencoders. In the binary bottleneck we change the generation of hash codes to be based on variational autoencoder, thus promoting learning of disentangled variables and allowing us to omit the regulariser, therefore simplifying the model. In the continuous bottleneck we employ

a variational autoencoder that is trained using a constrained optimisation setup, in order to better control the trade-off between compression and reconstruction quality of generated samples. Both of these improvements result in better hashing performance separately, and an even better result when applied together. The results have been thoroughly validated on all hash-code sizes (16-bit, 32-bit and 64-bit) and datasets (CIFAR and MS-COCO), showing that our method outperforms the state of the art in deep hashing without labelled data.

This research resulted in one conference paper [Verwilst 21].

- *Case study on efficient search of self-similar structures in electron microscopy images.*

  We demonstrate the potential of the application of our developed image dimensionality reduction methods by searching for regions of images that contain biological structures similar to the structure on the query image patch. We are using local image descriptors to encode the patches and accelerate the search process. The first steps that have been taken for this application show promising results.

  We are currently preparing a journal paper based on this research.

1.5 Publications

The work presented in this thesis has so far led to two journal publications as the first author (one included in the Web of Science and another in a peer-reviewed international journal), one journal publication in preparation, and seven publications in the proceedings of international conferences, five of which as the first author (one currently in review).

In addition to these publications, the source code and documentation for all publications has been made open source and is available on GitHub[1].

1.5.1 Publications in international journals

- **Nina Žižakić** and Aleksandra Pižurica. *Efficient local image descriptors learned with autoencoders.* IEEE Access, vol. 10, pages 221-235, 2021. [Žižakić 21a]

- **Nina Žižakić** and Aleksandra Pižurica. *β-variational autoencoders for learning invertible local image descriptors.* Image Processing & Communications, vol. 24, no. 1, pages 71-78, 2021. [Žižakić 21b]

- Zeno Dhaene, **Nina Žižakić**, Shaoguang Huang, Xian Li and Aleksandra Pižurica. *HSIToolbox: a web-based application for the classification of hyperspectral images.* SoftwareX, 2022 (in preparation). [Dhaene 22]

[1] https://github.com/nimpy

1.5.2 Publications in international conferences

- **Nina Žižakić**, Izumi Ito, Laurens Meeus and Aleksandra Pižurica. *Autoencoder-learned local image descriptor for image inpainting.* In BNAIC/BENELEARN 2019, volume 2491, 2019. [Žižakić 19a]
- **Nina Žižakić**, Izumi Ito and Aleksandra Pižurica. *Learning local image descriptors with autoencoders.* In International Conference on Image Processing and Communications, pages 214-221. Springer, 2019. [Žižakić 19b]
- **Nina Žižakić** and Aleksandra Pižurica. *Learned BRIEF – transferring the knowledge from hand-crafted to learning-based descriptors.* In 2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP). IEEE, 2020. [Žižakić 20a]
- **Nina Žižakić** and Aleksandra Pižurica. *Invertible local image descriptors learned with variational autoencoders.* In IEICE Information and Communication Technology Forum (ICTF) 2020, Proceedings. IEICE Europe Section, 2020. [Žižakić 20b]
- Laurens Meeus, Shaoguang Huang, **Nina Žižakić**, Xianghui Xie, Bart Devolder, Hélène Dubois, Maximiliaan Martens and Aleksandra Pižurica. *Assisting classical paintings restoration: efficient paint loss detection and descriptor-based inpainting using shared pretraining.* In Optics, Photonics and Digital Technologies for Imaging Applications VI, volume 11353, page 113530H. International Society for Optics and Photonics, 2020. [Meeus 20]
- Maxim Verwilst, **Nina Žižakić**, Lingchen Gu and Aleksandra Pižurica. *Deep image hashing based on twin-bottleneck hashing with variational autoencoders.* In 2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP). IEEE, 2021. [Verwilst 21]
- **Nina Žižakić** and Aleksandra Pižurica. *How to efficiently evaluate autoencoders for learning local image descriptors.* European Signal Processing Conference (EUSIPCO), 2022 (in review) [Žižakić 22].

1.6 Outline of the thesis

The structure of the remainder of this thesis is summarised over the following paragraphs.

Chapter 2 introduces autoencoders – the self-supervised deep learning method that is used throughout this thesis. The chapter starts with an overview of deep learning in general, before introducing some common concepts, terms and notations regarding deep neural networks. We then briefly discuss the history of autoencoders and present a summary of the different types of autoencoders with a special emphasis on classical and variational autoencoders (since they are used throughout this thesis). The chapter concludes with a discussion about the applications of autoencoders with an emphasis on learning local image descriptors and learning to hash images.

Chapter 3 investigates different aspects of learning local image descriptors using autoencoders. After presenting a review of the literature on methods for designing both hand-crafted and learning-based local image descriptors, we present different methods for evaluating local image descriptors. We first discuss the available benchmarks for descriptors before proposing a rapid approximate evaluation method for local image descriptors learned with autoencoders. Our results in this chapter provide valuable insights into different evaluation metrics that can be used for this task. Furthermore, we compare autoencoders and variational autoencoders to see which architecture is the most suitable for the task of learning local image descriptors and we perform a comprehensive analysis of the most important hyperparameters. Finally, we conduct research into invertibility of local image descriptors and propose a method for learning the local image descriptor specifically designed to be inverted.

Chapter 4 addresses the problem of designing local image descriptors specifically made for applications with single-image tasks. We propose a novel autoencoder architecture that yields a data structure that we call *intermediate representation*, which is a compact way of storing the descriptors of all the patches of an image. Intermediate representation enables fast retrieval of patch encodings without the memory requirements to store all the encodings separately. We show that this method does not sacrifice the descriptor's performance in order to achieve these computational memory and time improvements. We evaluate our method on a standard benchmark and compare its performance with other self-supervised-learning–based descriptors and with supervised-learning–based and hand-crafted descriptors. As a proof of concept, we integrate this method into an inpainting algorithm and show both qualitative improvements and an ability to handle higher-resolution images.

Chapter 5 examines the strengths and weaknesses of hand-crafted and learning-based local image descriptors and explores whether it is possible to achieve the strengths of both in a single descriptor. To that end, we propose in this chapter a framework for knowledge transfer from hand-crafted to learning-based descriptors that is based on autoencoders and that requires no labelled data. We utilise this framework to implement what we call *learned BRIEF* descriptor (based on the BRIEF descriptor by Calonder et al. [Calonder 10]). We show that our learned BRIEF consistently outperforms the original BRIEF descriptor, serving as a proof of concept for our knowledge transfer framework.

Chapter 6 focuses on image hashing using autoencoders. We give an overview of image hashing methods, both for the traditional image hashing and for the deep hashing, offering an explanation as to why the latter methods have had more success in recent literature. As part of the literature review, we introduce an important recent deep hashing method called twin-bottleneck hashing. We then propose a novel deep hashing method that builds upon twin-bottleneck hashing by improving both bottlenecks. Firstly, in the binary bottleneck, we change the generation of hash codes to be based on a variational autoencoder with disentangled variables and we omit the regulariser. Secondly, in the continuous bottleneck, we employ a variational autoencoder that is trained using a constrained optimisation setup to better control the trade-off between compression and reconstruction quality of generated sam-

ples. We discuss these design choices in detail and show through extensive experiments that these design choices lead to state-of-the-art performance. We showcase our deep image hashing method by employing it for content-based patch retrieval on electron microscopy images, where we observe that retrieved image patches correspond to the query patch in terms of the tissue type.

Chapter 7 concludes this thesis with a summary of the most significant results of this research and provides potential directions for future work.

2

Autoencoders

I will take the Ring, though I do not know the way.
—Frodo Baggins, from *The Lord of the Rings: The Fellowship of the Ring*

Over the past decades of artificial intelligence, autoencoders have played an indispensable role in the field of deep learning. They have been used for dimensionality reduction, for feature learning and more recently, for generative modelling. They also form the basis of the research presented in the following chapters of this thesis.

This chapter provides an introduction to the general theory behind deep learning and artificial neural networks as well as a formal overview of autoencoders themselves. We first introduce the preliminaries and most important concepts of deep learning methods, after which we discuss a brief history of autoencoders, describe their different types, and give an overview of the applications of autoencoders.

2.1 Deep learning for image processing – preliminaries

Artificial neural networks, in the following referred to simply as neural networks, are the cornerstone of deep learning, being the foundation of many modern AI methods. The goal of a neural network is to learn to approximate some useful function f^*. A neural network defines a mapping $y = f(x;\theta)$ and is trained to learn parameters θ that result in the best approximation of f^*. The learned function $f(x;\theta)$ is constrained by its specific architectural choice, namely (typically feedforward) layers where information flows from one layer to the next, building the complex function from simpler primitives (neurons).

Besides the neural network itself, for its training we also need a dataset on which it will be trained, an optimisation procedure that will be used to train it, and a loss function that will be minimised during the optimisation procedure.

The choice of a neural network comes down to the choice of its architecture. They are typically composed of many functions, known as layers in the deep learning terminology. For example, we might have functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ connected in a chain, forming a neural network $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$ with

depth 3. In deep learning terminology, drawing inspiration from neuroscience, the inputs and outputs to the layers are called neurons or activation units. When implementing these neural networks, these are simply numbers (usually between 0 and 1), where higher numbers indicate higher "activation" of this neuron. We can divide these functions into two categories: layers and activation functions. In the next section we give brief overview and describe the most common types of layers and activation functions used in neural networks for image processing tasks.

Concerning the choice of an optimisation algorithm, according to the recent review paper [Schmidt 21], the most commonly used ones are Adadelta [Zeiler 12] and Adam [Kingma 14], which are also used in this thesis.

The choice of the dataset is, in general, dependant on the application for which the neural network is going to be used. For example, if we want a neural network to learn to classify an image of a hand-written digit, we might want to use the MNIST dataset [LeCun 10] of labelled images of hand-written digits. In case we want our neural network to learn to classify objects found in natural images in general (e.g. a balloon or a strawberry), we might want to use ImageNet dataset [Deng 09] of more than 14 million labelled images (labels indicate what object(s) are pictured in an image).

With datasets, we also distinguish whether they consist of labelled data or not. If the data is labelled (e.g. for each image of a hand-written digit, we have information about which digit is in the image), we would typically use **supervised learning**, where the neural network learns to output the label assigned to each data sample, i.e., it learns a mapping between the input x (data sample) and its output y (label). On the other hand, for unsupervised or self-supervised neural networks, we do not need a labelled dataset. In the case of **unsupervised learning**, we do not learn a mapping, but typically only assume an input space consisting of data samples x for which to learn the inherent structure, without any y. An example of unsupervised learning is clustering. In the case of **self-supervised learning**, the neural network does learn a mapping, but it does not require a labelled dataset. For example, in the case of autoencoders, the mapping that is being learned is $x \to x$ (with some constraint on the network in order to learn to extract the most useful features of data).

In recent years, many leading researchers in the field, among which notably Andrew Ng, have been advocating for an approach to deep learning that puts more focus on the choice of the dataset, involving systematically improving the dataset in order to increase the accuracy the neural network. In order to measure the quality of different datasets, one could compare their performance on a fixed neural network architecture (usually taking one of the state-of-the-art architectures for a given task). While this approach is out of the scope of this thesis, it is worth noting this shift in deep learning trends.

2.1.1 Layers

We discuss the most common neural network layers that are used for image processing tasks and also throughout this thesis.

Fully-connected layer. This is the simplest type of neural network layer, as its name suggests, each neuron of its input is connected to each neuron of its output. Formally, the output of the i-th neuron of a fully-connected layer is calculated as

$$y_i = \sum_{u=1}^{n} (w_{iu} x_u + b_i),$$
$$\mathbf{x} = \{x_1, ..., x_n\}, \mathbf{W} = \begin{bmatrix} w_{11} & \dots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{n1} & \dots & w_{nn} \end{bmatrix}, \mathbf{b} = \{b_1, ..., b_n\} \tag{2.1}$$

where $\mathbf{x}$ is the input vector to the layer, $\mathbf{W}$ and $\mathbf{b}$ are the learned parameters of this layer (the weight matrix and the bias vector respectively), and n is the size of the layer, i.e., the number of output neurons in the layer (Figure 2.1). By performing this calculation for each output neuron of the layer, we get its output vector $\mathbf{y}$ (Figure 2.2).

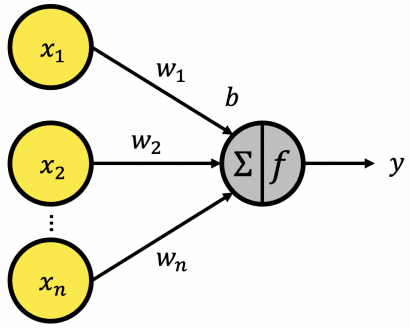


Figure 2.1: A single neuron in a fully-connected layer of a neural network.

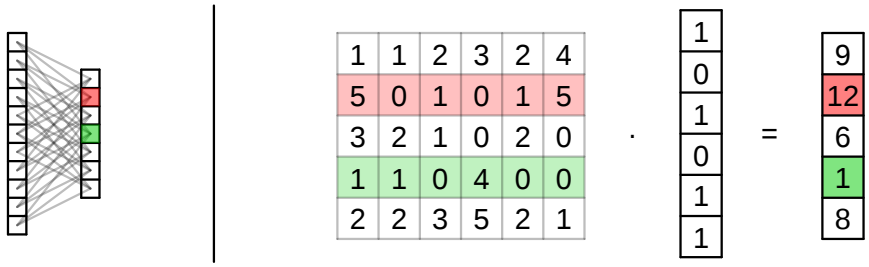


Figure 2.2: Fully-connected layer. Left: standard visualisation of this layer, inputs and outputs are shown as a column vectors. Right: an example showing the operation that the fully-connected layer performs.

One major criticism of fully-connected layers is that they do not take advantage of the fact that neighbouring pixels in natural images tend to be close

in value. Convolutional layers, which we will introduce next, are able to take advantage of that, while also reducing the amount of parameters to be trained. Modern neural networks for image processing tasks consist mostly of convolutional and max-pooling layers, which come with a several benefits for such tasks that we will discuss next.

Convolutional layer. There are two main ideas behind the convolutional layers that differentiate them from fully-connected layers: local connectivity and parameter sharing. Local connectivity means that the output neurons will be connected only to the input neurons that are within a small window in the spatial dimension. Parameter sharing refers to the fact that the parameters for all these windows (in the convolutional neural network terminology known as kernels) are the same. These two properties of convolutional layers imply several beneficial properties. The fact that the parameters are only local and are shared means that the convolutional layers have significantly less parameters than their fully-connected counterparts for the same input and output dimensions. This implies better performance in terms of computational memory and time. Secondly, it also implies a certain degree of equivariance to the translation of these layers, which is a desirable property for signal processing tasks. Furthermore, it allows for inputs of any size, which is a very useful property in these tasks – for example, having to have an image of a certain size in order to classify it would be impractical. Finally, convolutional architecture mimics the way our brains work when it comes to processing these inputs, we can think of each convolutional layer as taking an input and calculating a bit more abstract features. For example, when it comes to object recognition, the first layer of a CNN might extract edges, the second layer corners and contours, the third layer object parts, and then this would be used to classify the object. Here we formally define the most common type of convolutional layer, the two-dimensional convolutional layer, which is used in image processing neural networks (including neural networks in this thesis). Let $\mathbf{x}$ be the input tensor, an image. It has two spatial dimensions and a third dimension that represents different image channels (for a colour image, red, green and blue). We want to calculate the output tensor $\mathbf{y}$. The output neuron at location (i, j) in the k-th channel of the convolutional layer is calculated as

$$y_{ij}^{(k)} = \sum_{c \in C} \sum_{u=1}^{n} \sum_{v=1}^{n} w_{uv}^{(k,c)} x_{(i+u)(j+v)}^{(c)} + b^{(k)}, \tag{2.2}$$

where C is the set of input channel indices, $w^{(k,c)}$ are the weights of the convolutional kernel for the k-th channel applied on the c-th input channel, $b^{(k)}$ is the bias for the k-th channel, and $n \times n$ is the size of the convolutional kernel for this layer. This operation slightly differs from the mathematical operation convolution (of discrete functions) – here the kernel is not flipped, which is equivalent of the operation of cross-correlation, as is shown in Equation 2.2.

Max-pooling layer. This type of layer often comes after one or more convolutional layers. There are several motivations for using max-pooling – it provides non-linearity, it plays the role of dimensionality reduction, and reduces the number of training parameters (and hence the training time). Furthermore,

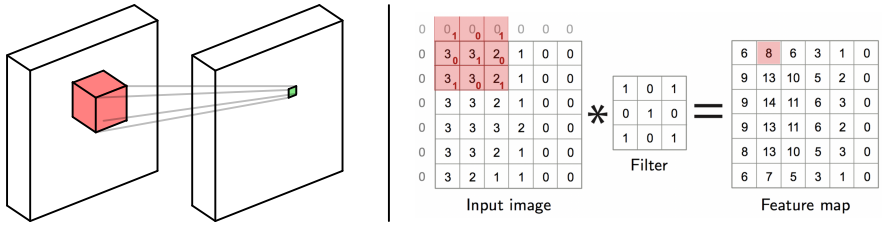


Figure 2.3: Convolutional layer. Left: standard visualisation of this layer, inputs and outputs are shown as a cuboids. Right: an example showing the operation that the convolutional layer performs.

it also enables the extraction of translation-invariant features. For an input tensor $\mathbf{x}$, having two spatial dimensions and a channel dimension, max-pooling is performed over each channel as follows:

$$y_{ij} = \max\left(x_{(pi+u)(pj+v)}\right), \tag{2.3}$$

with $u \in [0, p], v \in [0, p]$, where p is the max-pooling size. The outputs of all the channels constitute the tensor $\mathbf{y}$, which is the output of the max-pooling layer.

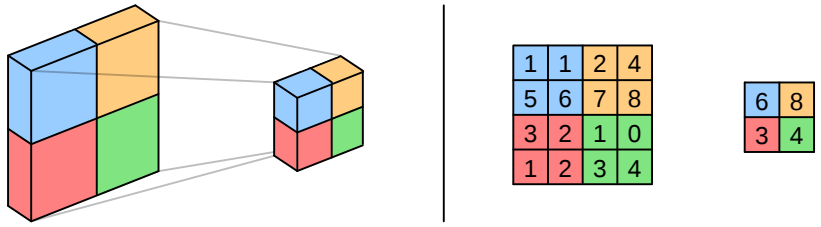


Figure 2.4: Max-pooling layer. Left: standard visualisation of this layer, inputs and outputs are shown as a cuboids. Right: an example showing the operation that the max-pooling layer performs.

2.1.2 Activation functions

It is a standard practice in neural networks to add a non-linear activation functions after a linear layer (such as fully-connected or convolutional layer). The reasoning for this is that the function we are trying to learn is most likely non-linear, therefore using only linear functions for its approximation will not lead to a good approximation. These non-linear activation functions are usually applied element-wise to the output of such a layer. They are typically very simple and fast to compute. Here we list some of the most common activation functions, which will be used later in the thesis.

Sigmoid activation function (also known as the logistic activation function) takes any real value as input and outputs a value between 0 and 1. It is

defined as:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

This function is used very commonly for the last layer of the neural network, when we want the output of the whole network to be between 0 and 1, or that each output neuron is in this range. It is used in binary classification (where the output can be interpreted as the percentage of certainty of the input belonging to the positive class) or when the output of the network is an image (whose pixel values are normalised to be between 0 and 1).

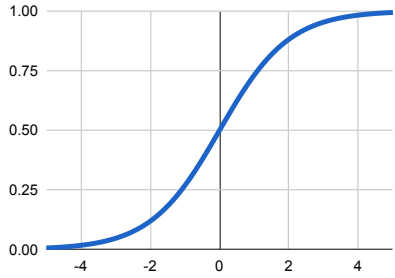


Figure 2.5: Sigmoid activation function

Tanh activation function is similar to the sigmoid activation function, having the same S-shape, with the difference that its output is in the range of -1 to 1. It is defined as:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.5}$$

It is zero-centred, and therefore preferred over the sigmoid activation function for the use after hidden layers, but it is less commonly used than ReLU-based activation functions.

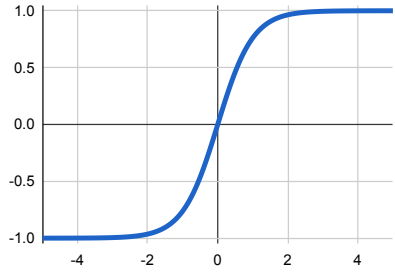


Figure 2.6: Tanh activation function

Rectified Linear Unit (ReLU) activation function is the most commonly used activation function (except for the choice of activation function that goes after the last layer, as was discussed in the paragraph on sigmoid). ReLU is

defined as:

$$f(x) = \max(0, x). \tag{2.6}$$

Applying this function to the output of a linear transformation yields a non-linear transformation. However, ReLU preserves many properties that make linear models easy to optimise with gradient-based methods due to the fact that it is piece-wise linear (with two linear pieces). This activation function is the default activation function recommended for use with most feedforward neural networks. However, its limitation is that for negative numbers, its derivative is zero. Therefore, during the optimisation (i.e. backpropagation), the weights and biases for some neurons are not updated, which can create dead neurons that never get activated. This phenomenon is called the "dying ReLU problem". In order to tackle this problem, several different variants of this function have been proposed, which are described in the following paragraphs.

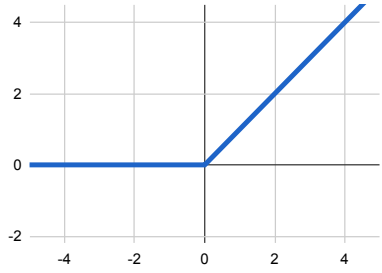


Figure 2.7: Rectified Linear Unit (ReLU) activation function

Parametric Linear Unit (PReLU) activation function is one such version of ReLU where the negative area has a small positive slope. Formally, it is defined as:

$$f(x) = \max(ax, x), \tag{2.7}$$

where a is the slope parameter. The gradient of the negative part of the function is a non-zero value and therefore, dead neurons would no longer be present in that region. However, in practice it is not clear whether this activation function leads to better results than ReLU.

Exponential Linear Unit (ELU) activation function is another version of ReLU that tries to tackle the dying ReLU problem. It is defined as follows:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha(\mathrm{e}^x - 1)) & \text{if } x < 0 \end{cases} \tag{2.8}$$

with α typically taking a value between 0.1 and 0.3. The idea of this function is that by introducing log curve for negative values of input, it helps the network nudge weights and biases in the right direction. Similarly to PReLU, in practice it shows comparable performance to ReLU.

There also exist other types of activation functions, including the smooth variants of ReLU and ELU, which are differentiable at zero (unlike ReLU and ELU). Examples of these include Gaussian-error linear unit (GELU)

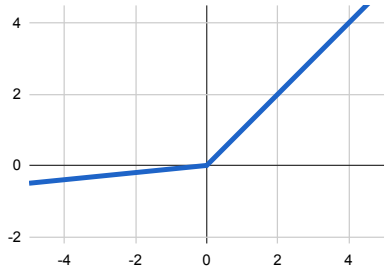


Figure 2.8: Parametric Linear Unit (PReLU) activation function

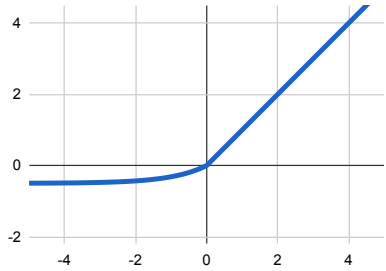


Figure 2.9: Exponential Linear Unit (ELU) activation function

[Hendrycks 16], Mish [Misra 19] and Smooth Activation Unit (SAU) [Biswas 21]. These activation functions show promising results in terms of performance on academic benchmarks. However, they are still not widely used and were not part of standard deep learning libraries at the time when the research in this PhD was conducted, hence they are also not included in the subsequent analysis. However, it can be of interest to include them in the future hyperparameter studies (using the same framework as described in Section 3.3).

2.1.3 Loss functions

Now that we have building blocks for the neural networks that will be used in this thesis, what is left to discuss is loss functions which the neural networks will be trained to minimise. Here we list the loss functions used in neural networks for image processing. These loss functions can all be used to quantify the difference between two images, which is an important property for training autoencoders that work with images, which we do in this thesis. In the case of autoencoders, the loss function will measure the difference between the input and the output image, with the goal of minimising this difference. In the case of variational autoencoders, there is an additional term in the loss function for regularisation, as will be explained in Section 2.4.4.

Mean squared error (MSE) is a very intuitive way of mathematically measuring the difference between two tensors. For two images x and y of size

$M \times N$, it is defined as:

$$\text{MSE}(x, y) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} (y_{ij} - x_{ij})^2, \tag{2.9}$$

One issue with using MSE (and other pixel-wise loss functions) is that it does not take into account relationships between different pixels, and can be heavily influenced even by a small pixel shift. Another problem with MSE is that it can lead to loss saturation ("plateau") when combined with sigmoid activation function, which is the most common activation function of the last layer of the network. This saturation can prevent gradient-based learning algorithms from making progress.

Binary cross-entropy (BCE) is a loss function taken from probability theory, and is commonly used because it avoids the problem of loss saturation by having a log that undoes the exponential in the sigmoid activation function. It is often the default choice for the loss function when the output of a neural network is an image. For two images x and y of size $M \times N$, it is defined as:

$$\text{BCE}(x, y) = -\frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(y_{ij} \log x_{ij} + (1 - y_{ij}) \log(1 - x_{ij}) \right). \tag{2.10}$$

This loss function is also calculated pixel-wise, and is therefore unable to take into account the relationships between pixels.

We now delve into the world of autoencoders, starting with a short overview of their history in the next section.

2.2 A brief history of autoencoders

The concept of autoencoders has been around since the eighties. It is difficult to pinpoint when they were first "born" because the literature is diverse and terminology has evolved over time, but there are three important points in time that mark the birth of autoencoders – when their concept was first presented, when the word autoencoder was coined, and when they gained fame and widespread use. The first time that the concept of teaching an artificial neural network to learn a useful encoding by training its output to be the same as input – the concept that we now call autoencoders – was mentioned (albeit indirectly) is in the paper by Rumelhart, Hinton and Williams in 1985 [Rumelhart 85]. They refer to it as "the encoding problem", and the idea was to encode 8 bits of data using a very simple neural network with only one hidden layer (Figure 2.10, left). The concept of autoencoders exists in the papers after that, but not under that name (for example, in [Kramer 91] they are referred to as autoassociative neural networks). The term 'autoencoder' was coined in 1994 [Hinton 94]. However, in a decade that follows, autoencoders and other artificial neural networks were not a trendy topic due to the lack of resources for their training. It was only in 2006 that the autoencoders gained the momentum, starting with the seminal paper (which is often quoted as *the* original

autoencoder paper) that was published in Science Magazine [Hinton 06], where autoencoders got their modern shape and form. We note that Geoffrey Hinton was present in all stages of the birth of autoencoders.

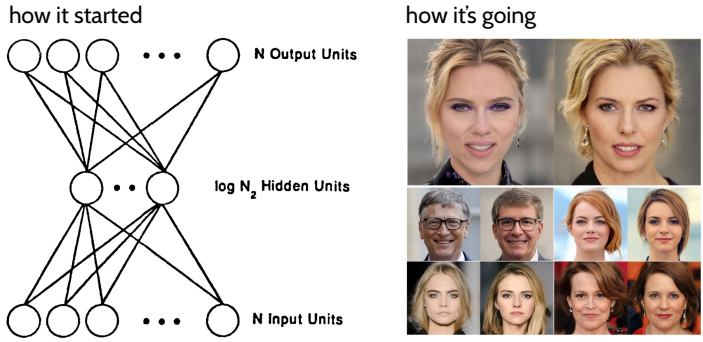


Figure 2.10: Autoencoders then and now. Left: first diagram of the concept of autoencoders at the time encoding a single byte of data [Rumelhart 85]; right: generation of images of faces based on images of celebrities using autoencoders [Pidhorskyi 20]. Images taken from [Rumelhart 85, Pidhorskyi 20].

In the past decade, there has been a lot of research and advancement in the field of deep learning, including in the field of autoencoders. Variational autoencoders proposed by Kingma and Welling [Kingma 13] in 2013 are probably the most important advancement in autoencoders in the last decade, adding a probabilistic model flavour to the autoencoders. Since then and inspired by the variational autoencoders, there have been several other variants of autoencoders proposed [Makhzani 15, Higgins 17, Tolstikhin 17, Pidhorskyi 20], and the (variational) autoencoders started being used as deep generative models, showing impressive results with, e.g. generating realistic images of faces (Figure 2.10, right).

A note on the terminology regarding autoencoders. It should be noted that, in the literature, autoencoders are sometimes referred to as unsupervised methods (including in some foundational and highly cited papers [Hinton 95, Kingma 19], and even in the Wikipedia page on autoencoders, as it currently stands [Wikipedia 22]). This characterisation arose from the intention to emphasise the difference with respect to supervised learning using labelled data, but is too coarse in terms of the common understanding of the concept 'unsupervised learning'. As it was introduced in Section 2.1, unsupervised methods (such as clustering) do not learn a mapping between input and output, but infer patterns or structural properties of the data in other ways. In the case of autoencoders, we do learn a mapping between the input and the output (albeit the output being the same as the input). Hence, although autoencoders do not use labelled data, they do not perform unsupervised learning in the strict sense. Therefore, in this work we do not refer to autoencoders as unsupervised methods but rather self-supervised, i.e. methods that learn a mapping between input and output without labelled data.

In the next section, we give an overview of the applications of autoencoders.

2.3 Applications of autoencoders

Dimensionality reduction is one of the first applications of autoencoders, and it is still an active field of research. It was also one of the first applications of deep learning and one of the early motivations for studying autoencoders, starting with the seminal work by Hinton and Salakhutdinov [Hinton 06].

One example of the use of dimensionality reduction is for finding a compact representation of a patch in an image, i.e., finding a local image descriptor. Local image descriptors are used in image processing tasks such as image denoising, inpainting, object tracking and motion estimation, where we employ them to identify corresponding parts of images. Autoencoders can be used here to encode image patches into these lower-dimensional representations which are invariant to a certain degree of geometric noise (translation, scaling, shearing). We research this application of autoencoders in chapters 3-5.

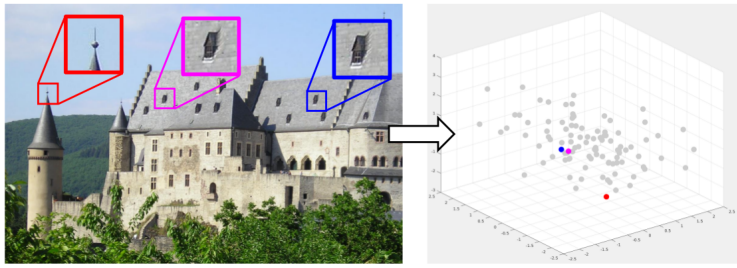


Figure 2.11: Dimensionality reduction – local image descriptors

Another use of dimensionality reduction is for information retrieval, the task of finding data in a database that resemble a query data sample. The most prominent type of information retrieval is content-based image retrieval, where images are retrieved based on visual and semantic similarity to the query image. The encoding of semantic information (semantic hashing) can be achieved with autoencoders, as we study in Chapter 6.

In the past, autoencoders have also been used for self-supervised pretraining of deep neural networks. For example, before training a deep neural network to classify images from a dataset, one would train an autoencoder on the same dataset and then use the learned weights of its encoder to initialise the classifier network. The idea was that autoencoders would learn useful features and thus facilitate faster and better training of the classifier. Nowdays, this application of autoencoders is not very common.

A recent application of autoencoders (more specifically, variational autoencoders) is for generative modelling. The idea is that, after training a VAE to encode and decode data from the learned latent space, we can also sample the latent space to generate new realistic data samples. This line of study is beyond the scope of this thesis.

In the following sections, we will have a closer look at these different types of autoencoders and define them in a formal way.

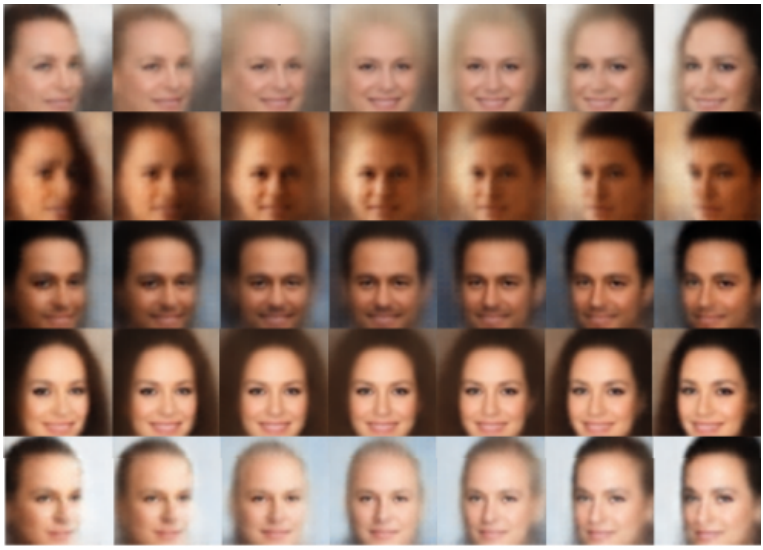


Figure 2.12: Image generation with β-variational autoencoders. Image taken from [Higgins 17].

2.4 Types of autoencoders

Autoencoders (AEs) [Hinton 06] are self-supervised neural networks used for learning compact representations of data. An autoencoder consists of two parts, an encoder and a decoder, both of which can be seen as a neural network. An autoencoder is trained by setting the target output values to be equal to the input values, while imposing a constraint on the middle layer.

There are different types of autoencoders that are distinguished based on the type of constraint (undercomplete autoencoders, sparse autoencoders), whether the autoencoder is a probabilistic model or not (variational autoencoders), based on whether the input to the autoencoder is corrupted or not (denoising autoencoders). Here we describe the most prominent types of autoencoders.

2.4.1 Classical autoencoders

The most common, default type of autoencoder is autoencoder that imposes only dimensionality constraint on the middle layer. These autoencoders are sometimes referred to as undercomplete. In this thesis we refer to them as classical autoencoders or simply autoencoders.

Formally, an autoencoder consists of an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$ which are parametrised with their weights $\mathbf{w}_\mathcal{E}, \mathbf{w}_\mathcal{D}$ and biases $\mathbf{b}_\mathcal{E}, \mathbf{b}_\mathcal{D}$, respectively. The autoencoder is trained to minimise the loss function J w.r.t. $\mathbf{w}_\mathcal{E}, \mathbf{b}_\mathcal{E}, \mathbf{w}_\mathcal{D}, \mathbf{b}_\mathcal{D}$:

$$\min_{\mathbf{w}_\mathcal{E}, \mathbf{b}_\mathcal{E}, \mathbf{w}_\mathcal{D}, \mathbf{b}_\mathcal{D}} J(\mathcal{X}, \mathcal{E}_{\mathbf{w}_\mathcal{E}, \mathbf{b}_\mathcal{E}}, \mathcal{D}_{\mathbf{w}_\mathcal{D}, \mathbf{b}_\mathcal{D}}), \tag{2.11}$$

i.e. to minimise the following expression:

$$\min_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E},\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}} \sum_{\mathbf{x}\in\mathcal{X}} \mathcal{L}(\mathbf{x}, \mathcal{D}_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}}(\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}(\mathbf{x}))) \tag{2.12}$$

where $\mathbf{x} \in \mathcal{X}$ is a data sample (e.g. an image) and $\mathcal{L}$ is some loss function. Autoencoders working with image data usually consist of convolutional layers, with an optional fully-connected layer at the end of the encoder and the beginning of the decoder. The encoder of a classical autoencoder usually consists of convolutional layers, each followed by a max-pooling layer, with an optional fully-connected layer at the end. The decoder normally mirrors the encoder, with convolutional–max-pooling combo being replaced with fractionally-strided convolutional layers, thus enlarging the size of their output.

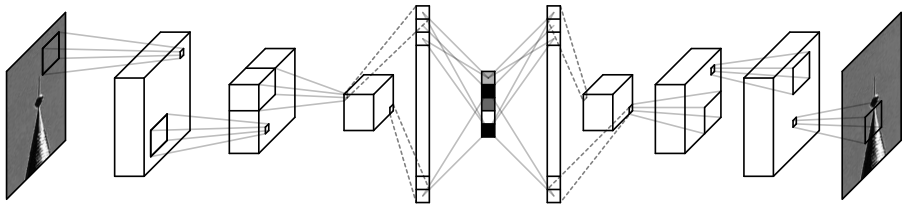


Figure 2.13: Classical autoencoder architecture

The idea of such an autoencoder is that, since the middle layer does not have the capacity to store the whole data samples, it needs to focus on their most important features – therefore the autoencoder learns to extract these features.

In practice this works well, however, theoretically it is possible that an autoencoder of this type simply "memorises" the whole training dataset without learning any useful features. For example, an autoencoder with a one-dimensional code could learn to represent each data sample $\mathbf{x}_i$ with its index i. Despite the fact that this does not happen in practice, different variants of an autoencoders have been proposed that try to tackle this weakness of classical autoencoders. In the following sections we discuss some of them.

2.4.2 Sparse autoencoders

Instead of having a dimensionality constraint imposed on the middle layer, it is also possible to impose a sparsity constraint. An autoencoder with such a constraint is called sparse autoencoder [Ng 11]. Sparse autoencoders are trained to minimise the loss function w.r.t. $\mathbf{w}_\mathcal{E}, \mathbf{b}_\mathcal{E}, \mathbf{w}_\mathcal{D}, \mathbf{b}_\mathcal{D}$:

$$\min_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E},\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}} \sum_{\mathbf{x}\in\mathcal{X}} \mathcal{L}(\mathbf{x}, \mathcal{D}_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}}(\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}(\mathbf{x}))) + \Omega(\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}(\mathbf{x})) \tag{2.13}$$

where $\mathbf{x} \in \mathcal{X}$ is a data sample, $\mathcal{L}$ is some loss function, and Ω is the sparsity penalty that is imposed on the middle layer.

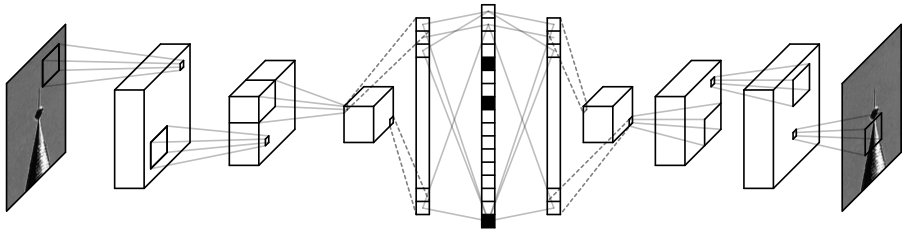


Figure 2.14: Sparse autoencoder architecture

Sparse autoencoders are typically used to learn features for another task, e.g. image classification. The sparsity constraint forces the autoencoder to respond to unique statistical features of the dataset it has been trained on (rather than simply copying input to the output like an identity function). Learning to perform the copying task with a sparsity penalty can therefore yield an autoencoder that has learned useful features as a byproduct. However, in recent years, sparse autoencoders have not been widely used. This is largely due to the fact that they are mostly used for self-supervised pre-training, and less so for dimensionality reduction (unlike other types of autoencoders which are used extensively for dimensionality reduction). For this reason, there exist fewer implementations of sparse autoencoders (using some popular deep learning libraries such as PyTorch and Keras) than those of e.g. variational or classical autoencoders, which further slows their practical adoption.

2.4.3 Denoising autoencoders

A denoising autoencoder takes yet another approach to ensure that the autoencoder learns a useful function. A denoising autoencoder takes a corrupted data sample (e.g. an image to which the noise has been artificially added) and learns to reconstruct the original. The idea behind this is that by learning to undo the corruption, the denoising autoencoder implicitly learns the structure of data [Alain 14, Bengio 13].

Denoising autoencoders are trained to minimise the loss function w.r.t. $\mathbf{w}_{\mathcal{E}}, \mathbf{b}_{\mathcal{E}}, \mathbf{w}_{\mathcal{D}}, \mathbf{b}_{\mathcal{D}}$:

$$\min_{\mathbf{w}_{\mathcal{E}}, \mathbf{b}_{\mathcal{E}}, \mathbf{w}_{\mathcal{D}}, \mathbf{b}_{\mathcal{D}}} \sum_{\mathbf{x} \in \mathcal{X}} \mathcal{L}(\mathbf{x}', \mathcal{D}_{\mathbf{w}_{\mathcal{D}}, \mathbf{b}_{\mathcal{D}}}(\mathcal{E}_{\mathbf{w}_{\mathcal{E}}, \mathbf{b}_{\mathcal{E}}}(\mathbf{x}))) \tag{2.14}$$

where $\mathbf{x} \in \mathcal{X}$ is a data sample and $\mathbf{x}'$ the data sample with added noise and $\mathcal{L}$ is some loss function.

2.4.4 Variational autoencoders

A pitfall of classical autoencoders is that they have no way of enforcing the continuity of the latent space and are thus unable to guarantee that the learned encodings are useful, e.g., that encodings of similar inputs are similar themselves (see Figure 2.16 (left)). Furthermore, when sampling from the latent

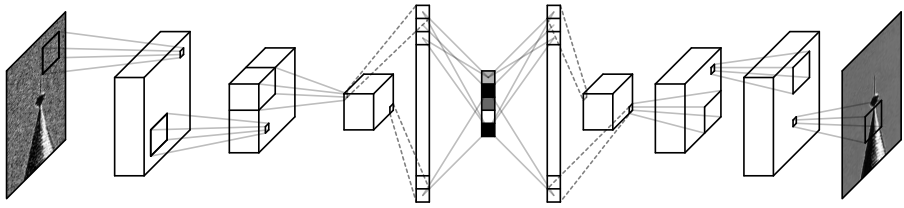


Figure 2.15: Denoising autoencoder architecture

space and then reconstructing the encoding back into the data space, we have no guarantee that the reconstruction will not look like random noise. Being able to reconstruct a randomly sampled encoding into the data space and obtain a data sample that appears to belong to the original dataset is a desired property that classical autoencoders to not possess.

To tackle these problems, Kingma et al. have proposed variational autoencoders (VAEs) [Kingma 13, Kingma 19]. Similarly to classical autoencoders, VAEs consist of an encoder and a decoder, with a middle layer on which a dimensionality constraint is imposed. In contrast to classical autoencoders, however, variational autoencoders are probabilistic models that assume a prior distribution of the latent space, giving significant control over how we want to model the latent distribution.

We start from a simple probabilistic graphical model (PGM) shown in Figure 2.17 (a). Let us consider some dataset $\mathbf{X} = \{\mathbf{x}^{(i)}\}, i = 1..N$, which consists of N i.i.d. samples of some variable $\mathbf{x}$. We can think of the data as lying on a manifold and assume it is being generated from some latent variable $\mathbf{z}$. We can only observe $\mathbf{x}$, but we want to infer the characteristics of $\mathbf{z}$, i.e. we want to compute $p_\theta(\mathbf{z}|\mathbf{x})$ that approximates the underlying conditional distribution $p^*(\mathbf{z}|\mathbf{x})$. Applying the Bayes' theorem, we obtain the following:

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})} \tag{2.15}$$

The issue that we run into is that, often, the integral of the marginal likelihood $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})\, d\mathbf{z}$ is intractable, and therefore, we cannot evaluate or differentiate the marginal likelihood. This implies that the posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is also intractable. Such intractabilities appear in cases of moderately complicated likelihood functions $p_\theta(\mathbf{x}|\mathbf{z})$, e.g. a neural network with a nonlinear hidden layer.

The **variational inference** approach to tackle this is to approximate the posterior $p_\theta(\mathbf{z}|\mathbf{x})$ with another distribution $q_\phi(\mathbf{z}|\mathbf{x})$ from some family Q. In order to approximate $p_\theta(\mathbf{z}|\mathbf{x})$ with $q_\phi(\mathbf{z}|\mathbf{x})$, we want to make these distributions as close as possible. Minimising Kullback-Leibler divergence from $p_\theta(\mathbf{z}|\mathbf{x})$ to $q_\phi(\mathbf{z}|\mathbf{x})$ is often used for this purpose:

$$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) = \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} \tag{2.16}$$

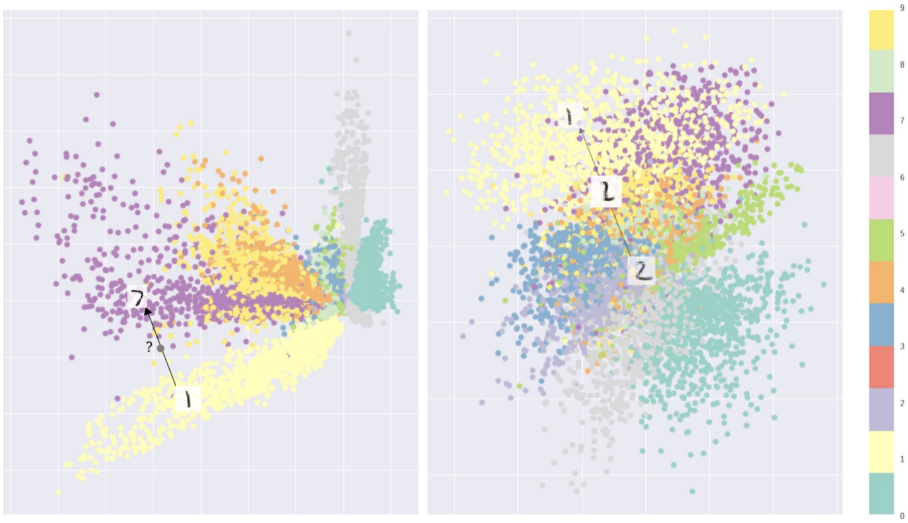


Figure 2.16: Two-dimensional latent space of MNIST dataset [LeCun 10] learned by an autoencoder (left) and variational autoencoder (right). In the autoencoder case (left), there are gaps between the encoding of some classes of digits (e.g. classes 1 and 7). If we were to sample from a gap as shown in the image, and then reconstruct this encoding back onto the original data space, we would have no guarantee that the resulting image would look like a digit. On the other hand, with variational autoencoders (right), sampling between the encodings of two data samples (digits 1 and 2 in this case) will result in a data sample that looks like a digit that is somewhere between 1 and 2. A VAE, unlike an AE, has a smooth latent space, as can also been seen in Figure 2.20.

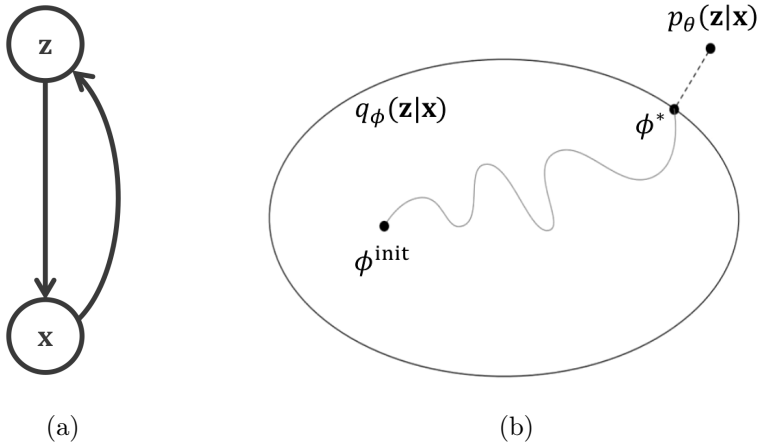


Figure 2.17: Variational autoencoders as probabilistic graphical models. (a) Probabilistic graphical model from which VAEs are built upon – the hidden variable $\mathbf{z}$ governs the process that generates data $\mathbf{x}$. (b) Approximating the posterior $p_\theta(\mathbf{z}|\mathbf{x})$ with a tractable distribution $q_\phi(\mathbf{z}|\mathbf{x})$ by finding the parameters ϕ such that the Kullback-Leibler divergence from $p_\theta(\mathbf{z}|\mathbf{x})$ to q is minimal.

Regarding the family of distributions Q, Gaussian mixture model is commonly used.

Therefore, we minimise Kullback-Leibler divergence from $p_\theta(\mathbf{z}|\mathbf{x})$ to $q_\phi(\mathbf{z}|\mathbf{x})$ (Figure 2.17 (b)):

$$\begin{aligned} D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} = \\ &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \frac{1}{p_\theta(\mathbf{x})} = \\ &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \left(\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} - \log p_\theta(\mathbf{x}) \right) = \\ &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} + \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}) = \\ &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} + \log p_\theta(\mathbf{x}) \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) = \\ &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} + \log p_\theta(\mathbf{x}) \end{aligned} \tag{2.17}$$

Rearranging Equation 2.17, we obtain the following:

$$\log p_\theta(\mathbf{x}) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) + \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \tag{2.18}$$

The second term on the right-hand side of this equation is called Evidence Lower Bound (ELBO), commonly labelled as $\mathcal{L}_{\theta,\phi}(\mathbf{x})$. ELBO gets its name because it is the lower bound for the evidence $p_\theta(\mathbf{x})$ (because $D_{KL} \geq 0$). When minimising the Kullback-Leibler divergence from $p_\theta(\mathbf{z}|\mathbf{x})$ to $q_\phi(\mathbf{z}|\mathbf{x})$ with respect to parameters ϕ, the left side of Equation 2.18 (the log-likelihood) is a constant. Therefore, minimising the KL divergence is equal to maximising ELBO. Writing out ELBO we obtain:

$$\begin{aligned} \mathcal{L}_{\theta,\phi}(\mathbf{x}) &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} = \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} = \\ &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \left(\log p_\theta(\mathbf{x}|\mathbf{z}) + \log \frac{p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right) = \\ &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}|\mathbf{z}) + \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} = \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) \end{aligned} \tag{2.19}$$

Therefore, ELBO, which we are trying to maximise can be expressed as follows:

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) \tag{2.20}$$

By maximising ELBO, we:

1. Maximise the marginal likelihood (the first term of the right hand side of Equation 2.20), thus improving our generative model (the decoder). This term can be viewed as reconstruction term from the perspective of neural networks, as in classical AEs.

2. Minimise the KL divergence from true posterior $p_\theta(\mathbf{z})$ to the approximation function $q_\phi(\mathbf{z}|\mathbf{x})$ (the second term of the right hand side of Equation 2.20), thus improving the encoder. This term can be viewed as regularisation term from the perspective of neural networks, making sure that the latent space distribution is as close as possible to the true posterior distribution that we have selected (usually Gaussian).

Therefore, starting from a probabilistic graphical model, using variational inference, we arrive at an autoencoder-like structure called variational autoencoder (VAE) [Kingma 13]. For this reason, we say that VAEs marry PGMs and deep learning.

The main benefit of VAEs is that we have control over how we model our latent distribution and can thus ensure a smooth latent space (Figure 2.16 (right)). This property is especially important for the applications where we want to achieve explainability, (e.g. medicine) but also in many other domains, such as in computer graphics (for smooth reconstruction of deformations). Furthermore, VAEs allow us to generate new data samples by sampling from the latent space, and then propagating this sample through the decoder to obtain the reconstruction. If we were to do this with a classical autoencoder, we would not be able to guarantee that the reconstruction sits on the data manifold (as shown in Figure 2.16 (left)). We show some examples of manifolds learned by VAEs in Figure 2.20.

Regarding the choice of approximate posterior $q(\mathbf{z}|\mathbf{x})$ – in theory it can be chosen freely, however, there are some practical considerations that narrow down this choice. In order to be able to efficiently optimise the ELBO, we need to be able to computationally efficiently compute and differentiate its probability density $q_\phi(\mathbf{z}|\mathbf{x})$, and also computationally efficiently sample from it, because both of these operations are performed for each data point in a batch at every iteration of optimisation. In practice, this often leads to choosing a simple Gaussian posterior.

β-variational autoencoders. Higgins et al. [Higgins 17] have proposed a generalisation of a variational autoencoder named β-VAE. In a β-VAE, the ELBO function from Equation (2.20) is modified to add more weight on the second term, thus allowing for a better control of the trade-off between the reconstruction capabilities and the smoothness and disentanglement of the latent space:

$$\mathcal{L}(\theta, \phi; \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \beta D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})). \tag{2.21}$$

2.5 Conclusion and summary

This chapter introduced autoencoders – a type of self-supervised deep neural network that has been relevant since the emergence of the field of deep learn-

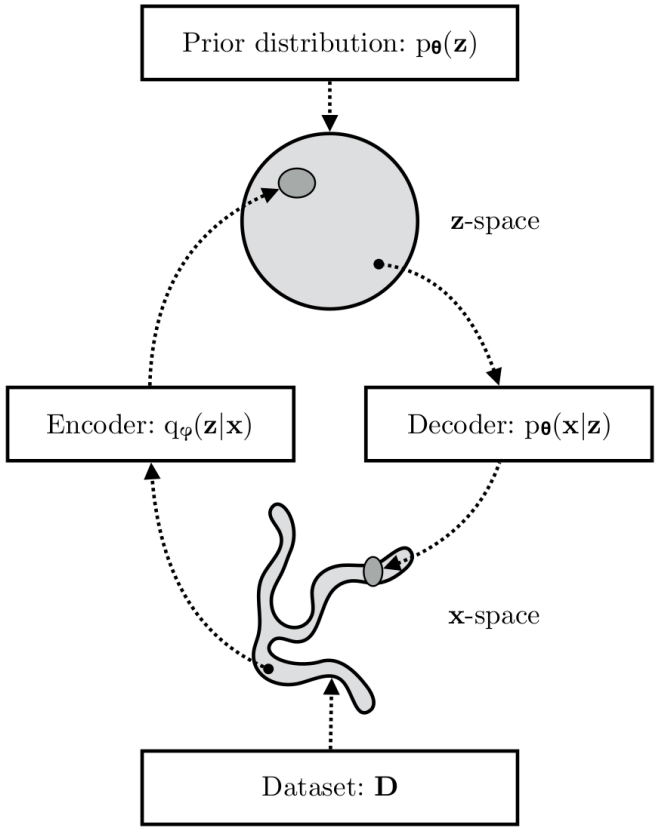


Figure 2.18: Variational autoencoder as a probabilistic graphical model. VAEs learn stochastic mappings between the **x**-space (observed data samples), whose distribution is typically complicated, and a latent **z**-space, whose distribution can be simple (e.g. Gaussian). Illustration taken from [Kingma 19].

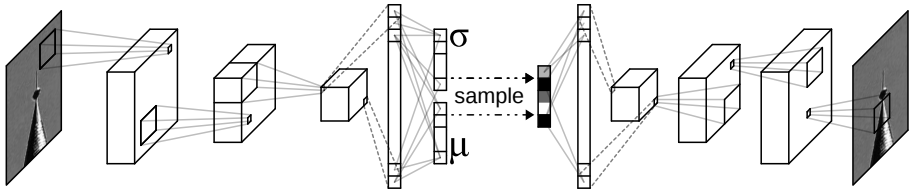


Figure 2.19: Variational autoencoder as a neural network. The encoder and the decoder normally consist of convolutional and max-pooling layers, with a VAE-specific sampling layer in the middle. The mean and standard deviation of the latent distribution are learned, and then sampled in each pass through the network in order to obtain the encoding of the VAE.

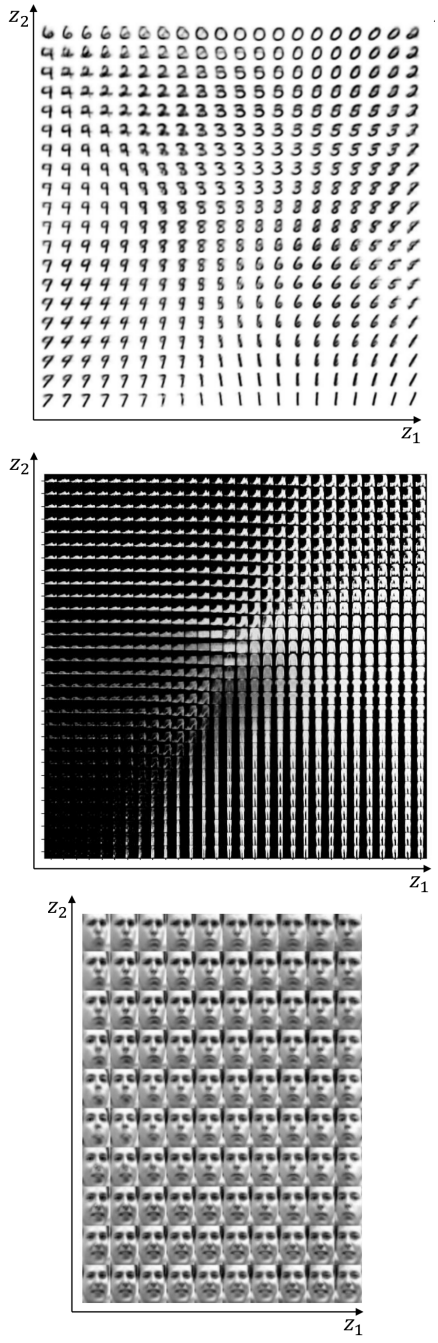


Figure 2.20: Manifolds learned with VAEs for different datasets: MNIST [LeCun 10] (up), Fashion MNIST [Xiao 17] (middle) and Frey Faces (bottom) [Frey 10]. The two-dimensional latent space is sampled equidistantly in a grid-like fashion, and we can observe the smooth transitions between different classes.

ing and that continues to be relevant to this day. There are different types of autoencoders based on their architecture (e.g. the constraint imposed on the middle (encoding) layer) or their stochasticity. Most commonly used types are the classical (undercomplete) AEs, sparse AEs, denoising AEs and variational AEs. Autoencoders have been used in dimensionality reduction problems (such as learning local image descriptors or performing semantic hashing for information retrieval), self-supervised neural network pretraining, and, recently, generative modelling.

3

Learning local image descriptors with autoencoders

It's a dangerous business, Frodo, going out your door. You step onto the road, and if you don't keep your feet, there's no knowing where you might be swept off to.
—Bilbo Baggins, from *The Lord of the Rings: The Fellowship of the Ring*

This chapter focuses on learning local image descriptors in a self-supervised way, using autoencoders and variational autoencoders. Specifically, we perform a thorough comparative analysis of these two approaches along with an in-depth analysis of the most relevant hyperparameters to guide their optimal selection. In addition to this analysis, we give insights into the difficulties and the importance of selecting right evaluation techniques during the self-supervised learning of the local image descriptors. We explore the extent to which a simple perceptual metric during training can predict the performance on tasks such as patch matching, retrieval and verification. Finally, we also perform research into the invertibility of local image descriptors (which is has been an active area of research in the past decade), and propose a descriptor specifically designed for being invertible. In the following chapter, we apply the concepts from this chapter in order to propose an autoencoder-based local image descriptor that allows for memory-efficient storing of patch encodings.

3.1 Introduction and overview of local image descriptors

Finding a compact representation of a small patch in an image, i.e., finding a local image descriptor, is a crucial building block of various image processing tasks. Image inpainting, registration, denoising, stitching, object tracking, motion estimation, and saliency detection are all examples of tasks where local image descriptors are used [Tuytelaars 08, Wang 11, Pernici 13, Patel 16, Nai 18, Joshi 20]. For example, in object tracking and motion estimation,

local image descriptors are used to identify corresponding objects in subsequent frames and to associate those objects between frames. Image inpainting relies on using parts of an image to fill in areas that may be missing or corrupted. By identifying the parts of the image that are the most similar to the parts of the image bordering the missing or corrupted areas, local image descriptors can significantly improve the performance of an inpainting algorithm.

Traditionally, local image descriptors were designed to use hand-crafted features. The most prominent kind of hand-crafted descriptors used to be the distribution-based descriptors, which are using distributions of image properties such as gradients to represent patches. They are sometimes referred to as the SIFT-based descriptors [Balntas 17], after the paramount work by Lowe [Lowe 99], one of the most cited papers in computer science. Other prominent hand-crafted descriptors include HOG [Dalal 05], SURF [Bay 08], GLOH [Mikolajczyk 05], PCA-SIFT [Ke 04], RSIFT [Arandjelović 12] and DAISY [Tola 09], to name a few. Some hand-crafted descriptors produce encodings in Hamming space – they are most commonly referred to as binary descriptors. Examples of such descriptors are BRIEF [Calonder 10], ORB [Rublee 11], BRISK [Leutenegger 11], FREAK [Alahi 12] and LDAHash [Strecha 11].

In recent years, a popular approach to solving many image processing challenges, including the design of local image descriptors, has shifted towards the use of deep learning methods. The first learned local image descriptors arose from the necessity to find a way of choosing parameters in the hand-crafted methods that does not involve hand-tuning [Winder 07, Jin 05]. As machine learning, and in particular, deep learning methods developed further, a growing number of learning-based descriptors started emerging. Nowadays, learned descriptors are mostly supervised methods, learning useful features on pairs of similar and dissimilar patches, striving for a high correlation between the similarity of the patches and the similarity of their descriptors. Most recent methods involve the use of convolutional neural networks [Fischer 14a, Han 15, Ono 18, Wang 20], some of which are siamese networks [Zagoruyko 15, Simo-Serra 15], or triplets [Balntas 16, Tian 19].

Nonetheless, despite the apparent improved performance on benchmarks, reportedly, hand-crafted descriptors are still chosen over learned ones in practical applications. One important practical reason is that the most popular computer vision library OpenCV still does not include implementations of any learned descriptors, while containing implementations of several different hand-crafted descriptors (SIFT, SURF, FAST, BRIEF, ORB). For this reason, most researchers or engineers, whose focus is not specifically to improve on the local image descriptors, will choose to use a descriptor provided by OpenCV as it is the simplest and most reliable solution. Using learned descriptors would require finding their implementations and even potentially training them. The question one might pose is: why are there no implementations of learned descriptors in OpenCV? We believe that several factors might be at play here. Firstly, it takes time for an algorithm or method to go from being first published in academic articles to being implemented in standard libraries, and that time may not have passed yet for the learned descriptors. Secondly, the programming language mismatch might also play a role here – OpenCV libraries

are implemented in C and C++, whereas most deep learning models are implemented in Python. And finally, the incentives to implement these models in OpenCV (or other standard libraries) are lacking.

Furthermore, comparative evaluation studies such as [Schonberger 17] suggest that, while learned local image descriptors show better performance on benchmarks, traditional hand-crafted descriptors such as SIFT show better performance when being part of a larger image processing task (such as object tracking or image registration). This has been attributed to the fact that the learned descriptors were trained too generally and thus often underperform on a specific image processing task compared to the pre-designed ones [He 18]. Moreover, since the majority of learning-based approaches are supervised, they require labelled data, which is often unavailable when creating descriptors for specific image processing tasks. An alternative is to turn to self-supervised models such as autoencoders (AEs) [Hinton 06] and variational autoencoders (VAEs) [Kingma 13]. In this section, we study this approach and propose solutions for the most prominent problems of this approach.

Autoencoders are neural networks where an encoder and decoder are simultaneously trained to encode data into and decode data from a compact encoding, as explained in more detail in Chapter 2. In recent years, there has been a surge of AE-based and VAE-based architectures in various computer vision tasks [Kingma 13, Makhzani 15, Higgins 17, Pidhorskyi 20]. However, they have been less studied so far for learning local image descriptors. The only other work on this topic (to the best of our knowledge) is that of Chen et al. [Chen 15] reported an AE-based descriptor showing promising results, however, the methods they used are no longer considered state of the art and they do not explore using variational autoencoders. We proposed in our previous work an approach for learning local image descriptors based on classical autoencoders [Žižakić 19a, Žižakić 19b] and variational autoencoders [Žižakić 20b]. In our experience, both AEs and VAEs have shown promising results for learning local image descriptors, however, a thorough comparative analysis was still missing. On one hand, VAEs facilitate learning a smoother latent space that is easy for interpolation – a property that seems useful for descriptors. On the other hand, VAEs are more complex and more difficult to work with. They introduce new hyperparameters (such as β – the weight of the Kullback-Leibler divergence term in the loss function), take slightly longer on average to train, and are not deterministic.

In this chapter, we compare the performance of AE-based and VAE-based descriptors and consider the advantages and disadvantages of the two approaches. Furthermore, we investigate some novel ways of optimising the performance of the two descriptor types. For example, we explore the use of a perceptual loss function and how different hyperparameters, such as activation functions and the level of data augmentation, impact the performance of learned descriptors.

Another issue we encountered during developing a local image descriptor using a (variational) autoencoder is not being able to tell how well it will perform as a descriptor once it is trained. The autoencoder is trained by minimising the loss function which measures some difference (e.g. mean squared error or

binary cross-entropy) between the input and the output of the network. However, we have noticed that having the lowest such difference does not necessarily lead to the best performing descriptors. It is only after evaluating the descriptor on a benchmark (such as HPatches [Balntas 17]), which takes a very long time (sometimes as long as the training itself), that we know how good the descriptor is.

To overcome these limitations, in this chapter, we provide deeper insights into the descriptor evaluation process and explore the selection of metrics (that measure the difference between the input and the output of a (V)AE) which can serve as "proxies" for how well the descriptor will perform.

In addition to these studies on local image descriptors with autoencoders, in this chapter we also consider the invertibility of descriptors. Over the past decade, there has been interest in inverting local image descriptors. The seminal paper on reconstructing an image from its SIFT descriptors has started this trend by Weinzaepfel et al. [Weinzaepfel 11], followed by works on inverting binary descriptors [d'Angelo 12], HOG [Vondrick 13], and, more recently, using deep learning for inverting descriptors [Mahendran 15].

In this chapter, we propose a self-supervised method that specialises in learning both a descriptor function that maps image patches to their encodings and an inverting function that decodes these encodings back into the original image patches. To the best of our knowledge, we are the first to present a descriptor that is optimised for inversion. Our method is using β-variational autoencoders, which we modify to achieve an optimal balance between preserving similarities between patches and achieving good invertibility. We perform a thorough analysis of how the β value influences this trade-off.

To summarise, the main contributions presented in this chapter are as follows:

1. We present a thorough comparison between autoencoder-based and variational-autoencoder–based approach for learning local image descriptors and analysis of hyperparameter importance for obtaining a successful descriptor. To our knowledge, no comparison between AEs and VAEs in this context has been carried out. We are also not aware of any other works using VAEs for learning local image descriptors.

2. We perform a thorough hyperparameter analysis to establish which hyperparameters are the most important when learning a descriptor using a (variational) autoencoder. In addition to that, we propose using perceptual loss for training the autoencoders, which proved to significantly increase the performance of the descriptor.

3. We provide important insights into evaluation metrics for the learned local image descriptors and propose a rapid approximate evaluation method for descriptors learned with autoencoders that shows high correlation with the established benchmarks such as HPatches.

4. Following the works of many supervised invertible local image descriptors, we propose a self-supervised invertible descriptor, which enables trans-

forming patch encodings back into image patches. We present a study on the trade-offs between descriptor's invertibility and its performance.

The rest of this chapter is organised as follows. In the following section, we discuss the benchmarks for local image descriptors. In Section 3.3, we compare the autoencoder-based and variational-autoencoder–based approach to learning descriptors, and carry out a hyperparameter study for descriptors learned in this way. In Section 3.4, we study the evaluation of autoencoders for learning local image descriptors and propose an approach for fast approximate evaluation of the trained models. In Section 3.5, we describe our proposed invertible local image descriptor and we perform an empirical study to evaluate it. We conclude the chapter in Section 3.6.

3.2 Benchmarks for local image descriptor evaluation

Evaluation of local image descriptors plays a crucial role in their design. The learning-based descriptors rely on the evaluation at different stages to make the decisions, both during the training of a neural network, and the optimisation of the hyperparameters. For a long time (up until 2017), the most widely-adopted benchmark for evaluating local image descriptors was the Oxford matching dataset [Mikolajczyk 05] (from 2005), consisting of only 48 images. Other early datasets include DTU Robots dataset [Aanæs 12], Hanover dataset [Cordes 13], Generated Matching dataset [Fischer 14a], WxBs dataset [Mishkin 15]. Most of them also contain a small amount of images, or do not support evaluation of different tasks that local image descriptors perform.

In 2017, Balntas et al. published widely adopted, comprehensive **HPatches benchmark** [Balntas 17]. It enables evaluation of local image descriptors' performance on three different tasks:

- The patch retrieval task tests how well a descriptor can match a query patch to a pool of patches extracted from many images, including many distractors.
- The image matching task tests to what extent a descriptor can correctly identify correspondences in two images based on a pair of patches – one patch from each of the images.
- The patch verification task measures the ability of a descriptor to classify whether two patches match, i.e., whether they are extracted from the same measurement, as defined in the benchmark.

Each task can be of varying difficulty level ('easy', 'hard' and 'tough' – referring to the amount of geometric noise, as defined in [Balntas 17]).

While this allows a comprehensive evaluation of a descriptor, the evaluation itself is very computationally expensive. It is, therefore, not viable to use this benchmark to quickly evaluate the learned descriptor after every training (or, better yet, every epoch in the training), in order to be able to get insights into

which aspects of the network work best, and thus be able to make informed decisions about how to select hyperparameters, architecture of the neural network, etc. For supervised learning, such an evaluation can be performed using the labels provided in the dataset, but in self-supervised learning, we have no labelled data and thus we need some effective way to predict descriptor's performance. In the case of autoencoders, one can evaluate the similarity between the input and the output image patch (that is learned to be as close as possible to the input one). It is expected that evaluating this similarity by the mean squared error (MSE) cannot predict well the performance on standard patch retrieval tasks as MSE can be heavily influenced even by a small pixel shift. Thus using this metric during training is likely to result in very different encodings of structurally similar, but somewhat shifted or otherwise slightly deformed patches. Therefore, in Section 3.4 we explore incorporating other image similarity metrics and analyse how well they can predict the performance of local image descriptors on standard tasks.

3.3 AE versus VAE and a hyperparameter study

Here we present an in-depth study on learning local image descriptors using (variational) autoencoders. First, we describe the selected hyperparameters (Section 3.3.1), then the experimental setup (Section 3.3.2) and, finally, the empirical evaluation (Section 3.3.3).

Why the focus on AEs versus VAEs? Classical autoencoders serve as a baseline for compacting the essential information from the input into a lower-dimensional representation. However, their lack of ability to ensure the continuity of the latent space may be a drawback when learning local image descriptors. Variational autoencoders are designed to ensure the smoothness of latent space, and are therefore a viable alternative to AEs in this task. We shall not consider sparse autoencoders as they are overcomplete, meaning that the encoding is larger in dimensionality than the input. Encoding patches into higher-dimensional, albeit sparse, vectors is not a desired property of a descriptor and would require these encodings to be further compressed. Denoising autoencoders add artificial noise to the input and are trained to denoise it, thus also learning to encode useful properties of data samples. We emulate this behaviour using data augmentation, which is taken as a separate hyperparameter, as we discuss later in this section. Contractive autoencoders have been shown to be related to the denoising ones [Alain 14], therefore, we do not consider them separately.

3.3.1 Overview of the hyperparameters

In choosing which hyperparameters to optimise and how, we need to consider two aspects: (i) how well the autoencoder learned its primary task – to encode the input into a low-dimensional representation and to recover a close replica of the input from that encoding, and (ii) how well the generated encoding performs as local image descriptor.

When considering the first objective, the main goal of hyperparameter search is to adjust the effective capacity of the neural network to match the complexity of its task at hand [Goodfellow 16a]. The way hyperparameters can influence this is by influencing the actual capacity of the network, the ability to successfully minimise the cost function, or the degree of regularisation [Goodfellow 16a]. The literature on how different hyperparameters influence these different aspects is plentiful.

The second objective is more elusive and less explored. While it is straightforward how to optimise and evaluate autoencoders' performance on reconstructing the input (which is what they are trained to do), it is less straightforward how to optimise the objective "learn the best possible descriptor". For example, data augmentation may lead to worse performance on the first objective (learning to perfectly reconstruct) by AE learning to output blurred patches, but may in fact lead to the better performance on the second objective (learning to preserve similarity among patches in the encodings, i.e. learning a good descriptor). For this objective, there exists no literature on which hyperparameters influence it and in which way. This is what we explore in this section.

Table 3.1 lists the hyperparameters that we address, with the concrete choices that we include in our analysis and the main empirical findings. Here we explain briefly the analysed hyperparameters and the empirical findings will be detailed in the next section.

An important parameter that we consider is the loss function used to calculate the differences between the input images and the output (reconstructed) images. The choice of loss function has been shown to strongly influence the performance of neural networks, be it an autoencoder [Pihlgren 20], or other types of neural networks [Janocha 17]. The default choice for the loss function of AEs (and the default for the reconstruction term in VAEs) is the binary cross-entropy, $\mathrm{BCE}(x, y) = -\sum_{i=1}^{N}\sum_{j=1}^{N} y_{ij}\log x_{ij} + (1-y_{ij})\log(1-x_{ij})$, where x and y are two images of size $N \times N$. Recently, a perceptual loss, multiscale structural similarity (MS-SSIM), has been shown to give significant improvements when training autoencoders [Pihlgren 20], but has not been used before for learning descriptors. MS-SSIM is a multiscale version of the SSIM, which is defined as

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{3.1}$$

where x and y are two windows, μ_x and μ_y are the average and σ_x^2 and σ_y^2 are the variance of x and y, respectively, σ_{xy} is the covariance between x and y, and c_1 and c_2 are the variables to stabilize the division with weak denominator. For more details on MS-SSIM, refer to [Wang 03].

The choice of activation function, in our experience, has a significant influence on the training and overall performance of the autoencoder. The Exponential Linear Unit (ELU) and Rectified Linear Unit (ReLU) have shown to be the best-performing ones, which is why we include these two in the analysis. Exponential Linear Unit is defined as follows:

$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha(\mathrm{e}^x - 1)) & \text{if } x < 0 \end{cases} \tag{3.2}$$

and Rectified Linear Unit as $\mathrm{ReLU}(x) = \max(0, x)$. We do not change the activation of the last layer, where we use sigmoid function as it is common when the output is in image format [Goodfellow 16a].

Data augmentation acts as a form of regularisation in the context of learning descriptors. If we add some geometrical noise to the input, e.g. rotation by 10°, and expect the output to be the original image, then the autoencoder will learn to ignore these minor variations in the rotation. It is similar for other types of geometric noise (translation, scaling, shearing) which we want our descriptor to be (to an extent) invariant to. We empirically explore this hypothesis in Section 3.3.3.

We also consider the choice of the type of autoencoder as a hyperparameter. Here we briefly discuss the motivation behind choosing the types of autoencoders for this study. Classical autoencoders that use dimensionality constraint on the encoding layer are simplest implementation of the concept of encoding useful information by learning to copy the input to the output with some constraint (i.e. the concept of autoencoders), and therefore the obvious first choice for learning a local image descriptor in a self-supervised way. However, their lack of ability to ensure the continuity of the latent space may be a drawback when learning local image descriptors, as was discussed in Section 2.4.1 (smooth latent space of an autoencoder should lead to increasing the similarity-preserving property of a descriptor learned from that autoencoder). Variational autoencoders are designed to ensure the smoothness of latent space, and therefore we study them as well. More specifically, we consider their generalised form, β-VAEs, where we can tune the smoothness of the latent space. Concerning sparse autoencoders, they are overcomplete, meaning that the encoding is larger in dimensionality than the input. Encoding patches into higher-dimensional, albeit sparse, vectors is not a useful property of a descriptor and would require these encodings to be further compressed. Therefore, sparse autoencoders are not considered in this study. Denoising autoencoders add artificial noise to the input and are trained to denoise it, thus also learning to encode useful properties of data samples. We emulate this using data augmentation, which is taken as a separate hyperparameter, as we discuss later in this section. Contractive autoencoders have been shown to be related to the denoising ones [Alain 14], therefore, we do not consider them separately.

Beta parameter is a parameter specific to VAEs (i.e. to their generalisation, β-VAEs) and is used in the loss function to regulate the trade-off between the reconstruction term and the term that enforces the latent space distribution to be as close as possible to the prior distribution. It is a very important parameter, as it influences the trade-off between VAE learning to reconstruct well and its latent space having a distribution close to the prior distribution. The correct choice of β has been shown to induce disentanglement of facial features on the datasets of images of faces [Higgins 17], while also resulting in blurred reconstructions of those images. We hypothesise that better disentanglement is related to the descriptor producing more informative encodings, and in this work we perform an analysis of the influences of β parameter on the descriptor's performance.

Hyperparam.	Values	Empirical findings
Loss function	BCE, MS-SSIM	MS-SSIM consistently outperforms BCE on all tasks and regardless of other hyperparameters. Loss function seems to be the most important parameter.
Activation functions	ReLU, ELU	ReLU is used by the best performing descriptors. When using MS-SSIM loss function, ReLU outperforms ELU, when using BCE loss, it depends on the data augmentation level – ReLU is better with less data augmentation and ELU is better with more data augmentation.
Data augmentation level	$\{0, 1, 2, 3\}$	Best performing models overall use no data augmentation, but when breaking down into HPatches tasks, this is not always the case. For the matching task data augmentation levels 1 and 2 are the best, for retrieval 1, and for verification 0. Using data augmentation higher than 2 seems to degrade the performance of the descriptor across all tasks.
The extent to which the latent space distribution is enforced to be as close as possible to the prior distribution	Variational autoencoder with $\beta \in$ {1e-05, 1e-04, 1e-03}, classical autoencoder	For all three HPatches tasks and overall, variational autoencoders with β values on the lower end (1e-05 or 1e-04) and classical autoencoders show similar (best) results. Classical autoencoders are simpler, deterministic, and more trivial to train (do not require tuning β parameter), however, for performance-sensitive applications VAEs are a marginally better option. When BCE loss is used, a VAE with a high β value will perform poorly.

Table 3.1: Summary of the analysed hyperparameters and our findings regarding their influence on the quality of local image descriptors learned with (variational) autoencoders.

We do not study the optimisation of the learning rate – the studies on it are plentiful. As it controls the first objective – effective capacity of the network, and there are many algorithms that automatically optimise it (such as RMSProp, Adam, Adadelta), its tuning is beyond the scope of this work. We adopt one of the standard optimisation algorithms (Adam) with default settings. Same holds for the weight decay coefficient.

Similarly, hyperparameters such as the number of hidden units in the layers and the number of layers in the network, have been researched elsewhere since they optimise the effective capacity of the network. Ultimately, they depend on the size of input patches. Standard regularisations such as dropout have also been explored in a general sense, they also optimise the first objective and are not specific to our problem, therefore, we do not explore them either.

We use grid search over the selected parameters – the reason for this approach (as opposed to a more heuristic-based search) is that we can get more insights into how they interact with each other when we have the results for the grid than if we had it randomly distributed.

3.3.2 The experimental setup

Our neural network models are built using PyTorch library for deep learning. We perform grid search for our hyperparameters using Weights & Biases Python library.

We use standard autoencoder architecture in these experiments. The encoder consists of three convolutional layers with zero padding and kernel size 3×3, each followed by a max-pooling layer. The last layer of the encoder is a fully-connected layer. The decoder mirrors the encoder, with a fully-connected layer followed by three transposed convolutional layers with stride 2 and kernel size 2×2.

The architecture of the variational autoencoder is the same as that of the classical autoencoder, except that one fully connected layer at the end of the encoder is replaced by two parallel ones (having the same input) – for the mean and variance of the Gaussian distribution. Using the mean and variance, we can sample from the Gaussian distribution in order to obtain the bottleneck layer – the encoding of our descriptor. The number of Gaussians corresponds to the size of latent space – in our case 32.

All the convolutional layers in the models have 32 filter maps as input and output 32 filter maps, except for the first and the last one – the first one has 1 filter map as input (because the input is a one-channel (grayscale) image), and the last one outputs 1 filter map in order to match the input of the model. The autoencoders are trained on grayscale patches so that they can be assessed with HPatches benchmark, but they can be easily extended to RGB patches by changing the number of input and output layers to 3.

After the training of the (V)AE has been completed, the decoder part of the network is discarded, and we are left with the encoder network, which is our local image descriptor – for a given image patch, it outputs its encoding.

For a data augmentation level a (where $a \in \{0, 1, 2, 3\}$), we perform the following transforms on the input patch to the (variational) autoencoder:

- rotation by a_r degrees, where $a_r \sim \mathcal{U}(-10a, 10a)$
- translation by a_t% of the input patch size, where $a_t \sim \mathcal{U}(-10a, 10a)$
- scaling of the input patch to a_{sc}% of the original, where $a_{sc} \sim \mathcal{U}(100 - 10a, 100 + 10a)$
- shearing transform where the new y axis forms an angle of a_{sc} to the original $yaxis$, where $a_{sc} \sim \mathcal{U}(-10a, 10a)$

We do not consider these types of data augmentation separately, as it would significantly increase the hyperparameter search space, and, in general, data augmentation is considered to have less impact on the training of neural networks than the other hyperparameters we have examined [Goodfellow 16b].

We use Adam optimiser for all neural networks in this chapter. The networks are trained on a dataset of 125k 65×65 patches that were extracted from the images from ImageNet [Deng 09], KonIQ [Hosu 20] and Visual Genome [Krishna 17] datasets. The patches were extracted using FAST (Features from Accelerated Segment Test) algorithm for feature detection [Rosten 06]. The patch size has been chosen because HPatches benchmark expects this patch size. The size of the encodings is 32. The ratio between training, validation and test set is $8 : 1 : 1$.

For the hyperparameter search we have created a full pipeline that allows training the models and their subsequent evaluation on HPatches (as well as using other metrics between the input and the output of the test set images). It was, therefore, enough to start one script in order to execute the whole hyperparameter grid search and evaluation. The code for the experiments in this chapter is open-source and can be found in our GitHub repository.

3.3.3 Empirical results for hyperparameter selection

We now evaluate the performance of descriptors learned with (variational) autoencoders, analysing jointly the influence of the selected hyperparameters. For each set of hyperparameters, we evaluated the learned descriptor on HPatches benchmark [Balntas 17] for all three tasks: matching, retrieval and verification. Since the performance on each of these tasks can yield different conclusions regarding the optimal values of the hyperparameters, we find it useful to have one uniting evaluation measure, but HPatches does not provide that. Thus we create an additional 'overall' performance metric by normalising the outputs from the three provided tasks between 0 and 1, and averaging the three normalised values.

The results can be found in Figure 3.1 (overall performance) and Figure 3.2 (performance on the three individual tasks). These results allow us to draw important conclusions about the choice of hyperparameters and the choice between AEs and VAEs. Table 3.1 presents the summary of the findings, which we discuss in more detail in the following paragraphs.

MS-SSIM loss yields better results than BCE on all tasks. This can be explained by the fact that BCE is a pixel-wise loss function which implies (1) that it does not consider the relations between different pixels in the patch,

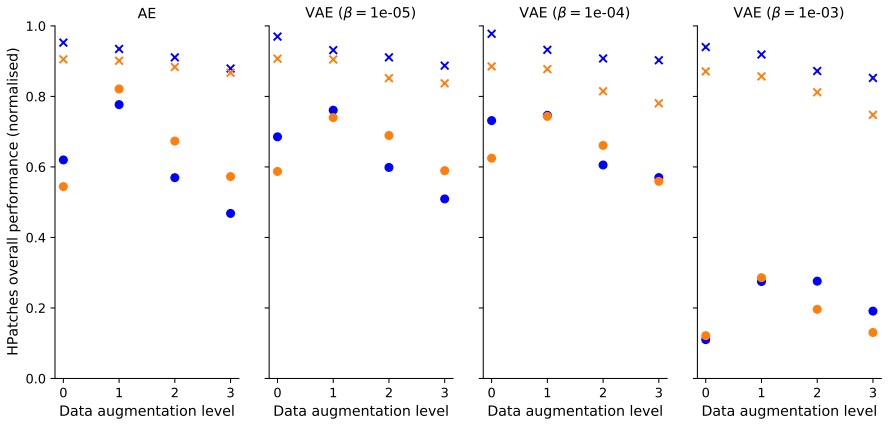


Figure 3.1: HPatches performance (normalised), showing the effect of different hyperparameter choices. β value differs across the graphs, increasing from left to right, in the left-most graph showing classical AE (essentially a deterministic VAE with $\beta = 0$), and the other three graphs VAEs with β values 1e-05, 1e-04, 1e-03, respectively. Data augmentation level is shown on x-axis of the graphs. The choice of activation function is indicated in different colours (ReLU, ELU) and the loss function with different markers ($\times$ MS-SSIM, $\bullet$ BCE).

leading to an encoding that is unable to capture spatial structures and (2) that it weighs all pixels equally, even though some groups of pixels may be more discriminative. MS-SSIM loss is a perception-based loss that alleviates these issues by considering the differences in structural information of an image patch (i.e. the inter-dependencies between pixels), as well as the perceptual aspects: luminance and contrast. It is important to note that the increased performance of descriptors trained using MS-SSIM loss comes at the cost of increased training time of an autoencoder. We accept this trade-off since the incurred cost only impacts the training time – the inference time (i.e. time it takes to calculate the descriptor) is not affected by the loss function.

Further on, the MS-SSIM loss function together with the ReLU activation yields the best descriptors in terms of performance on all tasks and for both autoencoders and variational autoencoders, and, in case of VAEs, this holds for different values of β parameter. Interestingly, this performance is best with little to no data augmentation.

Comparison between VAEs and AEs leads to interesting insights. For all the three tasks as well as overall, the best performing descriptors are always trained using variational autoencoders. However, the descriptors trained with classical autoencoders are usually performing only slightly worse. Furthermore, we have also observed that the worst performing descriptors also come from variational autoencoders (when β value is too high). Therefore, rather modest improvements that VAEs offer over AEs in these tasks come at a price of additional training (tuning the β parameter), requiring additional computational resources.

Regarding the choice of the loss function, we observe that working with BCE favours small amount of data augmentation, while MS-SSIM sometimes

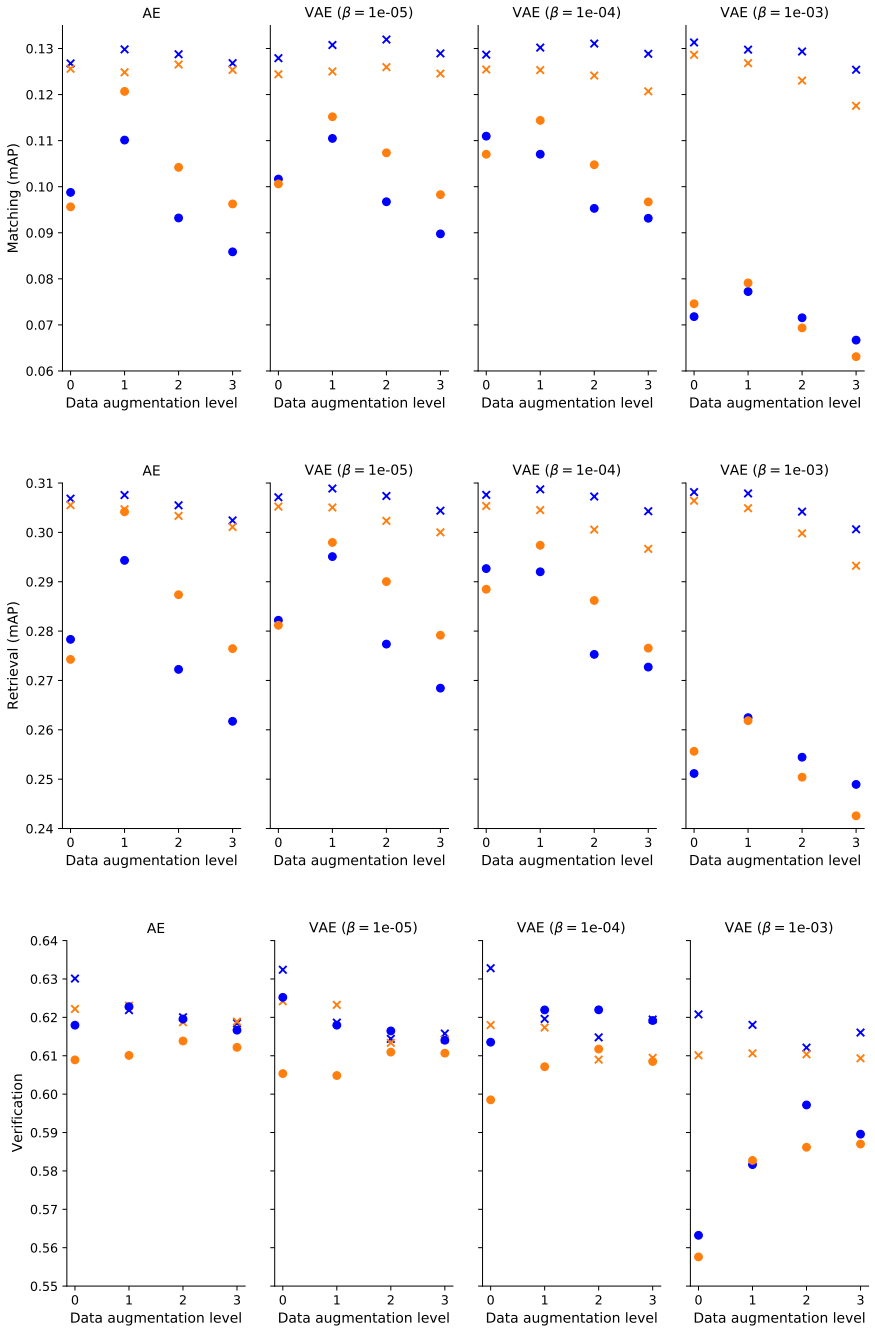


Figure 3.2: Performance by HPatches tasks: matching (top), retrieval (middle) and verification (bottom), showing the effect of different hyperparameter choices. For each task, β value differs across the graphs, increasing from left to right, in the left-most graphs showing classical AEs (essentially a deterministic VAEs with $\beta = 0$), and the other three graphs VAEs with β values 1e-05, 1e-04, 1e-03, respectively. Data augmentation level is shown on x-axis of the graphs. The choice of activation function is indicated in different colours (ReLU, ELU) and the loss function with different markers ($\times$ MS-SSIM, $\bullet$ BCE).

works best with no data augmentation at all. We hypothesise that BCE needs more data augmentation in order to "nudge" the network into preserving similarities between patches – otherwise it may learn to reconstruct very well, but also be very sensitive to small perturbations of the patches, which is undesirable. Surprisingly, moderate to high levels of data augmentation are negatively impacting the performance, regardless of other parameters and for all the tasks.

Similarly, a too high β value (when using VAEs) is detrimental to the performance of the learned descriptors. High β values mean more weight on the KLD term of the loss function – which is essentially a form of regularisation. However, same as with the data augmentation regularisation, having some KLD regularisation is better than having none.

It is also interesting to see that when using MS-SSIM, ReLU activation function works better than ELU for all the tasks, but when using BCE, there is not a single best activation function – ReLU works better than ELU for verification task, but ELU outperforms ReLU for the matching and retrieval tasks.

Looking at different tasks, we found that hyperparameters influence each other in the same way for matching and retrieval tasks, i.e. descriptors' performance on them is almost perfectly correlated (0.97). Verification, too, is highly correlated with the other two tasks (with matching 0.91 and with retrieval 0.94), but the best performing hyperparameter combinations are not always the same between verification and either of the two tasks. Most notably, with matching and retrieval, MS-SSIM always outperforms BCE. However, with verification task, this is not always the case (nonetheless, the models that are performing the best on this task do use MS-SSIM, and the ones performing the worst use BCE).

We also notice that for MS-SSIM, ReLU performs consistently better than ELU, but for BCE it is not clear which activation function is better.

3.4 Approximate evaluation of autoencoders for local image descriptors

As we discussed in Chapter 2, the key feature of autoencoders is that they do not need labelled data for training. Instead, they are trained to recreate at the output a faithful "copy" of the input from a latent representation in their middle layer. The idea is to learn to generalise by extracting the essential features of the input for a given task, from which endless examples of the same kind can be generated. What is assumed under generating examples "of the same kind" depends on the given task (e.g. in some imaging tasks, invariance to certain geometric transforms and/or noise is desired, and in others not). The question that poses itself then is – how to measure the "goodness of fit" of the obtained representation during training?

One approach is to use a benchmark designed for the task that the autoencoders are learning. For example, in the case of learning local image descriptors using autoencoders, one could use HPatches benchmark for evaluating local image descriptors [Balntas 17]. There are two issues with this approach: (1) The

benchmark evaluation itself might be too computationally intensive to perform it during the training (e.g. after every n epochs, or after every finished run in a hyperparameter sweep). For example, it takes around 40 minutes to evaluate a local image descriptor on the aforementioned HPatches benchmark (on an AMD Ryzen Threadripper 1920X 12-core processor with 32GB of RAM). (2) Evaluating autoencoder's performance on a benchmark requires a benchmark in the first place. This is problematic in the cases where one wants to use the autoencoder for a task for which a benchmark does not exist. Furthermore, the requirement of a benchmark partially defeats the purpose of AEs not requiring labelled data – since benchmark requires manual work and data labelling.

Another approach to measure the success of the learned representation is to evaluate the difference between the inputs and the learned outputs of the trained autoencoder. In practice, mean squared error (MSE) difference is usually used for this, especially during the training and/or hyperparameter sweeps. However, there are multiple issues with MSE (as pointed out by Pihlgren et al. [Pihlgren 20]): (1) MSE does not take into account relations between different pixels – it only matters that each output pixel is as close as possible to the corresponding input pixel. (2) All pixels are weighted equally even though some groups of pixels might be more important, e.g. the ones depicting the salient features in the image.

In some cases, other measures of difference are used for evaluating the usefulness of AEs, but they are typically also pixel-wise metrics. However, these measures of difference are often taken without a real explanation for why *they* have been chosen. To the best of our knowledge, there exist no studies that compare various measures of difference to see how they relate to the usefulness of the trained AEs.

Here we tackle this issue for autoencoders learned for local image descriptors. We investigate various measures of difference between the input and the output of the trained AE and compare their correlation with the HPatches benchmark for local image descriptors. In this way, we establish what measure of difference is the best "proxy" for how well the autoencoder is trained to perform as a local image descriptor. The idea is that one could then use this measure of difference during the AE training instead of evaluating the AE on the benchmark during the training, which is a lot more computationally intensive and sometimes even infeasible.

In the following section we introduce different image similarity metrics that we consider for the approximate evaluation of autoencoders. In Section 3.4.2 we research the usefulness of these image similarity metrics for evaluating autoencoders for learning local image descriptors: we describe our experimental setup in Section 3.4.2.1 and discuss the results in Section 3.4.2.2.

3.4.1 Image similarity metrics

Research into image similarity metrics has been a popular field of study with many different methods proposed in the past decades. Here we list some of the most important ones which we also examine as methods for approximate evaluation of autoencoders.

Mean squared error (MSE) is a common image quality measurement metric, calculated as $\mathrm{MSE}(\mathbf{x},\mathbf{y}) = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}(x_{mn}-y_{mn})^2$, for images $\mathbf{x}$ and $\mathbf{y}$ of dimensions $M \times N$. Peak signal-to-noise ratio (PSNR) is a metric derived from MSE, so we do not consider it separately. We do include a variant PSNR with blocking effect factor PSNR-B [Yim 10], which is also a common visual quality metric in some tasks.

We incorporate three other MSE-based metrics. Sliding window root mean squared error ($\mathrm{RMSE_{SW}}$) is defined as: $\mathrm{RMSE_{SW}}(\mathbf{x},\mathbf{y}) = \frac{1}{W}\sum_{w=1}^{W}\sqrt{\mathrm{MSE}(\mathbf{x}_w,\mathbf{y}_w)}$, where $\mathbf{x}_w$ and $\mathbf{y}_w$ are corresponding windows of $\mathbf{x}$ and $\mathbf{y}$, and W is the number of these windows. ERGAS (Erreur Relative Globale Adimensionnelle de Synthèse, from its French acronym) is also based on RMSE, for its definition, refer to [Wald 00]. Finally, normalised root mean squared error (NRMSE): $\mathrm{NRMSE}_f(\mathbf{x},\mathbf{y}) = \frac{\sqrt{\mathrm{MSE}(\mathbf{x},\mathbf{y})}}{f}$, where f is a normalisation factor. There is no standard method of normalisation across the literature, so we consider in this paper three different normalisation methods: (1) normalisation by averaged Euclidean norm of the first image $\mathbf{x}$ (the input image to the autoencoder), (2) normalisation by the intensity range of $\mathbf{x}$ and (3) normalisation by the mean of $\mathbf{x}$.

The next metric that we consider is adapted Rand error (ARE) [Arganda-Carreras 15], calculated as $\mathrm{ARE}(\mathbf{x},\mathbf{y}) = 1 - \frac{2pr}{p+r}$, where p and r are adapted Rand precision and adapted Rand recall (respectively), as defined in [Arganda-Carreras 15]. ARE has originally been developed for evaluation of (3D) image segmentation algorithms, but has since been used more broadly.

Variation of information (VoI) [Meilă 07] measures the distance between images in terms of the information loss and gain between them. It is defined as $\mathrm{VoI}(\mathbf{x},\mathbf{y}) = H(\mathbf{x}|\mathbf{y}) + H(\mathbf{y}|\mathbf{x})$, where $H(\cdot|\cdot)$ is conditional entropy. We also consider both conditional entropies separately for this study.

Visual information fidelity (VIF) is another information-theory–based metric proposed by Sheikh and Bovik [Sheikh 06] which quantifies the information shared between two images. For the formal definition of VIF, refer to [Sheikh 06].

Universal image quality index (UQI) is a metric proposed by Wang and Bovik [Wang 02] which models difference between images as a combination of three factors: loss of correlation, contrast distortion, and luminance distortion. It is calculated as follows:

$$\mathrm{UQI}(\mathbf{x},\mathbf{y}) = \frac{4\sigma_{\mathbf{x},\mathbf{y}}\mu_{\mathbf{x}}\mu_{\mathbf{y}}}{(\sigma_{\mathbf{x}}^2+\sigma_{\mathbf{y}}^2)(\mu_{\mathbf{x}}^2+\mu_{\mathbf{y}}^2)}, \tag{3.3}$$

where $\mu_{\mathbf{x}}$ and $\mu_{\mathbf{y}}$ are the average and $\sigma_{\mathbf{x}}^2$ and $\sigma_{\mathbf{y}}^2$ are the variance of $\mathbf{x}$ and $\mathbf{y}$, respectively, $\sigma_{\mathbf{xy}}$ is the covariance between $\mathbf{x}$ and $\mathbf{y}$.

Next, we discuss SSIM [Wang 04]. It is a perception-based metric that considers the differences in structural information of an image patch (i.e. the inter-dependencies between pixels), as well as the perceptual aspects: luminance and contrast. SSIM is defined as:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}, \tag{3.4}$$

where μ_x, μ_y, σ_x^2, σ_y^2 and σ_{xy} are defined as with UQI, and c_1 and c_2 are the variables to stabilise the division with weak denominator.

Spatial Correlation Coefficient (SCC) is calculated by applying Laplacian kernel to the images and then computing the correlation coefficient between the resulting images, as proposed by Zhou et al. [Zhou 98].

3.4.2 Proposed approximate evaluation method

We now investigate and describe our proposed method for approximate evaluation of the performance of a descriptor-learning autoencoder, based on how the autoencoder reconstructs patches. We first describe our experimental setup (Section 3.4.2.1) and then we present the results and discussion (Section 3.4.2.2).

3.4.2.1 The experimental setup

The primary goal of this section is to establish whether some relatively simple image similarity metric can serve as a low-complexity proxy for the computationally expensive evaluation on HPatches during the training of autoencoders. We perform this evaluation based on the difference between the input and the output patches of the autoencoder, as measured by the different image similarity metrics. Our objective is to find which image similarity metric shows the most similar results to the evaluation on HPatches benchmark, i.e. we want to find the metric that is the most correlated with HPatches evaluation.

To this end, we train the autoencoder with different hyperparameters, and for each fully-trained AE we note its performance on HPatches benchmark and we also note the average distance between the input and the output of the AE, measured by different image similarity metrics. We then compute the Pearson correlation between the score on HPatches and each of the metrics. We look at HPatches overall score (Figure 3.3), and the three different HPatches tasks (image matching, patch retrieval and patch verification – Figure 3.4) to establish which metric is the best for approximate evaluation of each of these. The values shown in these figures are absolute values of the correlations that are in the range [-1, 1] (where -1 means perfect negative correlation, 0 means no correlation, and 1 means perfect positive correlation).

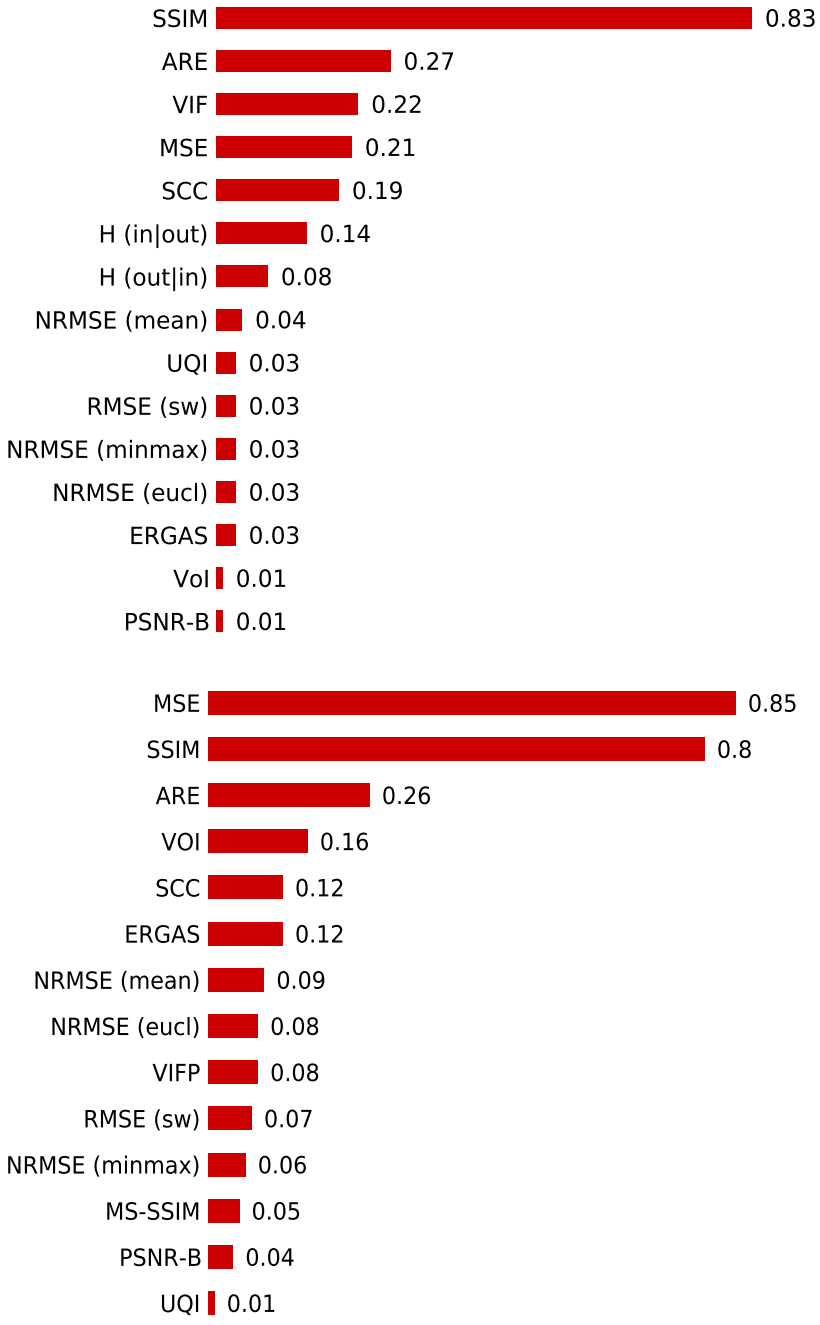


Figure 3.3: Absolute value of correlation between performance of a descriptor learned using an autoencoder on HPatches benchmark, and the distance in terms of different metrics between the input and output patches of that autoencoder (top: the hyperparameter sweep with both BCE and MS-SSIM loss; bottom: the hyperparameter sweep with only BCE loss). The range of the correlation values before taking the absolute value is $[-1, 1]$.

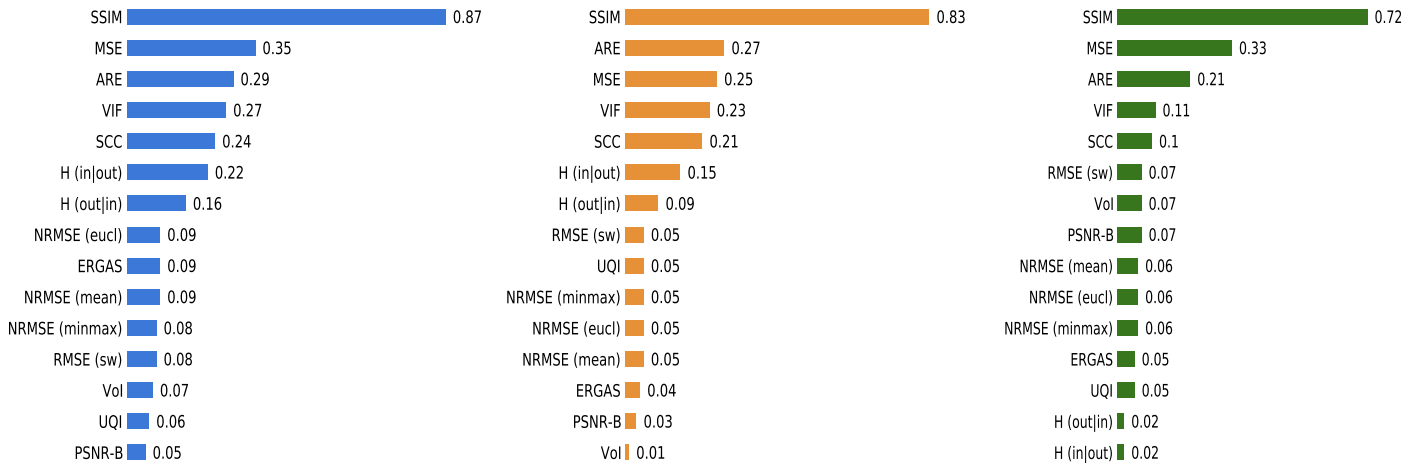


Figure 3.4: Absolute value of correlation between performance of a descriptor learned using an autoencoder on HPatches benchmark (for 3 different tasks: image matching, patch retrieval and patch verification), and the distance in terms of different metrics between the input and output patches of that autoencoder. The correlation has been measured on the hyperparameter sweep that includes both BCE and MS-SSIM loss. The range of the correlation values before taking the absolute value is $[-1, 1]$.

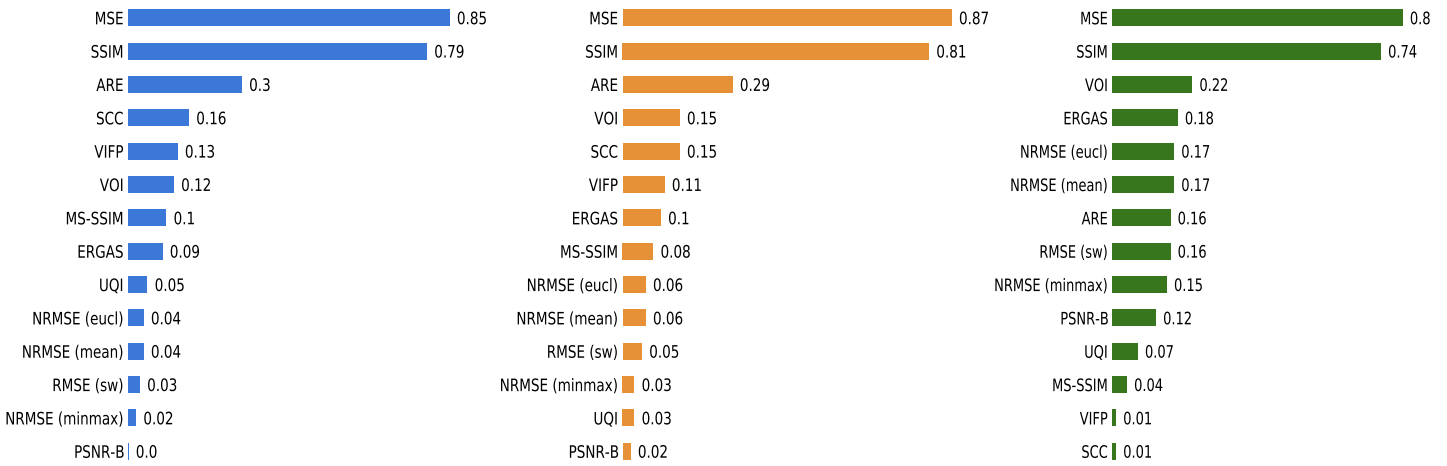


Figure 3.5: Absolute value of correlation between performance of a descriptor learned using an autoencoder on HPatches benchmark (for 3 different tasks: image matching, patch retrieval and patch verification), and the distance in terms of different metrics between the input and output patches of that autoencoder. The correlation has been measured on the hyperparameter sweep that includes only the BCE loss. The range of the correlation values before taking the absolute value is $[-1, 1]$.

The patches in this experiment are taken from a test set (containing 12.5k patches), which the autoencoder has not seen during training. The autoencoders were trained with different combinations of the following hyperparameters (hyperparameter sweep): learning rate (values: 1e-03, 1e-04, 1e-05), loss function (BCE and perceptual loss [Pihlgren 20]), activation function after all the layers except for the last one (ReLU and ELU), data augmentation level, and β parameter of the β-variational autoencoder (values: 1e-03, 1e-04, 1e-05, and 0 (classical AE)). We compute the correlations between the HPatches metrics and the approximate evaluation metrics on two different hyperparameter sweeps – one where the hyperparameter search space includes both BCE and perceptual (MS-SSIM) loss, and the other where we use only the BCE loss. In the former setup, the aim is to find a more general approximate evaluation method that works best on different loss functions. In the latter setup, the idea is find the approximate evaluation metric when using only the BCE loss, since this is the most common loss function used for training (V)AEs.

We use Adam optimiser for all trainings. The networks are trained on a dataset of 125k 65×65 patches that were extracted from the images from ImageNet [Deng 09], KonIQ [Hosu 20] and Visual Genome [Krishna 17] datasets. The patches were extracted using FAST (Features from Accelerated Segment Test) algorithm for feature detection [Rosten 06]. The patch size has been chosen because HPatches benchmark expects this patch size. The size of the encodings is 32. The ratio between training, validation and test set is $8 : 1 : 1$.

The benchmark that we use for the accurate evaluation of the performance of autoencoders is HPatches benchmark for local image descriptors [Balntas 17]. It enables evaluation of descriptors' performance on three different tasks (patch retrieval, image matching, and patch verification), each with varying difficulty levels ('easy', 'hard' and 'tough' – referring to the amount of geometric noise. For detailed definitions of this benchmark and these tasks, we refer the reader to the HPatches paper [Balntas 17].

The autoencoder evaluation using each of the metrics did not exceed one minute (on an AMD Ryzen Threadripper 1920X 12-core processor with 32GB of RAM), as opposed to the evaluation on HPatches benchmark which takes around 40 minutes on the same setup.

3.4.2.2 Results and discussion

Figure 3.3 shows the correlations between different image similarity metrics and the overall HPatches score, both for the hyperparameter sweep with BCE and MS-SSIM loss as well as for the hyperparameter sweep with only BCE loss. For the former hyperparameter sweep, it is quite notable that by far the best metric for approximate evaluation of autoencoders learning local image descriptors is structural similarity index (SSIM), showing 0.83 correlation with HPatches overall score. All other tested metrics show very low level of correlation: for ARE, VIF and MSE, these are 0.27, 0.22 and 0.21, respectively, and for others even smaller. Hence, we do not recommend using them for approximate evaluation of autoencoders. For the latter hyperparameter sweep, we notice that the SSIM metric no longer shows the highest correlation – MSE now shows the

highest correlation (0.85), while SSIM shows slightly lower (0.8). Both metrics show significantly higher correlation than other metrics (the next one being only 0.26).

Figure 3.4 shows the correlations between the metrics and each of the three HPatches tasks (image matching, patch retrieval and patch verification) calculated on the hyperparameter sweep with BCE and MS-SSIM loss. In all three cases we observe SSIM metric outperforming the other metrics by a large margin. We note that for image matching and patch retrieval there is a moderate correlation for ARE, MSE, VIF and SCC (above 0.2). For the task of patch verification, the correlations with all the metrics are lower than for the other two tasks.

Figure 3.5 shows the correlations between the metrics and each of the three HPatches tasks (image matching, patch retrieval and patch verification) calculated on the hyperparameter sweep with only BCE loss. Similarly to the case with the overall HPatches score, SSIM metric shows the second highest correlation – only slightly bellow the MSE metric, and both of them being significantly higher than other metrics.

While the high performance of SSIM metric in the first hyperparameter setup can be (partially) attributed to the fact that in half of the cases we were optimising the MS-SSIM loss, it is still notable how high SSIM performs when the autoencoders are optimised using only the BCE loss. We believe that the reason for such high performance of SSIM is that it is a perceptual metric, meaning that it is designed to mimic how humans would compare images. This is achieved by considering the differences in structural information of images (and image patches), i.e. inter-dependencies between pixels, as well as the perceptual aspects: luminance and contrast. These aspects are very important in particular for local image descriptors, which are designed to be able to match different part of images that correspond to the same details in real life.

It is not surprising that some pixel-based metrics which do not account for structural features are not well correlated with the ability of autoencoders to learn a *useful* encoding as local image descriptor.

Image quality metrics VIF and UQI are not suitable for this task, which is as expected given that their focus is on evaluating visual quality rather than matching content. It would be of interest to consider some other measures that reflect the structure of patches, e.g. comparing the patches' encodings using traditional local image descriptors such as SIFT [Lowe 99], SURF [Bay 08] or BRIEF [Calonder 10].

We believe that these results are not only important because evaluating autoencoders (for learning descriptors) with SSIM or MSE (depending on the loss function of choice) instead of on the benchmarks could save a lot of computational resources, but also we believe there is high value in the insights that we provided, including that some other metrics are not good for approximate evaluation of AEs.

Figure 3.6: Reconstructing an image from its SIFT descriptors. Image taken from [Weinzaepfel 11].

3.5 Invertible local image descriptors

In this section, we propose a method that specialises in learning both a descriptor function that maps image patches to their encodings and an inverting function that decodes these encodings back into the original image patches. Inverting local image descriptors has been an active area of research in the past decade, starting with the prominent work by Weinzaepfel et al. [Weinzaepfel 11] on reconstructing an image from its SIFT descriptors (shown in Figure 3.6). The authors used a database of descriptors and their corresponding patches to search for the nearest neighbour to the query descriptor, and then take the patch connected to the retrieved nearest neighbour. Further works on inverting other descriptors followed, including the inversion of binary descriptors [d'Angelo 12] and of HOG [Vondrick 13]. A more recent paper by Mahendran et al. [Mahendran 15] considers inverting descriptors back into patches using deep learning.

So far, to the best of our knowledge, no one has proposed learning the descriptor and its inversion function simultaneously, therefore, we find this an interesting research area to explore. The benefit of invertible image descriptors can be seen in the ability to then use the descriptors as a crude compression method. Furthermore, they can be used as the last-resort (partial) recovery of images that have been lost due to e.g. database failure, but which have been indexed in databases (used for search and retrieval purposes) using the encodings computed using local image descriptors.

We propose using a β-variational autoencoder for simultaneous learning of local image descriptors and their reconstruction back into image patches. Due to the nature of their architecture, both classical and variational autoencoders are ideal for the simultaneous learning of the descriptor function (the encoder part of the autoencoder) and the reconstruction function (the decoder part). For a comprehensive overview of variational autoencoders, refer to Section 2.4.4. In contrast to classical autoencoders, VAEs include additional regularisation that allows modelling the latent space to be continuous and to be easy to interpolate across, ensuring that similar input data samples (patches)

get mapped to similar points in the latent space (encoding), and vice versa. This similarity-preserving property is a property of paramount importance for local image descriptors. We also hypothesise that the additional regularisation of VAEs will allow for learning sharper reconstructions in comparison to methods based on classic autoencoders, which we will show empirically in the next section.

In β-VAE, a generalisation of the loss function of VAEs is achieved by adding the β weight to the KL term. In this way, we can control the trade-off between learning to faithfully reconstruct the input patches and preserving patch similarities in the latent space. By setting the right value of β we can increase the influence of the reconstruction term to ensure good invertibility of the descriptor. In contrast to descriptors based on classical autoencoders, however, the KL term in the VAE loss function ensures the continuity of the latent space which could not be guaranteed when using the classical autoencoders.

Once we trained the β-VAE, the encoder part of it is our descriptor, and the decoder part is the inversion function that maps the patch encodings back to the original patches.

Figure 3.7 illustrates the architecture of the variational autoencoder used in this section. The encoder consists of three convolutional layers followed by the fully-connected layers for the means and variances of Gaussian distributions. From these layers, we sample a vector that is the encoding of the input patch. We set the dimensionality of the latent space M (and therefore, the mean, variance, and the sampling layers) to be 128. The decoder architecture mirrors that of the encoder – at the beginning, there is one layer fully-connected to the sample (encoding), followed by three transposed convolutional layers. The dimensions of the output patch of our VAE are the same as the dimensions of the input.

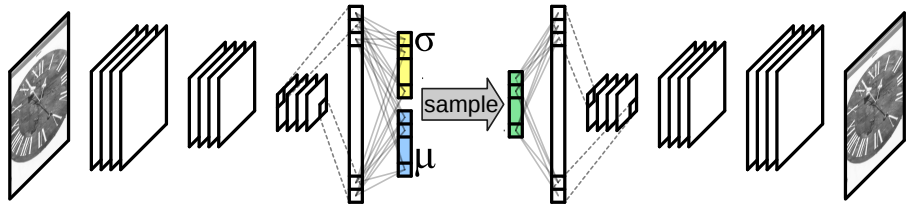


Figure 3.7: The selected variational autoencoder architecture that we used for learning the invertible local image descriptor.

Following the notation from [Higgins 17], we use β_{norm} as the main hyperparameter that we vary. β_{norm} is defined as follows:

$$\beta_{norm} = \frac{\beta M}{N}, \tag{3.5}$$

where M is the size of latent space and N is the input size. By normalising the β value, the analysis that we present in the following section can be applied to datasets of different patch size and different desired encoding sizes. We vary the β_{norm} values over several orders of magnitude – from 10^{-5} to 10^{2}. In the

following section, we show how the β_{norm} value influences the patch retrieval of the descriptor and its invertibility.

We use rectified linear unit (ReLU) activation functions after all layers, except the last layer, where we use the sigmoid activation function instead. We use Adam optimiser to learn the weights of the VAE, which is trained on a dataset of 80k 56×56 patches that were extracted from the images from the ImageNet dataset using FAST (Features from Accelerated Segment Test) algorithm for feature detection [Rosten 06]. The ratio between training, validation, and test set is $8 : 1 : 1$.

3.5.1 Experimental results

In this section, we show how the value of β_{norm} influences the proposed β-VAE–based local image descriptor and its performance with respect to patch retrieval (the main task for which local image descriptors are designed) and patch inversion from the patches' encodings.

We also evaluate both the retrieval and inversion capabilities of the proposed approach in comparison with a reference autoencoder-based descriptor. We compare our method only to this autoencoder-based descriptor, since non-autoencoder-based descriptors have no straightforward way of being inverted and thus give us no way of comparing their invertibility.

3.5.1.1 Evaluation on patch retrieval

Patch retrieval evaluation is performed as follows. We select a set of query patches within a test dataset of patches consisting of 12.5k patches. For each query patch, we retrieve the most similar patches by comparing their encodings as calculated by the descriptors. We show some examples of patches retrieved in such a way in Figure 3.8. The quality of patch retrieval is then evaluated based on two metrics (peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM)) between the query patches and patches deemed most similar to the queries based on the encodings computed with descriptors. The results are averaged over 12.5k patches in the test set.

We first examine the patch retrieval capabilities of the proposed β-VAE–based descriptor for different β_{norm} values. In Figure 3.9 (left), we see that, according to the PSNR metric, the patch retrieval seems to be the best when the β_{norm} value is the lowest, i.e., when the KL divergence term of the loss function is the closest to 0. However, when using a metric that better mimics human's perception of differences between images, SSIM, we see that adding a KL term is beneficial, as the patch retrieval in terms of SSIM shows a peak at β_{norm} value of 10^{-4}. In our case, this translates to β value of 0.0032.

Secondly, we compare the patch retrieval capabilities to a classical autoencoder-based descriptor. We present our results in Table 3.2. We observe that the invertible descriptor is outperformed by the classical AE-based one in terms of PSNR, however, in terms of SSIM, the proposed β-VAE–based descriptor shows slightly better performance. These results are consistent with the β_{norm} value analysis, since setting the β_{norm} to 0 would correspond to using a regular autoencoder.

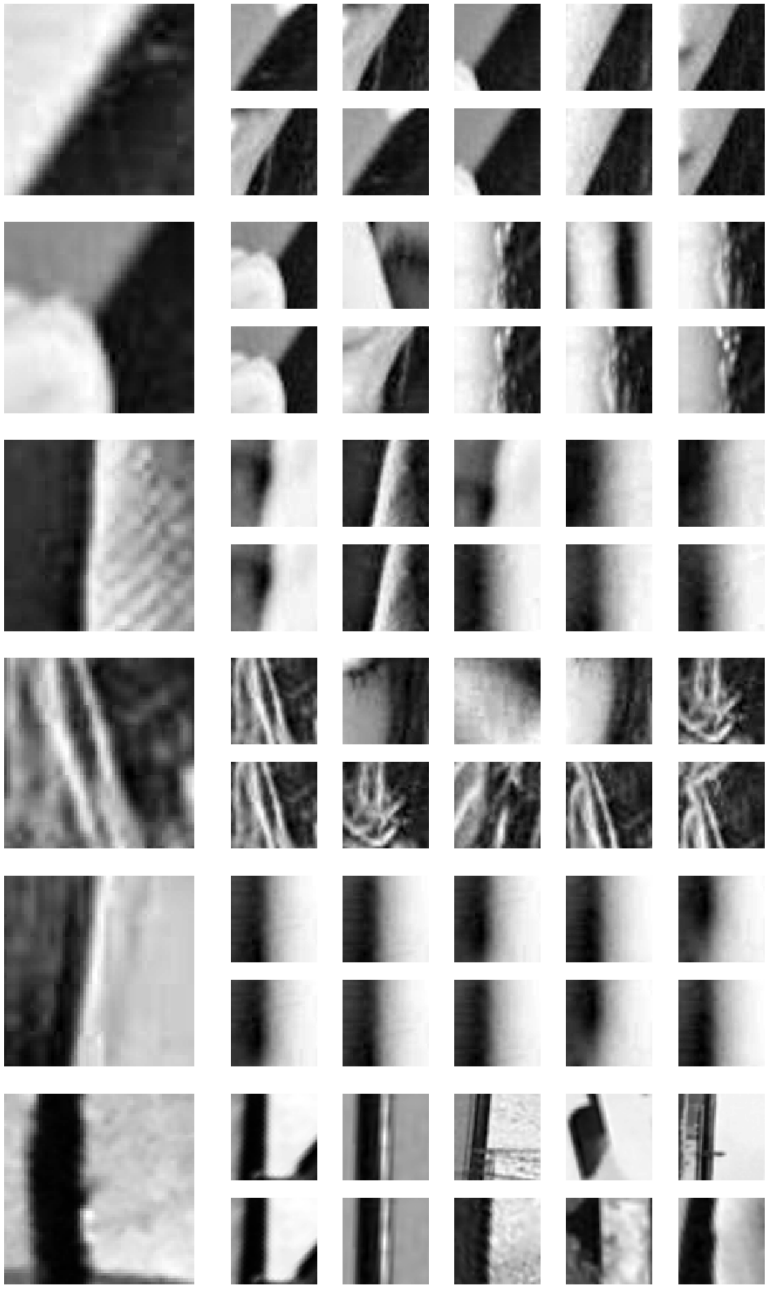


Figure 3.8: Patch retrieval examples. Large patch is the query patch. Top rows: classical AE-based descriptor; bottom rows: proposed invertible descriptor.

	PSNR [dB]	SSIM
Classical AE-based descriptor	**26.0**	0.23
Proposed invertible descriptor	24.6	**0.25**

Table 3.2: Patch retrieval performance comparison, averaged over 12.5k query patches.

According to these experiments, we can claim that the proposed invertible descriptor shows promising results in the main task for which descriptors are designed: retrieving patches.

3.5.1.2 Evaluation of invertibility

Now we evaluate the extent to which a descriptor can reconstruct the original patch from its encoding. For a test set of patches, we measure the average difference between the original patch, and the patch reconstructed from the encoding via the descriptor.

We again first show the analysis for different β_{norm} values (Figure 3.9 (right)). Here we see the best performance (in terms of both PSNR and SSIM) for the β_{norm} value of 10^{-4}. We conclude that the KL divergence (albeit weighted very lightly) has a positive influence on the invertibility of the descriptor. Therefore, using β-VAE for invertible local image descriptor indeed makes sense – we can benefit from the regularisation by the KL divergence term and also adjust the extent to which it is taken into account.

We also compare our descriptor to the autoencoder-based descriptor (Table 3.3). The proposed invertible descriptor shows better results than the classical AE-based one across both metrics: PSNR and SSIM. In Figure 3.10, we show some examples of patches reconstructed with the proposed VAE-based descriptor. We can observe that the proposed descriptor outperforms the reference method and is able to reconstruct the patches with improvements in capturing details and with less artefacts.

	PSNR [dB]	SSIM
Classical AE-based descriptor	16.0	0.10
Proposed invertible descriptor	**20.2**	**0.51**

Table 3.3: Patch reconstruction performance comparison on image patches cropped from images from ImageNet dataset, averaged over 12.5k patches.

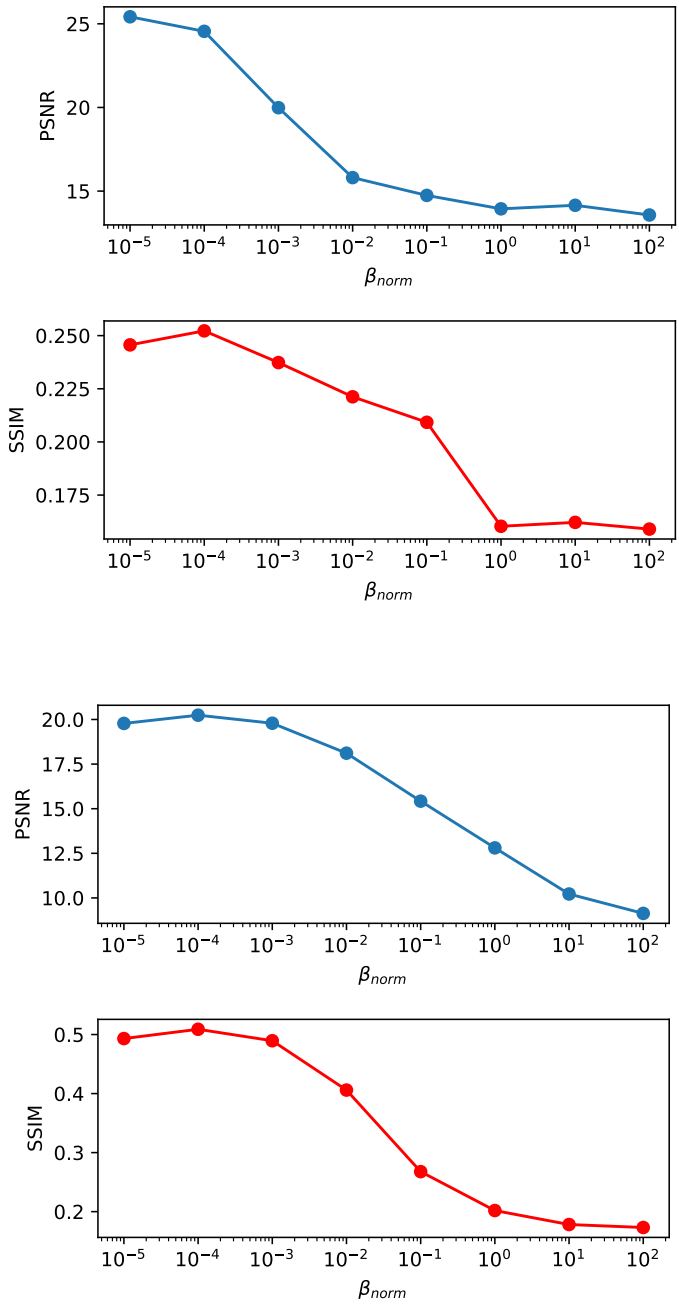


Figure 3.9: Comparison of patch retrieval performance (top two) and patch reconstruction performance (bottom two) for different β_{norm} values.

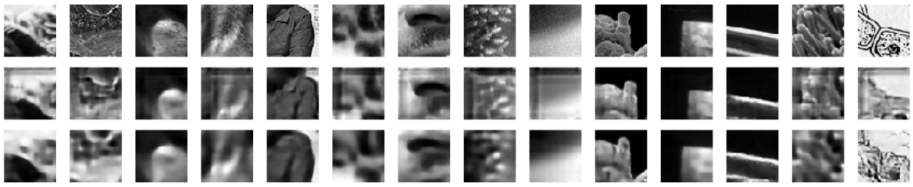


Figure 3.10: Examples of patch reconstruction based on the descriptor's encoding. Top row: original patches; middle row: reconstructed patches using a classical AE-based descriptor; bottom row: reconstructed patches using proposed invertible descriptor.

3.6 Conclusion

In this chapter, we carried out an analysis of the hyperparameters' influence on the performance of the descriptor learned with an autoencoder. We observed that autoencoders using MS-SSIM loss function, ReLU activation functions, and little to no data augmentation are producing the best descriptors in terms of performance on all HPatches tasks and for both classical and variational autoencoders.

In addition, we performed a thorough comparison between autoencoders and variational autoencoders approach for learning local image descriptors. We show that VAE-based descriptors produce marginally better results than AE-based descriptors on HPatches benchmark, but due to other difficulties that come with VAEs (extra hyperparameters to tune being the most prominent one), classical autoencoders are most likely a better choice for learning a descriptor.

We also investigated which metrics are the best for rapid evaluation of autoencoder-learned descriptors, and found SSIM distance between input and output patches of the autoencoder to show a consistently high correlation with the descriptors' performance on HPatches benchmark. We therefore propose using this metric for rapid approximate evaluation of local image descriptors learned with autoencoders, especially when trained by optimising the MS-SSIM loss.

Furthermore, we presented a novel descriptor based on β-variational autoencoders that is optimised for being inverted. We performed experiments to investigate the trade-off between descriptor's invertibility and its ability to preserve patch similarities in the latent space, and found that the proposed descriptor is capable of being inverted without this negatively affecting its performance as a descriptor.

The results in this chapter present a solid ground for further research and innovation in the field of self-supervised deep learning for local image descriptors, which we partake in the next chapter. The research in this chapter led to two published journal articles (one included in the Web of Science [Žižakić 21a] and another in a peer-reviewed international journal [Žižakić 21b]) and two conference papers (one of which is currently under review) [Žižakić 20b, Žižakić 22].

4

Memory-efficient autoencoder-based local image descriptors

Shall I describe it to you? Or would you like me to find you a box?
—Legolas, from *The Lord of the Rings: The Fellowship of the Ring*

While in the previous chapter, our focus was on optimising autoencoder-based and variational-autoencoder–based architectures for learning local image descriptors, here we introduce an architectural change to improve further their efficacy in image processing tasks. We modify the encoder part of the autoencoder to introduce a structure that we call *intermediate representation*. The key idea is to enable extracting an encoding of a single patch within the image with minimal computation, while having a representation that is not memory intensive.

4.1 Introduction

Over the recent years, camera resolution in smartphones has been increasing drastically. In 2022, even budget phones take 4K resolution images (around 8 megapixels [Goulekas 01]), while the camera resolution of the high-end phones is often over 50, or even 100 megapixels. As a consequence of this progress, image processing of photos taken by smartphones has become very computationally intensive, rendering some image processing tasks infeasible.

Let us, as an example, take image processing tasks that require patch matching, i.e. finding the most similar patches in an image for a query patch. Patch matching is used in tasks such as image denoising, inpainting, panorama stitching, motion estimation, etc. Comparing a query patch with all the possible patches in a 4K resolution image requires almost ten million patch comparisons for a standard patch size (65×65 pixels). If the patches are compared in a trivial way, i.e. by comparing each pixel of every patch, that results in tens of billions pixel comparisons, rendering it infeasible even for high-end processors that these phones have. A way to overcome this issue is to, instead of

comparing patches pixel by pixel, compare patches' descriptors – the compact encodings of the patch.

This way, we can reduce the number of comparisons from the order of magnitude of thousands to the order of magnitude of tens – a two orders of magnitude decrease. In order to obtain the patches' encodings, traditionally we have two options: (1) calculate and store the encodings for all the patches once before the start of the image processing algorithm or (2) calculate the encodings on demand every time patch comparison (or patch matching) needs to be performed.

Both of these approaches have their shortcomings. The first approach requires a large block of memory to be reserved for storing the patches' encodings, e.g. a single 4K image would call for several gigabytes of memory to store all the patches' encodings (for a standard 65×65 patch size). The reason for such high memory demand is that overlapping patches' encodings are, in general, not overlapping themselves, i.e. one needs to store a separate encoding for every single patch, even if an encoding exists for another one with 99% overlap. The second approach, i.e. calculating the encodings on demand, would partially defeat the purpose of using patch descriptors in the first place due to computational overhead – for each patch comparison, we would have to first calculate the patches' encodings using a descriptor, and then compare the encodings.

In this chapter, we propose a novel approach that improves significantly the efficiency of patch comparisons within a given image. The key idea is to create an *intermediate representation* as a compact way of storing the descriptors of all the patches of an image because the IRs of overlapping patches overlap themselves. For example, the IRs of two patches with 90% overlap will also have 90% overlap, and thus this overlap can only be stored in memory once. Extracting a descriptor from the intermediate representation is then done fast using only one max-pooling operation. We refer to the proposed intermediate representation as IR.

Intermediate representation is produced by a specific autoencoder-based architecture that we proposed, as will be described in the following sections. Using autoencoders allows our descriptors to be learned in a self-supervised way, enabling them to be fine-tuned for a particular use-case (type of data), e.g. photographs of paintings (as we will show at the end of this chapter). The descriptors learned in a supervised way or designed using hand-crafted features do not support such specialisation based on the type of data at hand. For a more detailed overview of the advantages of local image descriptors learned in a self-supervised way, see Section 3.1.

We build on top of the results from the previous chapter, where we performed a thorough analysis to find the optimal hyperparameters for (variational) autoencoders for learning local image descriptors. We also use the rapid approximate evaluation method (proposed in the previous chapter) to efficiently evaluate the descriptors' performance during the training of our (variational) autoencoders.

As a proof of concept for the proposed intermediate representation, we integrate our descriptor into an existing inpainting algorithm [Ružić 15] to show the benefit of this representation. We hypothesise that the improved inpaint-

ing results come from being able to search over a larger patch space – made possible due to our memory-efficient descriptor. To achieve such fine-tuning with other (supervised) descriptors, it would be necessary to have a labelled set for the type of images that need to be inpainted, which is unrealistic in most cases. As a case study, we used high-resolution photographs of the panels of Ghent Altarpiece [KIK-IRPA 10], on which we fine-tuned the descriptor and tested our improved inpainting algorithm.

The rest of this chapter is organised as follows. In Section 4.2 we present our proposed autoencoder architecture that yields the desired intermediate representation structure. In Section 4.3, we present the experimental evaluation of our method in three different ways: by evaluating it on an established descriptor benchmark (Section 4.3.1), by evaluating its robustness to noise (Section 4.3.2) and to missing data (Section 4.3.3). We integrate our proposed method into an inpainting algorithm in Section 4.4. Finally, we conclude this chapter in Section 4.5.

4.2 Reducing computational memory with intermediate representation

We propose the architecture of the encoder part of our autoencoder where we discard all the max-pooling layers in the encoder except for the last one, whose spatial extent we increase (in order to sufficiently reduce the dimension of the encoding layer – the last layer of the encoder). The intermediate representation is then the data structure that is obtained after the last convolutional layer and just before the max-pooling layer, and the patch encodings are obtained from the intermediate representation using a single max-pooling operation.

Formally, let $I := I^{(0,:)}$ be the input image, where with superscript $(0,:)$ we denote all the channels of the 0-th layer of the encoder. We define the intermediate representation $\mathcal{IR}(I)$ as:

$$\mathcal{IR}(I) = I^{(L,:)}, \tag{4.1}$$

$$I^{(L,c)} = \mathcal{A}(\mathcal{C}_{l_L}(\mathcal{A}\ldots(\mathcal{C}_{l_1}(I^{(0,:)})))). \tag{4.2}$$

where L is the number of convolutional layers in the encoder $\mathcal{E}$, $I^{(l_i,c)}$ is the c-th channel of the output of the l_i-th layer, $\mathcal{A}$ is some activation function, and $\mathcal{C}_{l_i}$ is the l_i-th convolutional layer.

From the intermediate representation of an image $\mathcal{IR}(I)$, we obtain the descriptor for a patch $I_{[i..i+p][j..j+p]}$ whose top left corner is in position (i,j), where p is the patch size, as follows

$$\mathcal{E}(I_{[i..i+p][j..j+p]}) = \mathcal{MP}(\mathcal{IR}(I)_{[i..i+p][j..j+p]}). \tag{4.3}$$

We now discuss the impact of removing all the max-pooling layers (except for the last one). Using max-pooling is usually motivated by its advantages – adding non-linearity, playing the role of dimensionality reduction, and with

that reducing the number of parameters to be trained and hence the training time. However, it has been shown that (max-)pooling is not necessary for a successful neural network, and other methods have been proposed to replace it [Tobias-Springenberg 14]. Indeed, non-linearity between layers is already achieved with non-linear activation functions. Dimensionality reduction is a crucial property of autoencoders and thus we do leave one max-pooling layer with large spatial extent at the end of the encoder to reduce the dimension of the code layer. The longer training time due to removing other max-pooling layers is a trade-off for decreasing the computational time and memory while using the descriptor.

The proposed approach is beneficial in image processing problems that require many patch comparisons within a single image. IR is obtained by propagating the complete image (containing patches of interest) through the convolutional layers in the encoder, but not the max-pooling. This is done only once and before the actual processing starts. During the particular image processing task, the descriptors are extracted from the stored IR using the fast max-pooling operation on the corresponding section of the IR. Figure 4.1 shows this process visually and Figure 4.2 shows the architecture of our network and the IR.

The memory reduction is achieved due to it being sufficient to store only the IR of the whole image (the IRs of patches are simply cropped from the IR of the whole image in the same way a patch would be cropped from an image) and to get the descriptors on demand using the fast max-pooling in contrast to storing an encoding for a patch at each possible location in an image (Figure 4.1). It is not necessary to separately store IRs of all the patches because the IRs of neighbouring patches are overlapping (in the same way neighbouring patches are overlapping in an image). Conversely, this is generally not the case for the encodings of other descriptors – even for two patches shifted by one pixel in an image, their two encodings may be arbitrarily different and one would therefore need to store them both in memory. IR thus results in a tremendous decrease in memory usage for image processing applications, as shown in Figure 4.4. This decrease could make some algorithms that use many patch comparisons feasible for use on large images.

Our architecture does, however, require slightly longer training time compared to the one with the standard use of max-pooling (measured in minutes – amounting to less than 5% increase for the additional training time), but it reduces the computational time and memory while using the descriptor. We accept the incurred computational time of training the autoencoder (which can be performed only once) as a trade-off for huge savings in the computational memory while using the descriptor.

To summarise, the idea of the proposed intermediate representation is to allow IR-based descriptors to save memory (with respect to non-IR descriptors) while keeping their performance on the same level. In the next section, we show both the memory savings and the performance of IR descriptors that is comparable to that of non-IR descriptors.

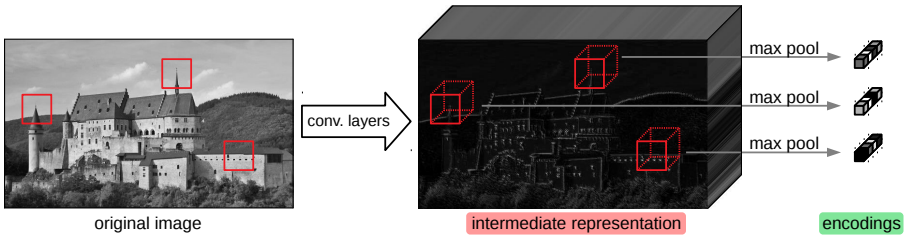


Figure 4.1: Exploiting the proposed intermediate representation (IR) of an image in algorithms that require many patch comparisons. The IR is calculated once from the original image through the convolutional layers of the encoder. In algorithms that need to compare patches (e.g. inpainting), the descriptors are extracted from the IR using the fast max-pooling operation, and then compared.

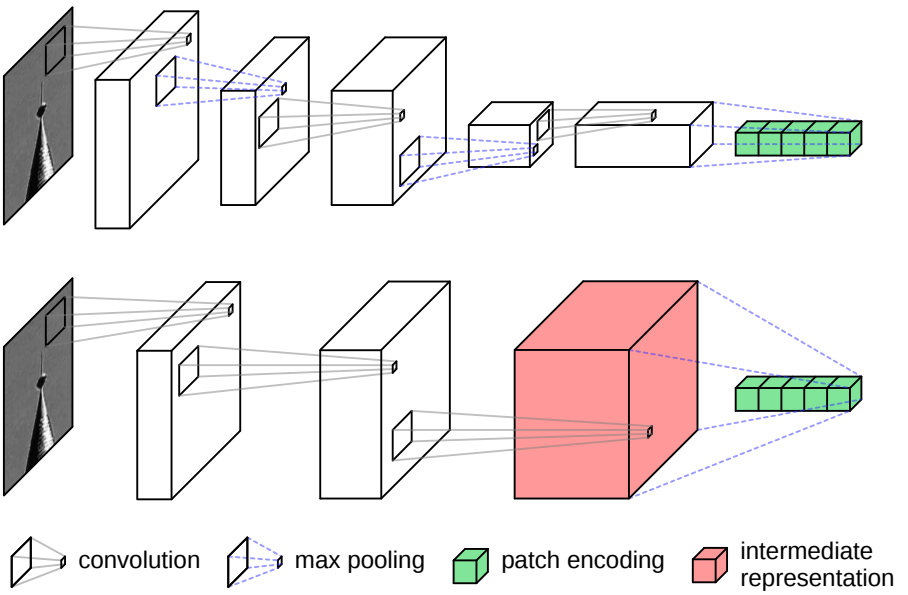


Figure 4.2: Top: traditional encoder architecture in autoencoders, with max-pooling layers after all convolutional layers. Bottom: proposed encoder architecture, which omits max-pooling layers after all the convolutional layers but the last one, in order to obtain an intermediate representation (IR) of image that preserves the spatial information in the height-width plane.

4.3 Experimental results

We evaluate the proposed descriptors in multiple ways. In section 4.3.1, we use the popular HPatches benchmark for the evaluation of local image descriptors [Balntas 17]. The idea is to compare the proposed IR descriptors with analogous non-IR descriptors (both of the types learned without labelled data), as well as with state-of-the-art supervised-learning–based descriptors and hand-crafted descriptors (to put the results of these self-supervised methods into perspective).

In Section 4.3.2, we evaluate descriptors' robustness to noise – how they behave when noise is added to the patches. This property is not only important for applications in image inpainting, but also for image denoising. In Section 4.3.3, we evaluate descriptors' robustness to missing data, i.e., how they behave when parts of patches are missing. The robustness to missing data is an important property for applications in image inpainting. In the next section (Section 4.4) we plug in our descriptor into an inpainting algorithm to showcase these properties and how they improve the inpainting.

4.3.1 Evaluation on HPatches benchmark

In this section, we use the well-established HPatches benchmark [Balntas 17] to evaluate the performance of different descriptors. We examine the performance of descriptors that incorporate our proposed IR method and compare their performance with their non-IR counterparts, as well as with the only other AE-based descriptor reported so far, from Chen et al. [Chen 15]. To put the results of these self-supervised methods into perspective, we also compare them with a state-of-the-art supervised descriptor DeepDesc [Simo-Serra 15] and with an established hand-crafted descriptor SIFT [Lowe 99]. The goals of this evaluation are (1) to show comparison between AE-based and VAE-based descriptors, (2) to show comparison between IR-based and non-IR–based descriptors, (3) to show how our (V)AE-based descriptors compare to the only other AE-based descriptor (from Chen et al. [Chen 15]), and (4) to show how self-supervised-learning–based descriptors compare to the state-of-the-art supervised descriptors and hand-crafted descriptors. We report both performance evaluated on HPatches benchmark and performance in terms of memory requirements while using these descriptors as part of an image processing algorithm on a single image.

We have selected the best performing models based on the analysis of hyperparameters in Section 3.3 (for both models we choose ReLU activation function, no data augmentation, MS-SSIM loss function and for the VAE β value of 0.0001).

Figure 4.3 shows descriptors' performance evaluated on all three tasks of HPatches benchmark. We can conclude that incorporating the IR architecture into a (V)AE-based descriptor does not significantly impact its performance as evaluated on HPatches benchmark – AE-IR version slightly outperforms AE while VAE slightly outperforms VAE-IR. All four models' performance is very similar, with VAE and AE-IR exhibiting the best performance. However, the

memory-efficiency of the IR versions is significantly higher (as shown in Figure 4.4). The similarities between these methods can be seen in figures 4.5 and 4.6, which shows examples of the retrieved patches that are the most similar to a query patch for all our methods. The patches were retrieved using the descriptors, based on the similarities of patch encodings.

We also observe that all our four models outperform the descriptor trained with autoencoders from [Chen 15]. However, it is clear from the Figure 4.3 that there is still a gap in performance between these self-supervised models on one hand and supervised and hand-crafted models on the other hand. This is to be expected given that the supervised models can leverage labels that show semantic similarities between patches without these patches being very similar in terms of their pixel values. These results also suggest that there is further room for improvement and innovation in the (V)AE-based descriptors in order to bridge the peformance gap with the hand-crafted ones.

When it comes to the memory requirements the proposed IR-based descriptors (both AE-based and VAE-based) show a clear advantage over those that do not use IR, such as the descriptor from Chen et al. [Chen 15], SIFT [Lowe 99], DeepDesc [Simo-Serra 15], and descriptors trained with regular AEs and VAEs (Figure 4.4). This is because our method takes advantage of the fact IRs of overlapping patches are overlapping themselves, so it is enough to simply store IR of the whole image and then extract from it descriptors using max pooling. Therefore, we can confidently say that IR architectural change is beneficial in image processing tasks on a single image because it shows a significant reduction in memory usage while keeping the descriptor's performance on the same level.

4.3.2 Robustness to noise

We test the noise robustness of two versions of our descriptor, v32 and v128 (named after the dimensionality of the descriptor for 16×16 patches). We compare them to the exhaustive search on pixel intensity values, and to the existing descriptor trained with the autoencoder of [Chen 15]. We trained all the descriptors on the dataset described in Section 3.3.2.

The evaluation is performed as follows. We select a set of query patches within an image with added Gaussian noise. For each query patch, we retrieve the k most similar patches either by comparing their descriptors or by using exhaustive search over the pixel values. The quality of patch retrieval is evaluated based on the sum of square differences (SSD) between the pixels in query and retrieved patches before the noise was added. The standard deviation of the Gaussian noise was varied between 0 and 50.

Figure 4.7 summarises the results of our patch retrieval experiments and some visual examples are shown in Figure 4.8 (top two examples). When no noise is present, the exhaustive search retrieves the patches that are the most similar to the query patch. However, noise deteriorates the performance of the exhaustive search, whereas our descriptor v128 shows little decrease in performance.

Our descriptors also show superior performance compared to the existing

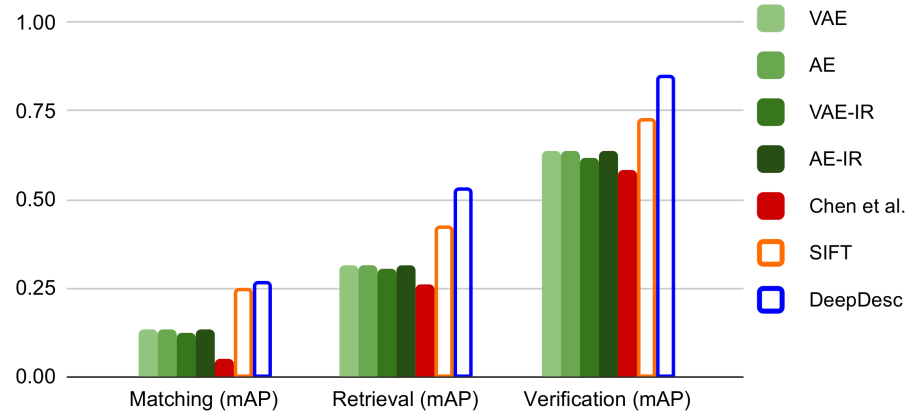


Figure 4.3: Comparison of performance on HPatches tasks (matching, retrieval and verification) between descriptors learned with VAE and AE (both non-IR and IR variants), a different autoencoder-based descriptor from Chen et al. [Chen 15], hand-crafted descriptor SIFT [Lowe 99] and a supervised-learning–based descriptor DeepDesc [Simo-Serra 15] which achieves state-of-the-art performance [Balntas 17].

Figure 4.4: Memory needed for storing patch encodings with descriptors that do not incorporate IR architecture (Chen et al. descriptor, descriptors trained with regular AEs and VAEs) for all patches in an image compared to the memory for storing the intermediate representation of the image (showing versions where encodings are of length 32 and 128, respectively), from which a descriptor for a single patch can be obtained with minimal computation.

Figure 4.5: Patch retrieval examples. Large patch is the query patch. Rows (top to bottom): AE-based descriptor, VAE-based descriptor, AE-based descriptor with IR, VAE-based descriptor with IR.

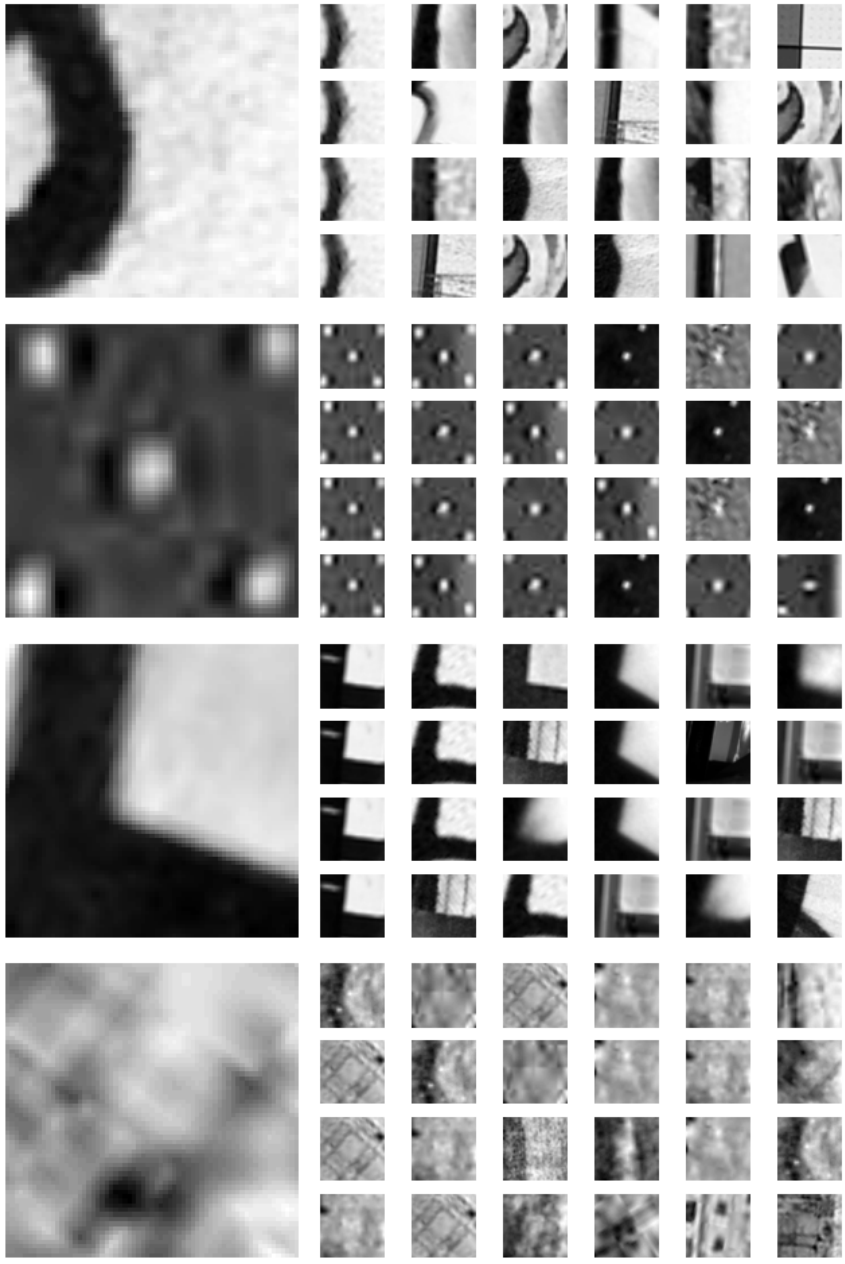


Figure 4.6: Patch retrieval examples. Large patch is the query patch. Rows (top to bottom): AE-based descriptor, VAE-based descriptor, AE-based descriptor with IR, VAE-based descriptor with IR.

descriptor learned with autoencoders. Our method v128 shows better results than Chen [Chen 15], while having the same patch descriptor dimensionality. Furthermore, our method v32 that shows similar results to [Chen 15] has an order of magnitude lower dimensionality of the descriptor when encoding a single patch. The dimensionality comparison changes even more in our favour when encoding the whole image due to the usage of the IR (Figure 4.4).

4.3.3 Robustness to missing data

We set up an experiment to determine the capability of the proposed descriptor when working with patches that contain missing regions. This type of operation is of interest for applications such as inpainting and scene reconstruction from multi-view data. The setup is similar to the noise robustness evaluation, but here parts of the query patches have been randomly removed. We trained all the descriptors on the dataset described in Section 3.3.2.

For the query patches with missing parts we want to find the best matching undamaged patches. We are searching for the matching patches by comparing the descriptors of the non-missing parts. The numerical evaluation is done based on the SSD values of the complete (undamaged) query and the found match. The results are shown in Figure 4.7 (bottom), again comparing our two descriptors, descriptor from [Chen 15], and the exhaustive search over pixel intensity values. Visual comparison is shown in Figure 4.8 (bottom two examples).

The conclusions from these experiments are similar to those with the noisy patches. When the missing area in a patch is small, exhaustive search retrieves the best results. However, as the missing area is increasing, our descriptor v128 starts performing better and overtakes the exhaustive search, showing more robustness to missing data than the exhaustive search. Both of our proposed methods outperform the existing method [Chen 15], with a slightly larger margin than in the case of noisy patches. In the following section we explore how our descriptors perform when being part of an inpainting algorithm.

4.4 A case study – inpainting

We illustrate the use of our proposed IR-based local image descriptors in image inpainting. The goal of image inpainting is to reconstruct the missing region of an image in a visually plausible way using the information from the surrounding regions of the image. Exemplar-based (sometimes called patch-based) inpainting methods fill in the missing region by sampling and copying the patches from the undamaged part of the image. These methods require many patch comparisons in order to find appropriate patches to be used to fill in the missing area. The patch size normally used is between 10 and 20 pixels. While it is usually selected based on the image resolution, in some cases the optimal patch size may depend on the image content (see, e.g., the discussion in [Ružić 15]). Automatic choice of the patch size based on the image content is out of the scope of this thesis. As the image dimensions are growing nowadays (with some of the bleeding-edge mobile phones having 100-megapixel cameras),

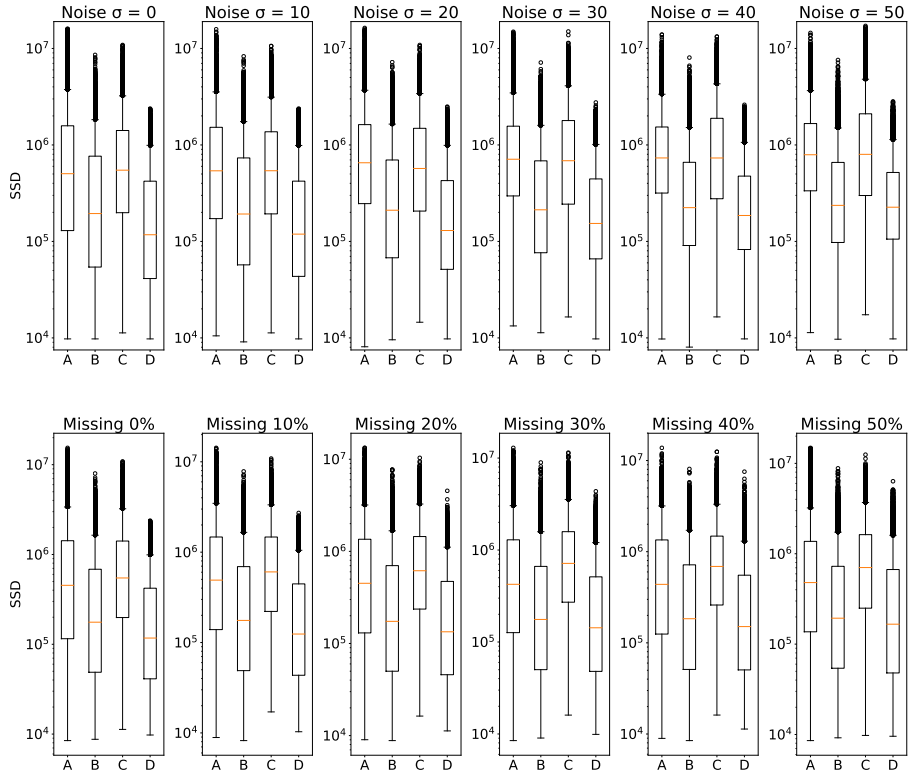


Figure 4.7: Comparison of descriptors' robustness to noise (top) and missing data (bottom). A – proposed descriptor v32, B – proposed descriptor v128, C – Chen et al. [Chen 15], D – exhaustive search on pixel intensity values. The plots are showing SSDs of ground truth pixel values of patches found by the descriptors (in A-C) and exhaustive search (D), based on the noise (top) and percentage of missing area in a patch (bottom).

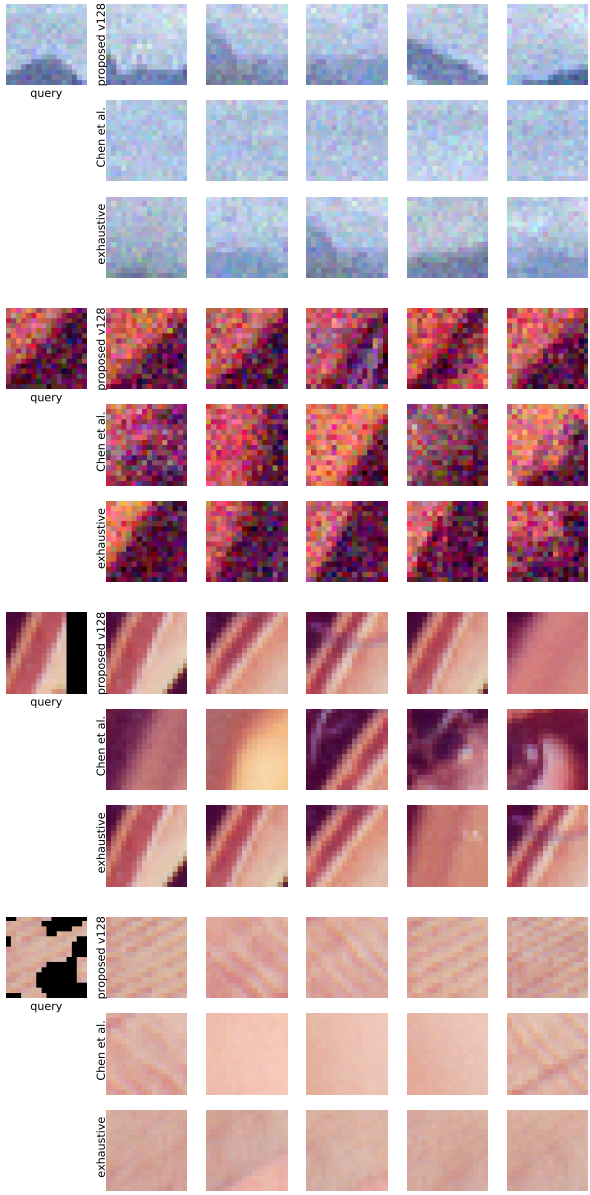


Figure 4.8: Noisy patch retrieval (top two) and patch retrieval where the query has missing parts (bottom two). For each query, the first row corresponds to the proposed descriptor v128; the second row: the descriptor from [Chen 15], and the third row: exhaustive search. The missing parts of the query patches (bottom two) are shown in black.

these inpainting algorithms are becoming infeasible. This presents a window of opportunity for the local image descriptors to speed up these algorithms and make them usable in the high-definition context of the images that we have today.

Patch-based inpainting algorithms including [Criminisi 04, Komodakis 07, Le Meur 11, Voronin 14, Ružić 15, Ghorai 16, Newson 17, Zhang 19] search for well-matching candidate patches within some search window or within image segments with given textural and colour characteristics (searching for well-matching candidates in the whole image would be too computationally intensive). We search instead well-matching patches using our learned patch descriptor and we incorporate this patch search into the inpainting algorithm from [Ružić 15].

We test the resulting method in virtual restoration of master paintings. As a case study, we use images from the panels of the *Ghent Altarpiece* [KIK-IRPA 10], on which a larger virtual restoration study is being performed [Pižurica 15, Sizyakin 20]. The paint-loss areas to be inpainted are detected with the algorithm from [Meeus 19].

Figure 4.10 shows the inpainting on a part of the panel *the Prophet Zachary*. On this particular panel, the paint-loss areas are showing as light brown. Figure 4.9 shows the zoomed details. We have also used the inpainting algorithm without the descriptors, however, we were not able to obtain the inpainting results on the whole panel without using the descriptor due to the memory error on our computer (an AMD Ryzen Threadripper 1920X 12-core processor with 32GB of RAM).

The inpainting results are very promising and show that our descriptor was both able to improve visually the inpainted images as well as the computational aspect of the inpainting.

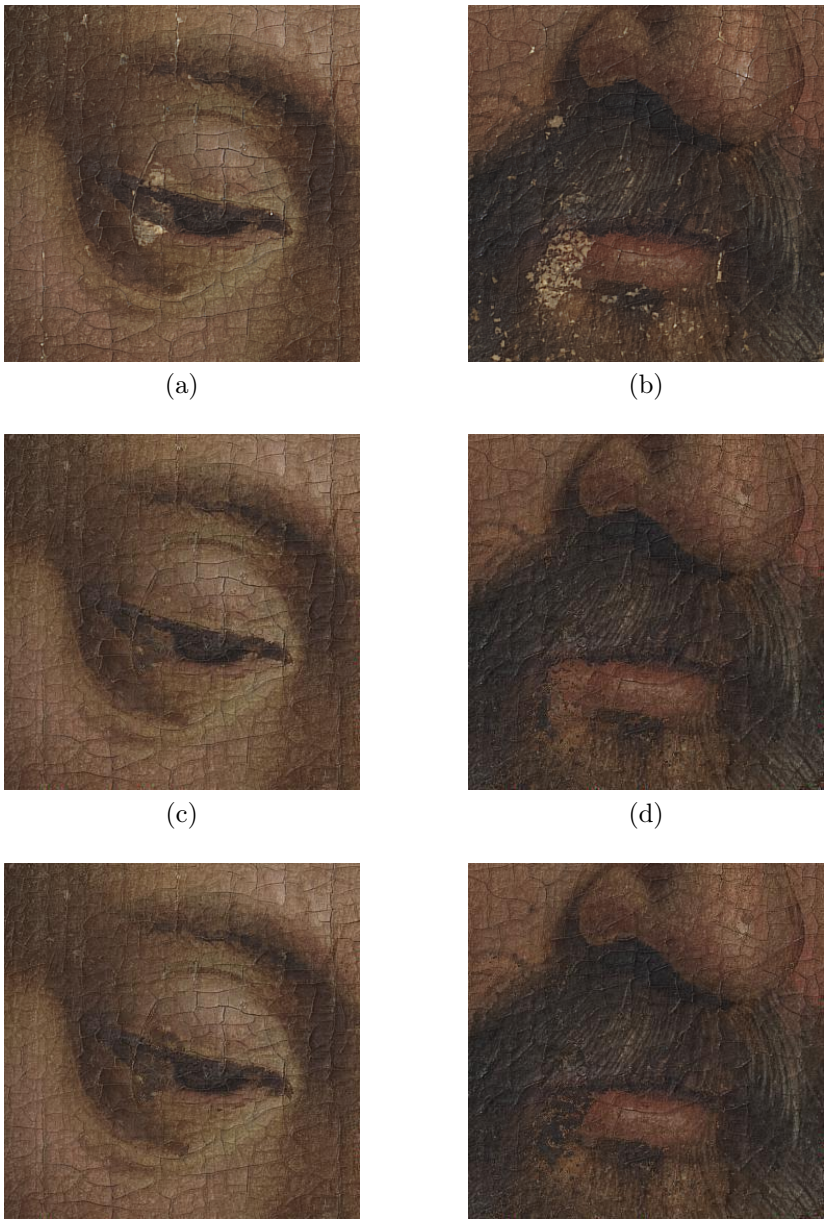


Figure 4.9: Image inpainting results. (a)-(b) Original image detail containing paint-loss (showing as light brown); (c)-(d) inpainted with a patch-based method using the proposed local image descriptor; (e)-(f) inpainted without using the descriptors. Image copyright: Ghent, Kathedrale Kerkfabriek, Lukasweb; photo courtesy of KIK-IRPA, Brussels.

(a) Original

(b) Paint loss areas to be inpainted

(c) Inpainted with a patch-based method using the proposed local image descriptors.

Figure 4.10: Image inpainting results. Image copyright: Ghent, Kathedrale Kerkfabriek, Lukasweb; photo courtesy of KIK-IRPA, Brussels.

4.5 Conclusion

In this chapter, we proposed an intermediate representation (IR) structure – an improvement to the encoder architecture of an autoencoder which produces descriptors that perform better or equally as good but save memory in comparison to existing methods when used for patch search and matching within a single image.

We have evaluated our IR-based descriptor on HPatches benchmark comparing it not only to other self-supervised-learning–based descriptors, but also with state-of-the-art supervised descriptors and hand-crafted descriptors. The evaluation shows that our method performs better or equally as good as other self-supervised-learning–based methods, while improving on the memory-efficiency of storing the descriptors. There is still a gap in performance between these self-supervised models on one hand and supervised and hand-crafted models on the other hand, leaving room for further innovation in the (V)AE-based descriptors.

We have also evaluated the proposed descriptors' robustness to noise and missing data against an existing descriptor learned with autoencoders from [Chen 15] and exhaustive search over pixel intensity values. The proposed descriptors show improved results, and superior robustness to both noise and missing data in comparison with exhaustive search.

As a proof of concept, we have integrated our descriptor into an inpainting algorithm that resulted in visual improvements over the inpainted images and the ability to handle higher resolution images.

The research in this chapter contributed to three conference papers [Žižakić 19a, Žižakić 19b, Meeus 20] and one journal paper [Žižakić 21a].

5

Transferring the knowledge from hand-crafted to learning-based descriptors

My precious.
—Gollum, from *The Lord of the Rings: The Two Towers*

While in the previous chapters, we were focusing on local image descriptors designed only to learn from data, in this chapter, we present a novel approach for designing descriptors that learn not only from data but also from hand-crafted descriptors. The proposed method is a type of a knowledge transfer (as will be described later in this section) which has, to the best of our knowledge, not been considered before. In particular, we construct a learning model that first mimics the behaviour of a hand-crafted descriptor and then learns to improve upon it in a self-supervised manner, using autoencoders. This is a general framework for knowledge-transfer from hand-crafted to learning-based descriptors, and we demonstrate its use by constructing the *learned BRIEF* descriptor based on the well-known hand-crafted descriptor BRIEF.

5.1 Introduction

As we have discussed in Chapter 3, local image descriptors play a crucial role in many image processing tasks and can be categorised into hand-crafted and learning-based methods. While learning-based methods show superior performance on benchmarks [Balntas 17, Fischer 14b], hand-crafted descriptors still outperform the learned ones in many cases of practical interest and are often a descriptor of choice in practice [Schonberger 17]. Therefore, the hand-crafted methods should not be neglected in research, but used to improve the learning-based methods.

In this chapter, we present a novel framework for constructing a learning model that first mimics the behaviour of a hand-crafted descriptor and then learns to improve upon it in a self-supervised manner. Two important assets of the proposed approach compared to most of the current descriptor learning

methods are: (1) improved explainability, inherited from a chosen hand-crafted descriptor that it builds upon, and (2) no need for excessive labelled datasets. Compared to the hand-crafted descriptor, we achieve better performance and avoid the need for various ad-hoc choices of the parameters. Due to the self-supervised nature of our method, the descriptor can be fine-tuned for various applications without the specific labelled dataset for that application.

The framework that we present is a form of knowledge transfer. Knowledge transfer is a term used in machine learning community where a neural network is bootstrapped from an initial training based on, for example, a small or a private dataset [Papernot 16]. In the case of our framework, this bootstrapping process is performed by training the network using a hand-crafted descriptor as a ground truth. The network is then further trained in a self-supervised manner as part of an autoencoder.

In a different context, the connections between model-based approaches and deep learning have been studied by unrolling the iterative optimisation algorithms [Gregor 10, McCann 17, Monga 21]. In a seminal work [Gregor 10], it was shown that an iterative optimisation algorithm like iterative thresholding (ISTA) [Daubechies 04] can be unrolled (unfolded) into an equivalent representation using a convolutional neural network. This research domain is currently very active, with many applications in image reconstruction and compressed sensing [McCann 17], and in various inverse imaging problems [Monga 21]. These CNN-based versions of the unrolled iterative optimisation algorithms are not only much faster than their underlying counterparts but also allow for an elegant end-to-end optimisation of the parameters. Moreover, the deep learning networks constructed this way typically outperform the original iterative algorithm. In fact, it can be shown [McCann 17] that a properly trained deep learning model will always perform at least as well or better than the underlying model-based algorithm. In the same way, we claim that our learned autoencoder-based local image descriptor, with proper training, will be superior to the hand-crafted one that served as its initialisation. We will support this claim with a thorough evaluation on the widely used benchmarks for patch descriptors.

Our method is a general framework for transferring the knowledge from an arbitrary local image descriptor. In this chapter, we use the proposed framework to transfer the knowledge from the BRIEF descriptor [Calonder 10], which enjoys popularity as a computationally simple and efficient descriptor. However, our method can be applied on other descriptors too, as we will explain in the Section 5.2.

To summarise, the main contributions in this chapter are twofold:

- We propose a framework for the transfer of knowledge from a hand-crafted descriptor to a self-supervised-learning–based descriptor. At the time of writing, no such framework has been reported, to the best of our knowledge.
- Within this framework, we propose an elegant method for implementing a learned BRIEF.

The rest of this chapter is organised as follows. In the following section, we

introduce our framework for knowledge transfer from hand-crafted to learning-based descriptors. In Section 5.2, we first describe BRIEF descriptor from [Calonder 10] (Section 5.3.1), and then we apply our framework to construct our *learned BRIEF* descriptor (Section 5.3.2). In Section 5.4 we evaluate the performance of our learned BRIEF in comparison with the original BRIEF descriptor. Section 5.5 concludes this chapter.

5.2 Framework for knowledge transfer from hand-crafted descriptors

The first contribution that we present in this chapter is a framework for the transfer of knowledge from a hand-crafted descriptor to the self-supervised-learning–based descriptor. The framework we propose consists of four steps (Figure 5.1):

Step 1 – Construct and optimise a neural network $\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}$ that mimics patch encoding produced by a hand-crafted descriptor $\mathcal{H}$. For a set of patches $\mathcal{P}$ and loss function $\mathcal{L}$ where the loss function depends on the activation function chosen for the last layer, solve:

$$\min_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}} \sum_{\mathbf{p}\in\mathcal{P}} \mathcal{L}(\mathcal{H}(\mathbf{p}), \mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}(\mathbf{p}))$$

Step 2 – Set $\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}$ to be the encoder $\mathcal{E}$ of an autoencoder, and train the decoder $\mathcal{D}_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}}$, minimising the loss between its output and the input patch:

$$\min_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}} \sum_{\mathbf{p}\in\mathcal{P}} \mathcal{L}(\mathbf{p}, \mathcal{D}_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}}(\mathcal{E}(\mathbf{p})))$$

Step 3 – Train the whole autoencoder, consisting of the encoder $\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}$ and the decoder $\mathcal{D}_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}}$:

$$\min_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E},\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}} \sum_{\mathbf{p}\in\mathcal{P}} \mathcal{L}(\mathbf{p}, \mathcal{D}_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}}(\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}(\mathbf{p})))$$

Step 4 – Fine-tune the whole autoencoder on a dataset $\mathcal{P}_{Spec}$ specific to the desired application:

$$\min_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E},\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}} \sum_{\mathbf{p}\in\mathcal{P}_{Spec}} \mathcal{L}(\mathbf{p}, \mathcal{D}_{\mathbf{w}_\mathcal{D},\mathbf{b}_\mathcal{D}}(\mathcal{E}_{\mathbf{w}_\mathcal{E},\mathbf{b}_\mathcal{E}}(\mathbf{p})))$$

The first step is specific to the hand crafted descriptor that is chosen for knowledge transfer, and may involve creating specific neural network architectures that are functionally similar to the hand-crafted descriptor. In some cases, as we will show is the case with BRIEF in subsection 5.3.2, it is possible to completely implement the hand-crafted descriptor using standard neural network layers (convolutional, fully-connected). In other cases, it may be necessary to make the network learn to output the descriptor using training. In this case, the network would be trained on a set of input patches until its output was sufficiently close to the output of the hand-crafted descriptor.

In the second step we train the decoder network to learn to reconstruct the patch from its encoding, similarly to what Mahendran et al. [Mahendran 15] did, but using a simple CNN without any regularisation. We found that, for reconstructing BRIEF, this setup achieves a good starting point for the training of the complete autoencoder, as we will discuss in subsection 5.3.2.

Next, in the third step, we simultaneously train both the encoder and decoder parts of the autoencoder. We train the autoencoder on the same general dataset of patches used in both Step 1 and Step 2. In order for the learned descriptor to achieve rotation and translation invariance, it is possible to add geometric noise to the input patches while keeping the output patches the same. We further discuss the effect of the geometric noise adding in the Section 5.4.

In the final, fourth step, the autoencoder is fine-tuned from the previous step on an application-specific set of patches.

After the training is complete, the decoder part of the network can be discarded, and we are left with the encoder network, which is our local image descriptor.

In this chapter, we demonstrate our framework using BRIEF as the hand-crafted descriptor from which we transfer the knowledge into a neural network.

5.3 A learned variant of BRIEF

We first describe the BRIEF descriptor from [Calonder 10] (Section 5.3.1), and then we show how we apply our general framework to design the proposed learned BRIEF descriptor (Section 5.3.2).

5.3.1 BRIEF descriptor

BRIEF (Binary Robust Independent Elementary Features) is a binary local image descriptor that, for an input patch, calculates its binary code [Calonder 10]. The code is calculated as follows (see Figure 5.2-a (top)). First, the input patch is smoothed in order to decrease the sensitivity to noise. The smoothing is done using an averaging kernel of size 9×9. Then, a binary feature vector is created, composed of the binary test responses. A binary test τ between pixels coordinates x and y on a patch $\mathbf{p}$ is defined by:

$$\tau(\mathbf{p}; x, y) = \begin{cases} 1 & \mathbf{p}(x) \geq \mathbf{p}(y) \\ 0 & \mathbf{p}(x) < \mathbf{p}(y) \end{cases} \tag{5.1}$$

The image patch is then encoded as the n_d-dimensional bit string:

$$f_{n_d}(\mathbf{p}) = \sum_{1 \leq i \leq n_d} 2^{i-1} \tau(\mathbf{p}; x_i, y_i). \tag{5.2}$$

The pairs of pixel coordinates (x_i, y_i) are drawn from a Gaussian distribution around the centre of the patch: $(\mathbf{X}, \mathbf{Y}) \sim$ i.i.d Gaussian$(0, \frac{1}{25}s^2)$, where $s \times s$ is the size of the patch. In the case of OpenCV implementation of BRIEF, these pairs of coordinates are then hard-coded and used across all configurations of the descriptor.

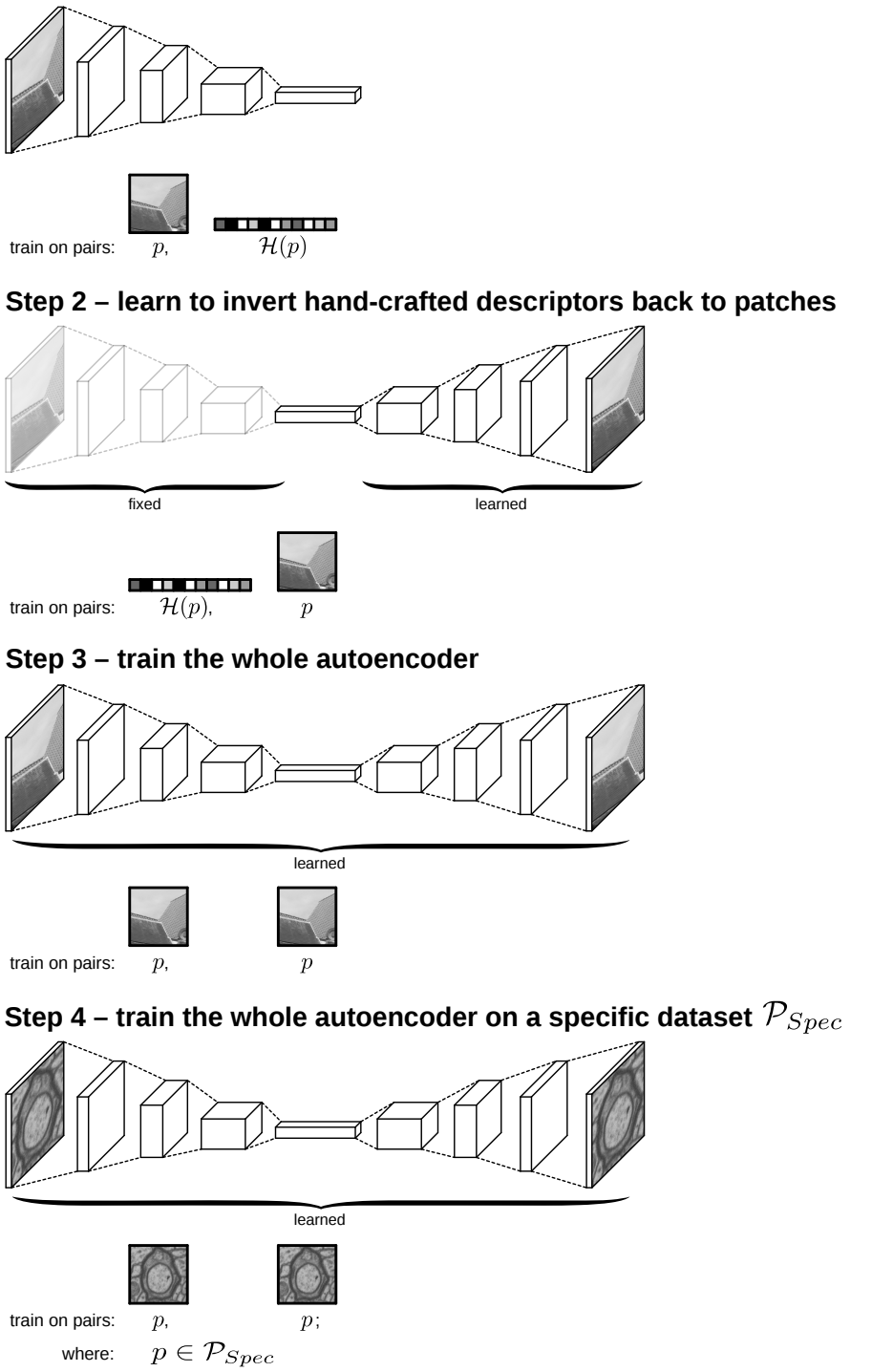


Figure 5.1: General framework for knowledge transfer from hand-crafted to self-supervised-learning–based descriptors.

5.3.2 Learned BRIEF

We now construct a learned variant of the BRIEF descriptor using the proposed framework.

We mostly focus on describing Step 1, since it is specific to the BRIEF descriptor. This step involves realising the original BRIEF descriptor as a convolutional neural network. In the case of BRIEF, in order to achieve a network that mimics the descriptor's output, this step does not necessitate training the network but simply setting the weights and biases of the network to match BRIEF's behaviour. The architecture of this neural network is comprised of several convolutional layers that implement BRIEF's average blurring, and one fully-connected layer that implements BRIEF's binary tests (intensity comparisons between pixels).

In the original implementation of BRIEF, the average blurring is performed with a 9×9 averaging kernel, i.e., in the deep learning terminology, a convolutional layer with one kernel of the size 9×9. In our implementation, we use four convolutional layers, each with one 3×3 averaging kernel, instead of a single layer with a 9×9 averaging kernel. This architecture is commonly used in modern CNNs and yields a sufficiently similar blurring to the blurring used in BRIEF. The activation functions of the convolution layers are rectifier linear units which have no effect at this point (since all the values of the kernel are greater than zero), but serve as non-linearity during the retraining of the network.

The binary pixel tests are implemented as a fully-connected layer (preceded by flattening the output from the previous convolutional layer in raster-scan fashion). Binary pixel test τ defined in (5.1) can be reformulated as:

$$[\sigma(\mathbf{p}(x) - \mathbf{p}(y))], \tag{5.3}$$

where $\sigma(\cdot)$ is the sigmoid function and $[\cdot]$ denotes rounding to the nearest integer. The proof follows:

$$[\sigma(\mathbf{p}(x) - \mathbf{p}(y))] = \left[\frac{1}{1 + e^{-\mathbf{p}(x)+\mathbf{p}(y)}}\right] =$$
$$\begin{cases} [A], & A \in [0.5, 1) \quad \text{for} \quad \mathbf{p}(x) \geq \mathbf{p}(y) \\ [A], & A \in (0, 0.5) \quad \text{for} \quad \mathbf{p}(x) < \mathbf{p}(y) \end{cases} =$$
$$\begin{cases} 1 & \mathbf{p}(x) \geq \mathbf{p}(y) \\ 0 & \mathbf{p}(x) < \mathbf{p}(y) \end{cases} = \tau(\mathbf{p}; x, y)$$

From this formulation, we can see that the binary test vector can be calculated with a fully-connected layer whose weight matrix $\mathbf{w}$ is built as follows. For all the binary tests $\tau(\mathbf{p}; x_i, y_i)$ indexed with $i, 1 \leq i \leq n_d$, we set the weights $w_{ix_i} = 1, w_{iy_i} = -1, w_{ij} = 0, j \neq x_i, j \neq y_i$. In other words, each neuron of the output of the fully-connected layer (i.e., each neuron of the binary code) is connected with exactly two neurons of the input of the fully-connected layer (two pixels on which the binary test will be applied), one with the weight $+1$, and the other with the weight -1, see Figure 5.2-a (bottom). The biases are set

to 0. What is left is to set the activation function of this fully-connected layer to sigmoid, and the output of the whole network will be the BRIEF descriptor of the input patch.

For Step 2, we use a decoder network consisting of a fully-connected layer (to mirror the network from Step 1), and several convolutional layers followed by upsampling until we reach the output of half of the size of the original patch. Learning to decode into a downsampled patch makes the autoencoder more resistant to Gaussian noise and, to some extent, to geometric noise. Step 3 involves training with artificial geometric noise added to the input patches while leaving the output patches the same. This should further increase descriptor's resilience to geometric noise such as rotation, translation, and shearing. Our results from the HPatches benchmark in the following section show that this step is quite effective in making our descriptor resistant to these kinds of geometric noise. The descriptor obtained from this step only differs in its weights from the BRIEF descriptor implemented as a CNN (from Step 1). In the final step, one would then apply further training with respect to a particular dataset, such as MRI images or multimodal macro photography. These applications, however, are beyond the scope of this thesis.

We set binary cross-entropy as the loss function $\mathcal{L}$. We use Adadelta optimiser for all neural networks in this chapter. The networks are trained on a dataset of 80k 56×56 patches that were extracted from the images from the Imagenet dataset using FAST (Features from Accelerated Segment Test) algorithm for feature detection [Rosten 06]. The ratio between training, validation, and test set is $8 : 1 : 1$. Our implementation is written in Keras and is publicly available. [1]

5.4 Experimental results

We evaluate the performance of our learned BRIEF in comparison with the original BRIEF descriptor, as implemented in the OpenCV library. We also evaluate the performance of a descriptor that uses the same architecture as learned BRIEF, but has only been trained once from scratch, without been initialised to mimic BRIEF and then fine-tuned. The evaluation is performed on HPatches [Balntas 17], a comprehensive benchmark for evaluating local image descriptors which was described in Section 3.2. We consider all tasks and difficulty levels provided in the benchmark, and we discuss the results in the following.

The patch retrieval task tests how well a descriptor can match a query patch to a pool of patches extracted from many images, including many distractors. Figure 5.3 shows the performance of three descriptors for different sizes of test dataset (i.e., different sizes of the pools of patches in which the matching patches are searched for), comparing performances on all three difficulty levels. Our learned BRIEF shows improvement over the original BRIEF for each difficulty and for each size of pool of patches. It also shows slightly (but consistently) higher mean average precision than the descriptor that has not

[1] https://github.com/nimpy/learned-brief

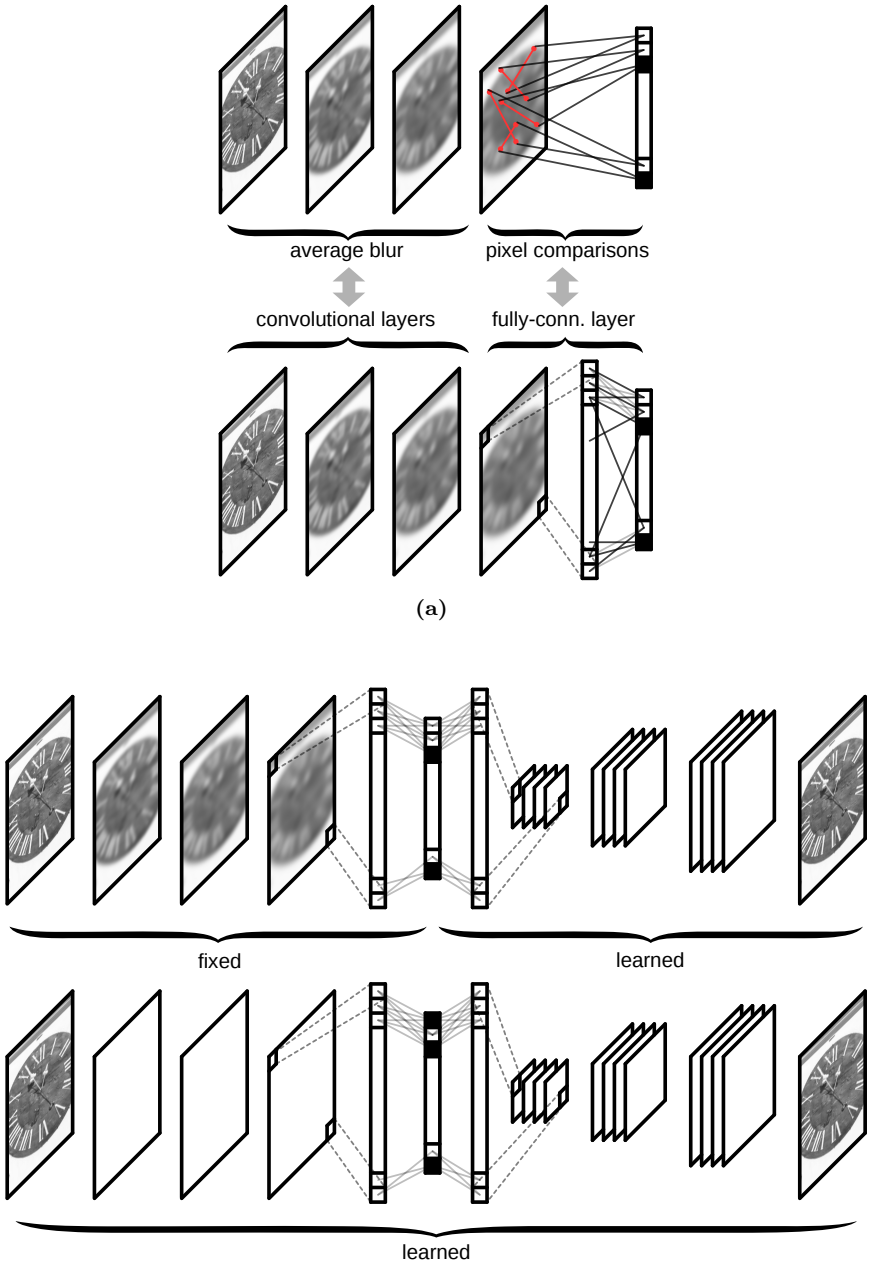


Figure 5.2: Visualisation of the proposed knowledge transfer and improvement framework from a BRIEF descriptor to an autoencoder-learned descriptor. (a) Step 1 – implementing BRIEF as a CNN. Original BRIEF implementation (top) and our proposed as a CNN (bottom). (b) Top: Step 2 – learning to invert a descriptor into its patch by training the decoder part of the autoencoder. The encoder is here set to be the network from step 1, implementing BRIEF. Bottom: Step 3 – training the whole autoencoder on a general dataset of patches.

been initialised to mimic BRIEF. Based on the results from the benchmark (as shown in Figure 5.3), we conclude that learned BRIEF outperforms the other two descriptors by a constant margin irrespective of the amount of geometric noise. The most influential factor is rather the size of the pool of patches – we observe the largest improvement on smaller pools that are typical in tasks such as panorama stitching, object tracking, and inpainting. Learned BRIEF remains superior to BRIEF also on large pools of patches and after a certain pool size (above ∼10000 patches) the improvement stabilises.

The image matching task tests to what extent a descriptor can correctly identify correspondences in two images based on a pair of patches – one patch from each of the images. Our results for this task are shown in Figure 5.4. Similar to the previous task, the amount of geometric noise is varied. The results show that learned BRIEF also outperforms BRIEF on the image matching task, no matter the amount of geometric noise. However, in this task we do not see much difference between learned BRIEF and learned descriptor without any initialisation. The descriptor that was trained from scratch even slightly outperformed learned BRIEF on the easy amount of noise but shows the same performance when there is more geometric noise present.

We show in figures 5.5 and 5.6 examples of retrieved patches, comparing learned BRIEF with BRIEF. These patches have been taken from a subset of the HPatches dataset. We compare the encoding of a query patch (showed larger in the figure), with the encodings of other patches, using both the original BRIEF descriptor and our learned BRIEF. We have observed that in most cases, the first patch retrieved by our descriptor is a much better match than the first patch retrieved by the original BRIEF. Furthermore, we have observed that the original implementation generally retrieves a visually dissimilar patch within the first five retrieved patches. This characteristic was not present in learned BRIEF.

5.5 Conclusions

In brief, we introduced in this chapter a framework for transferring knowledge from a hand-crafted descriptor to a self-supervised-learning–based descriptor. We demonstrated the use of this framework by creating the learned BRIEF descriptor based on the BRIEF hand-crafted descriptor. Furthermore, we proposed an elegant implementation of BRIEF as a convolutional neural network. Using HPatches benchmark for evaluating local image descriptors, we showed that our learned BRIEF descriptor outperforms consistently the original BRIEF.

This chapter acts as a proof of concept of our framework for knowledge transfer from hand-crafted descriptors, showing that a learned descriptor created in this way can outperform its hand-crafted counterpart. Regarding patch retrieval, the experiments showed consistently the benefits of initialising the descriptor as BRIEF and fine-tuning it further on a dataset of patches. On the patch matching, however, the descriptor that was learned from scratch showed equal performance to learned BRIEF. We can conclude that the experimental results do not support strongly using our framework for the tasks of patch

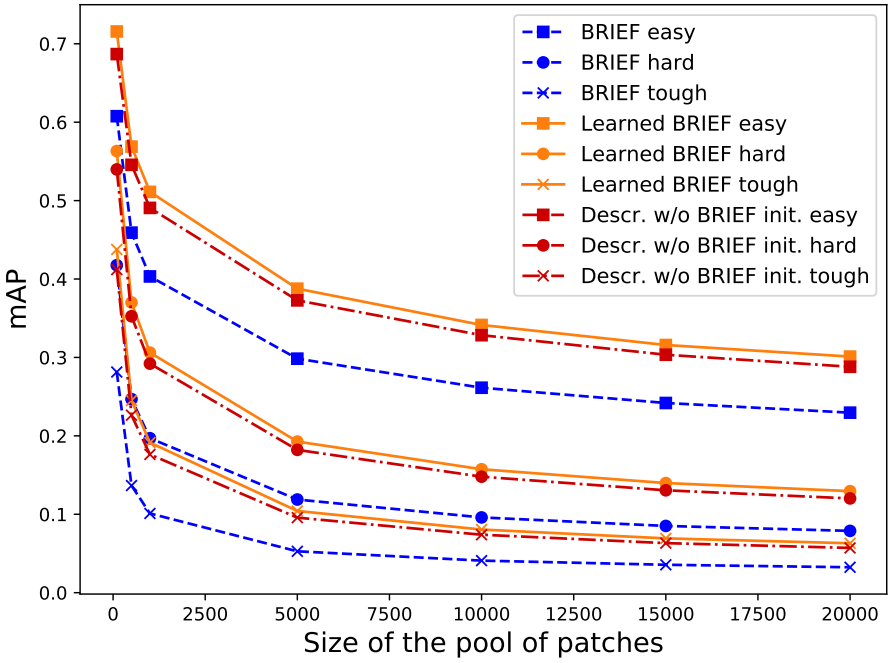


Figure 5.3: Performance comparison on the patch retrieval task between BRIEF, learned BRIEF, and the descriptor with same architecture as learned BRIEF but without BRIEF-mimicking initialisation.

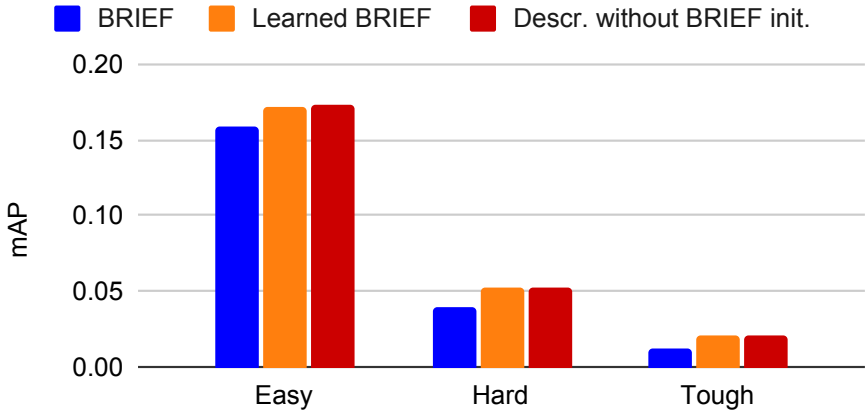


Figure 5.4: Performance comparison on the image matching task between BRIEF, learned BRIEF, and the descriptor with same architecture as learned BRIEF but without BRIEF-mimicking initialisation.

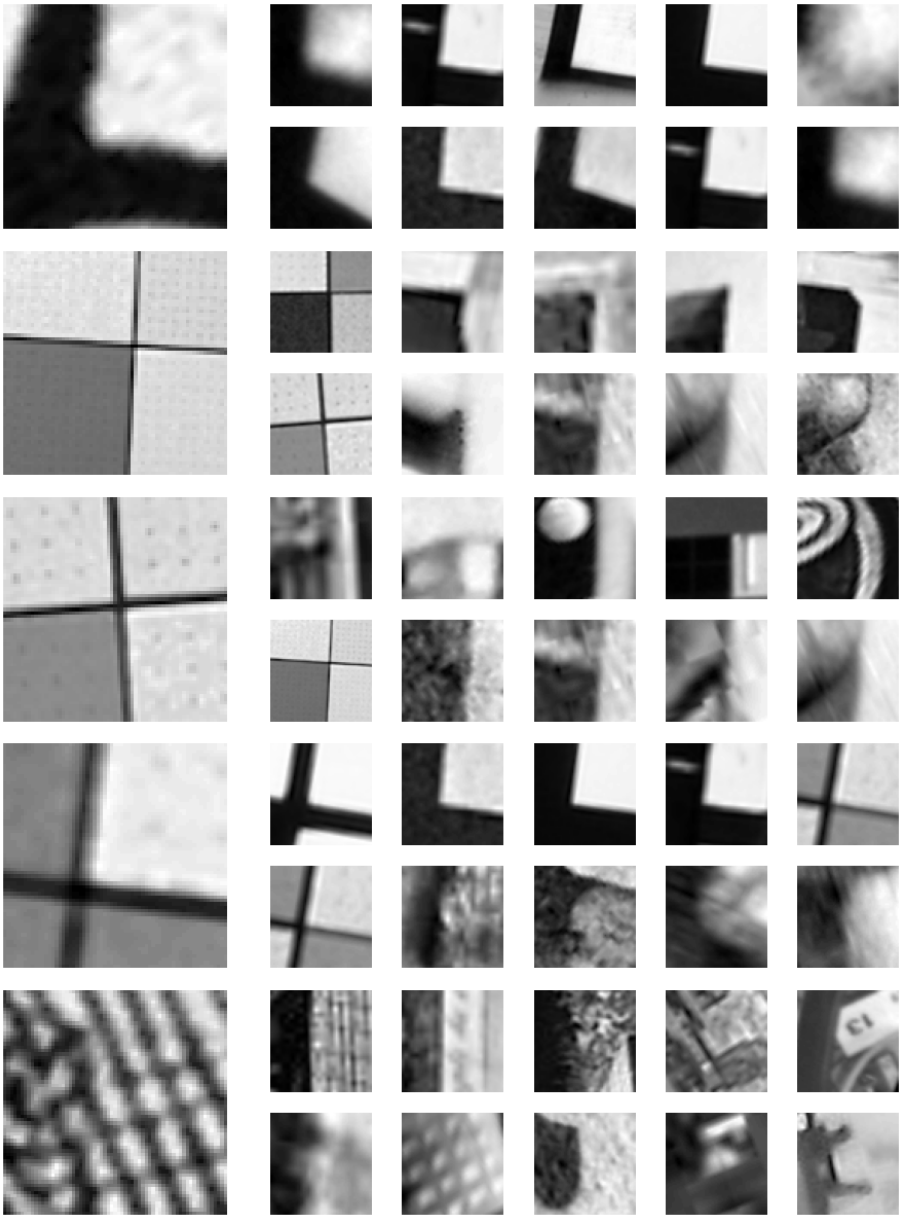


Figure 5.5: Patch retrieval examples. Large patch is the query patch. Top rows: original BRIEF descriptor; bottom rows: learned BRIEF.

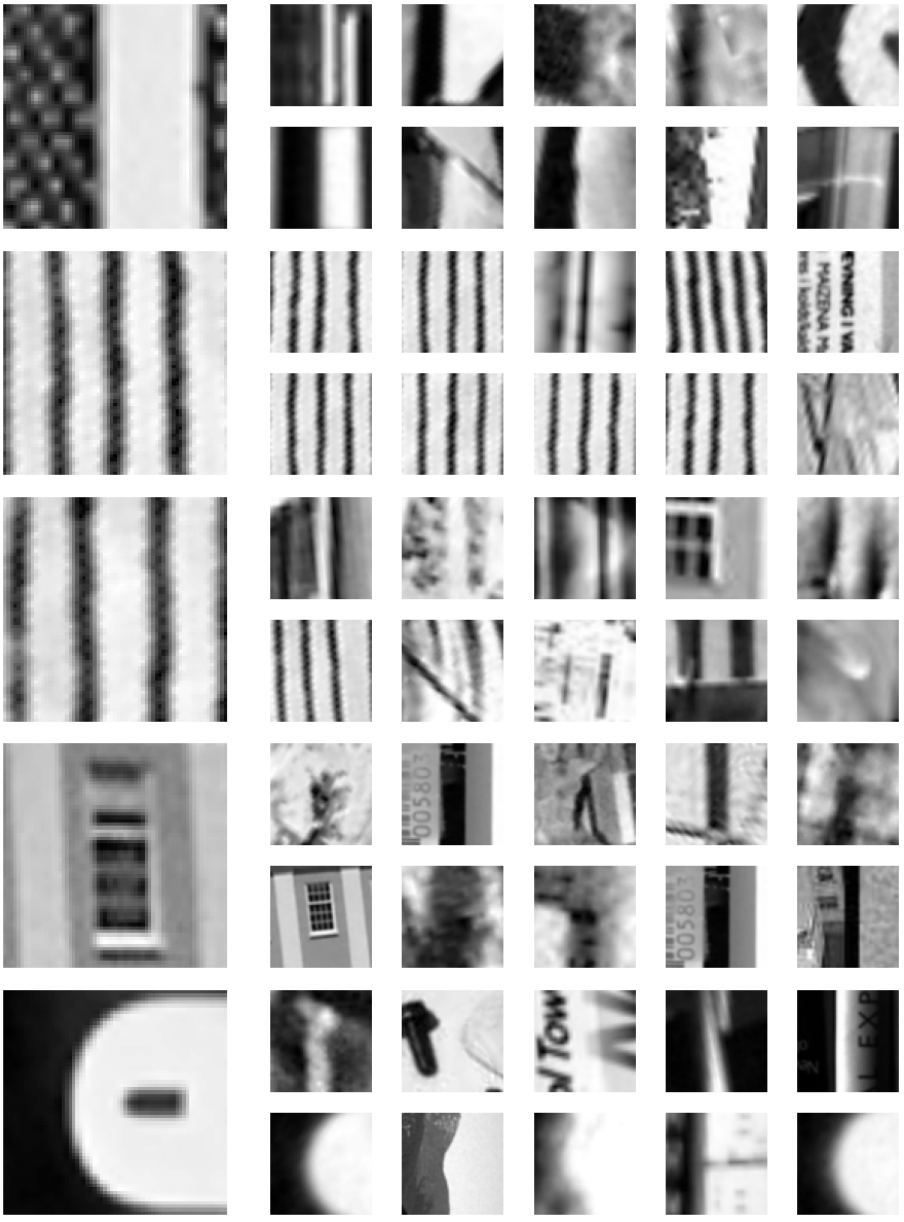


Figure 5.6: Patch retrieval examples. Large patch is the query patch. Top rows: original BRIEF descriptor; bottom rows: learned BRIEF.

matching, but certainly do for the task of patch retrieval.

The research in this chapter was reported in one conference paper [Žižakić 20a].

6

Deep image hashing with autoencoders

The battle of Helm's Deep is over; the battle for Middle Earth is about to begin.
—Gandalf, from *The Lord of the Rings: The Two Towers*

The focus in previous chapters was on local image descriptors – image dimensionality reduction methods for small parts of images (image patches) that contain simple structures. In this chapter, we explore image dimensionality methods for the whole images – image hashing methods. While the two categories of methods are similar, there are several important differences between them. With image hashing, the images to be encoded are larger and the encodings are smaller than with local image descriptors. Encodings for image hashing are often 16, 32, or 64 bits, as opposed to local image descriptors, whose encodings are, in general, not binary vectors. One implication of these differences is that, while local image descriptors can encode (nearly) all the information in patches, the idea of image hashing methods is to encode some general information about the image, e.g. whether it is an image of a dog. Related to this, the idea of image hashing methods is that all images that have the same high-level description (such as being an image of a dog) would be mapped to the same hash encoding. Conversely, the idea of local image descriptors is that the patches that are capturing the same *measurement* (region of a scene) would not necessarily have the same, but very similar encoding. That is to say, image hashing methods are less fine-grained. While local image descriptors encode patches into (often) unique encodings, image hashing methods perform dimensionality reduction more akin to clustering.

6.1 Introduction

Due to the growth of computer vision applications, there has been an ever-increasing supply of high-dimensional visual data. Thus, there is an increase in demand for accurate and efficient retrieval methods for these datasets. Traditional nearest neighbour search methods tend to struggle with high-dimensional data. Not only are these techniques impacted by the increasing computational

cost for calculating element-wise distances, they also fail to translate semantic information into these distances. A promising solution to this problem is presented in the field of image hashing. Here, data in high-dimensional space is mapped onto low-dimensional hash codes in such a way that distance between data points is conserved into those hash codes. This allows for the application of traditional methods on the converted low-dimensional data.

Hashing methods can be both data-independent and data-dependent (learning to hash). Data-independent hashing methods are based on hand-crafted features, and can thus only capture visual similarity between images, rather than a semantic similarity. Data-dependent methods, on the other hand, use deep learning to achieve semantic similarity and have shown state-of-the-art performance over classical data-independent methods [Dizaji 18, Shen 15, Carreira-Perpinán 15, Shen 20]. Data-dependent hashing techniques can be subdivided in two classes: those that require labelled data (supervised techniques) and those that do not (e.g. self-supervised or unsupervised techniques). While supervised techniques can use labels to increase retrieval accuracy significantly, they also depend on said labels. In many real-world applications, datasets may not have semantic annotations. Here we opt for self-supervised hashing techniques, which use the intrinsic data structure to determine meaningful hash codes.

Many self-supervised techniques are already established, examples of which are autoencoder networks [Carreira-Perpinán 15,En 17,Dai 17], adversarial networks [Dizaji 18,Zieba 18,Song 18] and graph-based networks [Weiss 09,Liu 11]. A recent work by Shen et al. [Shen 20] shows state-of-the-art results. The authors describe a hybrid autoencoder and graph-based network for hashing, called Twin Bottleneck Hashing (TBH). The architecture leverages a code-driven graph, allowing it to circumvent the static-graph problem which is inherent with precomputed graphs.

The main contribution of this chapter is a novel method for deep image hashing based on variational autoencoders. We leverage and optimise the architecture from twin-bottleneck hashing with recent insights from the field of variational autoencoders. In particular, we propose two separate improvements over TBH that are based on variational autoencoders. We experimentally show an overall increased performance on the CIFAR-10 (60k images) and MS-COCO (330k images) datasets for both improvements separately, with an even larger performance increase when the improvements are applied together.

The rest of the chapter is organised as follows. We provide an overview of state-of-the-art on hashing for content-based image retrieval in Section 6.2. In Section 6.3, we present our image hashing method which builds upon TBH by improving the binary bottleneck (Section 6.3.1) and expanding the continuous bottleneck (Section 6.3.2). In Section 6.4, we experimentally evaluate the two modifications, both separately and together, in all cases showing improvement over the state of the art. We conclude our work in Section 6.6.

6.2 Hashing for content-based image retrieval – an overview

Similarly to the local image descriptors (as we have seen in Section 3.1), hashing techniques can be also categorised into data-independent (hand-crafted) and data-dependent methods (learning to hash). Data-independent methods tend to rely on hand-crafted features to extract hash codes, without using the data distribution. These methods are often referred to as locality sensitive hashing (LSH) methods, named after a family of hash functions that maps similar inputs to the same hash code. Two works pioneered this technique [Charikar 02, Andoni 06]. There are a lot of variations using different distance metrics [Indyk 98, Dasgupta 11, Kulis 11, Ji 12] or changing the search method [Panigrahy 05, Bawa 05, Pan 12].

Data-dependent methods may not optimally exploit the full data distribution. These methods often fail to capture semantic similarities between data points [Wang 17]. Data-driven methods are showing much more promising results in the field of image retrieval [Luo 20]. Most of the work in this field has focused so far on supervised techniques, which exploit the data distribution together with its annotations. Over recent years, many different architectures have been proposed for supervised learning to hash. Perhaps the most straightforward techniques make use of a deep encoder to directly encode inputs to hash codes [Gong 13, Erin Liong 15, Shen 15, Wang 16b]. Yang et al. presented a fine-tuning process to convert such an encoder of a classifier into a deep hashing network [Yang 17]. Another popular supervised technique trains on predicting a pairwise loss function [Lin 13, Xia 14, Liu 16, Shi 16]. A natural extension to this are triplet-based losses, where a query data point is compared to both a similar and a dissimilar data point [Wang 16a]. Supervised generative adversarial networks have proven to be viable hashing methods [Qiu 17, Cao 18, Wang 18], as well as supervised autoencoder-based networks that exploit the models' bottleneck to extract hash codes [Cao 16, Dadaneh 20].

While supervised data-dependent methods show promising retrieval results, they require annotated datasets. This is a severe limitation, since, for many real-world datasets, it is not feasible or it is simply too costly to make annotations. Therefore, there is a demand for methods that can learn useful hash functions solely based on data. There is a wide variety in the type of self-supervised methods. Generative adversarial networks again show state-of-the-art performance in this field [Zieba 18, Dizaji 18, Song 18, Cao 18]. Similarly, autoencoders also remain a relevant technique [Carreira-Perpinán 15, En 17, Hansen 20]. Dai et al. build on the idea of a variational autoencoder, applying it to construct hash codes directly from the bottleneck [Dai 17]. Hu et al. propose a self-supervised technique which assigns pseudo labels to the data using precomputed features and shows promising results [Hu 17]. The approach optimises its hash function to maximally compress the dataset and is a generative approach since it can be used to regenerate the inputs. A recent work by Shen et al. [Shen 20] introduced a method called twin-bottleneck hashing (TBH) that uses autoencoder-type architecture which implements elements from graph-based learning. This architecture inspired our work and thus we describe it

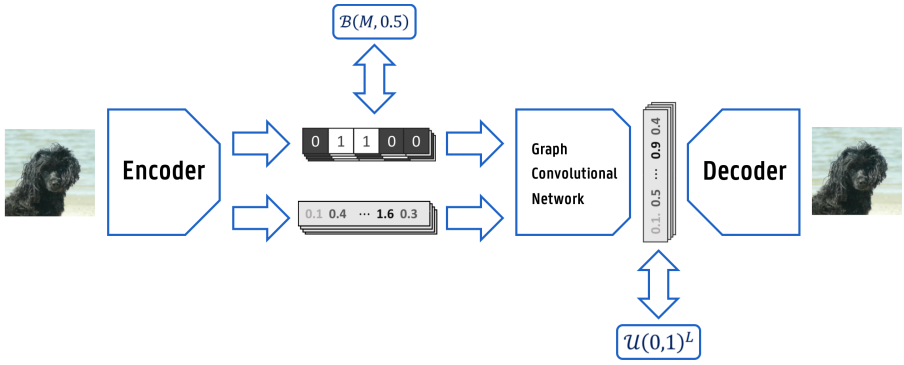


Figure 6.1: Schematic of the twin-bottleneck hashing (TBH) method [Shen 20].

in more detail in following section. While TBH provides state-of-the-art self-supervised hashing, we show that, by improving certain components of the architecture, we can improve its performance even further.

Twin-Bottleneck Hashing

Twin-bottleneck hashing (TBH) [Shen 20] is a deep image hashing method with two autoencoder-like bottlenecks connected to a graph convolutional network (architecture shown in Figure 6.1). We explain here briefly its main components.

Twin bottlenecks. The most notable feature of TBH is the use of two separate bottlenecks instead of one. First, a binary bottleneck, which is responsible for generating the actual hash codes. Inspired by Dai et al., a stochastic neuron is used to generate the hash codes [Dai 17]. An intrinsic problem that arises is that we want binary codes to remain as small as possible, this limits the flexibility of the model. To alleviate this problem, a second larger continuous bottleneck allows for the model to capture much more complex representations.

Graph convolutional layer. The graph convolutional layer creates a pathway to calculate the gradients for both bottlenecks. It uses the binary variables to create a similarity graph, which is used to create the convolutional filter along with a set of trainable weights. Intuitively, the graph convolutional layer will penalise unrelated samples that are close together in Hamming space. One of the major contributions of TBH is that the similarity graph is now constructed and optimised during training, rather than built on precomputed features.

Regularisation. TBH uses two discriminators as a means of regularisation. One discriminates hash codes from random binary codes. The other one discriminates the output of the graph convolutional layer with uniformly distributed samples between zero and one.

6.3 Variational Autoencoder Twin-Bottleneck Hashing

Here we propose a self-supervised hashing approach, which builds on the twin-bottleneck hashing (TBH) model and improves upon it. Specifically we improve two main aspects of the TBH approach. First, we change the way binary codes are generated. This way we directly influence the ability of the model to learn representations. Second, we design a more powerful generative bottleneck. This improves the model as a whole, and thus also the retrieval scores. Our overall objective function consists of three components: a regularisation term on the binary bottleneck L_{BBN}, a constrained optimisation setup between the reconstruction loss and the regularisation on the continuous bottleneck L_{CBN} and lastly a discriminator term on the output of the graph convolutional network L_D analogous to TBH's approach. This results in the following objective:

$$L_{OBJ} = L_{BBN} + L_{CBN} + L_D \,, \tag{6.1}$$

with L_{BBN} and L_{CBN} explained in more detail in the following sections.

6.3.1 Improving the binary bottleneck

We focus first on the generation of the binary hash codes. Instead of generating hash codes with a stochastic neuron [Dai 17] like in TBH, we use a variational autoencoder scheme. In recent years, variational-autoencoder–based methods have shown improvements over the state of the art in image generation, classification and particularly representation learning. This motivates us to explore their potential for improved image hashing too. Learning a better representation in the bottleneck will enable us to capture more information from the input images to the hash codes and thus achieve better retrieval accuracies.

We thus design the binary bottleneck, which generates hash codes, as a variational autoencoder with stochastic sampling. We intend to optimise the hash codes to have statistically independent bits. This relates to a common topic in variational autoencoders where the latent distribution in the bottleneck is optimised to be disentangled. Chen et al. [Chen 18] stated that Kullback-Leibler (KL) divergence term in the ELBO objective for variational autoencoders can be decomposed in three terms: Index-Code mutual information, total correlation and dimension-wise KL divergence. They claimed that penalising the total correlation term by an extra factor would be responsible for a disentangled latent distribution. This results in latent variables that map to more explainable features like colour, shape etc., but it sacrifices some quality of the models' reconstructions.

We apply these principles to our hash code generation to measure if disentangled representations would benefit our models' performance. We formulate the loss on the binary bottleneck L_{BBN} as in a standard variational autoencoder with the decomposed KL term.

$$L_{BBN} = KL[q(z,n)||q(z)p(n)] +$$

$$\beta * KL[q(z)||\prod_j q(z_j)] + \sum_j KL[q(z_j)||p(z_j)]\,, \quad (6.2)$$

where β is a new hyper-parameter to penalise the total correlation term, x and z are the data and latent variable respectively, and n is a uniform random variable on $\{1, 2, ..., N\}$ which indexes the data points. Our encoder is defined as $q(z|n) = q(z|x_n)$ and the data's prior distribution $p(n) = 1/N$. From these we can derive $q(z, n) = q(z|n)p(n) = q(z|n)\frac{1}{N}$ and the aggregated posterior $q(z) = \sum_{n=1}^{N} q(z|n)p(n)$. Following [Chen 18], we express $KL[q(z, n)||q(z)p(n)]$ as the index-code mutual information. This is the mutual information between the data variable n and the latent variable z on the empirical data distribution $q(z, n)$. The second term, $\beta * KL[q(z)||\prod_j q(z_j)]$, is referred to as the total correlation. The penalty on this term forces the model to find statistically independent factors in the data distribution. The last term, $\sum_j KL[q(z_j)||p(z_j)]$, is the dimension-wise KL divergence, which encourages individual latent dimensions to represent their corresponding prior.

We expect that hashing with disentangled latent variables will force bits to allocate to clear, explainable features, allowing our model to preserve semantic similarity between samples. Note that these extra constraints on the bottleneck could also worsen the models' ability to fit to the input data. Experimentally however, we see a slight improvement in the precision of the architecture as will be shown in Section 6.4. In addition, we omit the discriminator on the binary bottleneck, as our implementation now uses KL divergence to regularise the binary bottleneck.

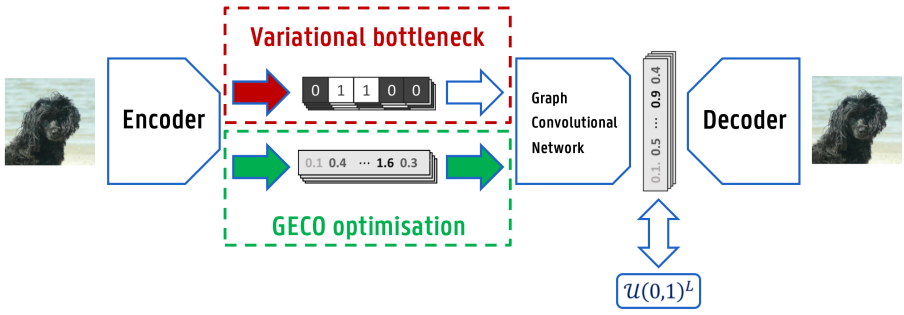


Figure 6.2: Schematic of the proposed variational autoencoder twin-bottleneck hashing (VAE-TBH) method, showing the improvements made to the original TBH method. In the binary bottleneck (top bottleneck) we change the generation of hash codes to be based on variational autoencoder with disentangled variables and we omit the regulariser (Section 6.3.1). We also change the continuous bottleneck (bottom bottleneck) to use a variational autoencoder that is trained using a constrained optimisation setup, in order to better control the trade-off between compression and reconstruction quality of generated samples (Section 6.3.2).

6.3.2 Expanding the continuous bottleneck

Now we turn to the design of the continuous bottleneck. Our idea is to convert the more standard autoencoder structure to a variational-autoencoder–type structure. This expanded architecture should allow the model to better capture input data, specifically boosting the shared encoder and thus achieve better results when generating hash codes. Rezende et al. [Rezende 18] propose a robust algorithm for optimising variational autoencoders. We apply their method on the continuous bottleneck and redefine accordingly the ELBO objective into a constraint optimisation problem:

$$L_{CBN} = \mathbb{E}_{p(x)}[KL[q(z|x);\pi(z)]] + \lambda^T \mathbb{E}_{p(x)q(z|x)}[C(x,g(z))]\,, \quad (6.3)$$

$$\text{with } C(x,g(z)) = \|x - g(z)\|^2 - \kappa\,, \quad (6.4)$$

where we balance compression $\mathbb{E}_{p(x)}[KL[q(z|x);\pi(z)]]$ on the bottleneck with reconstruction quality of our generated samples $\mathbb{E}_{p(x)q(z|x)}[C(x,g(z))]$. For the compression term, we use the KL divergence between the encoder's generated Gaussian distributions $q(z|x)$ and a set of normalised Gaussian priors $\pi(z)$. We use L2-loss, $\|x - g(z)\|^2$, to represent our reconstruction loss. This allows for an elegant trade-off where we can choose to invest a bit more in compression as we do not care much about our generated samples. A new hyper-parameter κ is set to achieve the desired reconstruction quality, which controls the value of λ^T, as is described in the optimisation scheme proposed by Rezende et al. [Rezende 18].

We expect as the model improves through the continuous bottleneck, that the shared encoder and decoder will improve. This benefits the binary bottleneck directly and should result in better retrieval scores for our model. In the following section, we will show that this indeed improves the performance of the model.

6.4 Experimental results

We evaluate our method in comparison with the related state-of-the-art methods in the field of self-supervised hashing. We perform experiments on both CIFAR-10 and MS-COCO dataset (examples of images from these datasets shown in Figures 6.3 and 6.4). Additionally, we extract features from the fc_7 layer using AlexNet [Krizhevsky 09]. This is the same extraction method as described in TBH. We train and evaluate our models in TensorFlow. For the baseline method TBH, we experiment with the same setup and hyper-parameters as stated in their paper. As for our method, we use the following hyper-parameters. TBH hyper-parameters are kept the same, with $\lambda = 1$ and $L = 512$. We train in batches with batch size $= 1024$ using learning rate $= 1e-4$. For the GECO implementation we use $\kappa = 2400$, and clip the λ parameter between $1e-6$ and $1e12$ to avoid extreme values during training.

Figure 6.5 shows precision-recall curves for our method, the TBH baseline and each of the two improvements from sections 6.3.1 and 6.3.2 separately on

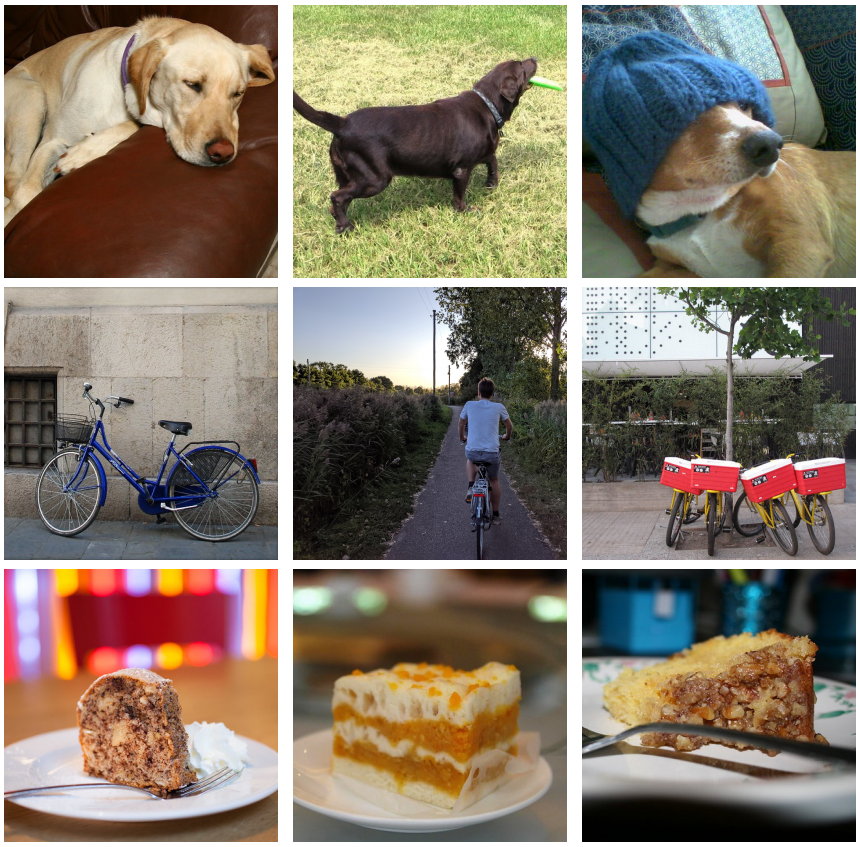


Figure 6.3: Example images from MS-COCO dataset

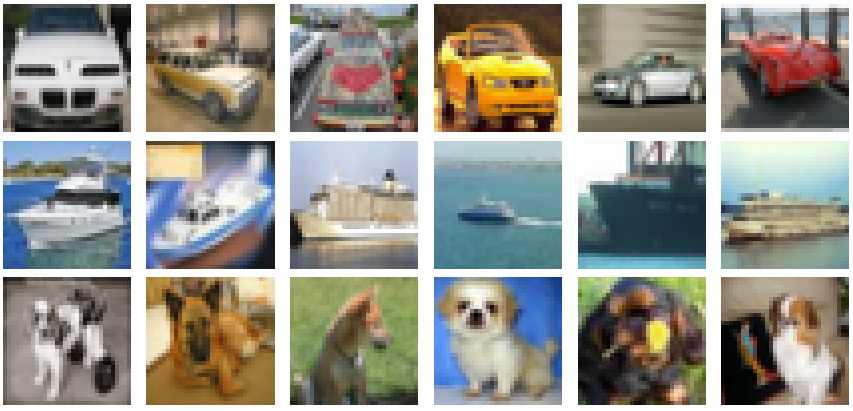


Figure 6.4: Example images from CIFAR-10 dataset

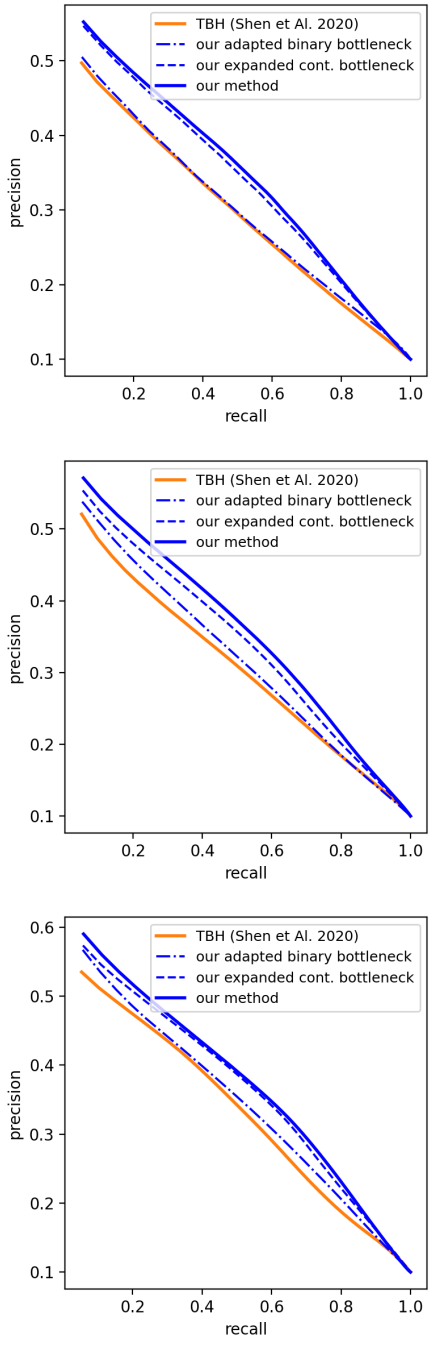


Figure 6.5: Comparison of precision-recall curves on the CIFAR-10 dataset for TBH, our improvements from sections 6.3.1 and 6.3.2 separately and together. We use 16-bit, 32-bit and 64-bit codes (top to bottom respectively) to evaluate these models.

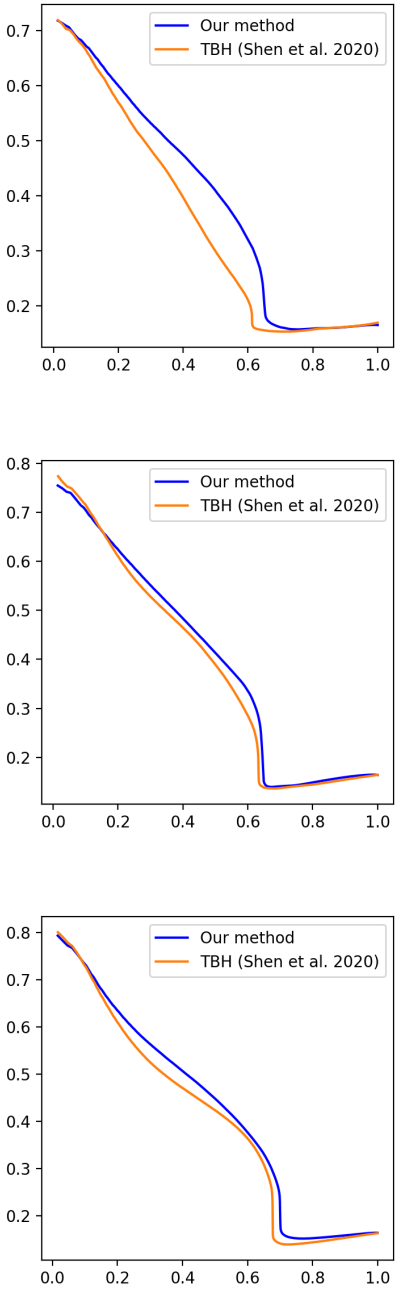


Figure 6.6: Comparison of precision-recall curves on the MS-COCO dataset for TBH and our method. We use 16-bit, 32-bit and 64-bit codes (top to bottom respectively) to evaluate the models.

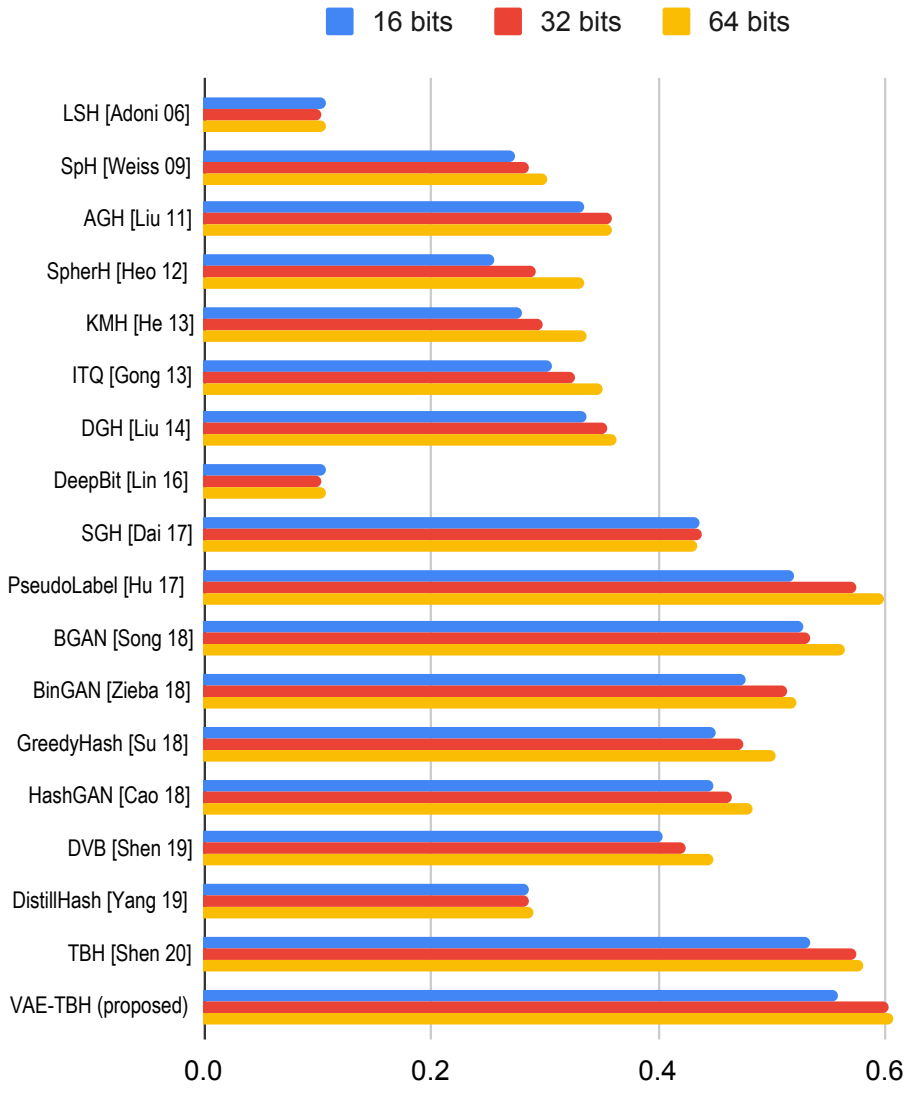


Figure 6.7: mAP of our method and state-of-the-art self-supervised hashing methods on the CIFAR-10 dataset.

Figure 6.8: Retrieval examples for both TBH (bottom rows) [Shen 20] and our method (top rows), the correct classes are labelled green.

CIFAR-10 dataset. We see that our method improves precision scores across the board for 16, 32 and 64 bits codes. This confirms that the changes we make to the bottlenecks of TBH do in fact improve the generated hash codes and thus the retrieval scores. Looking at the components separately, we see that expanding the continuous bottleneck contributes to the majority of our improvement. Changing the binary bottleneck has a less significant effect.

Figure 6.6 shows precision-recall curves for our method and TBH baseline on MS-COCO dataset. On this dataset, the curves for both methods look different in comparison to those on the CIFAR-10 dataset. This suggests that there is a group of data samples that cause confusion and retrieving later samples starts to increase precision. Our method outperforms TBH on this dataset as well.

Table 6.7 shows a comparison between more state-of-the-art methods on CIFAR-10 dataset. To evaluate methods, we use mean average precision (mAP) at 1000. This measures the average relevance scores of a set of the top-1000 images in response to a query. Our method shows improved results over multiple baselines.

To conclude the results, we show some example retrievals from both our method and the TBH baseline in Figure 6.8. The query examples are randomly selected from the test set and retrievals are returned from our train set. On such a small set of queries, it is difficult to clearly see our improvements – there are examples of images where our method performs slightly better than TBH, and vice versa. For the most part, however, we observe that queries where our method struggles are the same ones that are difficult for the TBH method, which is to be expected since these are more difficult queries and since these two methods share similarities.

6.5 Search of self-similar structures in electron microscopy images

In this section, we illustrate the use of our proposed image dimensionality reduction methods in the application of bio-medical imaging. Our use-case is content retrieval in electron microscopy (EM) datasets. The idea of this use-case is to help medical doctors or biologists search for similar structures in large datasets. If they, for example, notice an anomaly in the tissue, it would be beneficial to locate other such anomalies, searching across different datasets they have access to.

We are using image dimensionality reduction methods to encode the patches and thus accelerate the search process. We encode all the patches (for a given stride) using our proposed image hashing method, and compare these patch encodings with the encoding of the query patch, retrieving the ones that are the most similar.

In Figure 6.9, we show the results of applying the developed hashing method to content retrieval in different EM datasets. We present the results where the best matches are retrieved for a given query patch. The retrieved patches share similarities in the semantic sense with the query (all the retrieved patches contain parts of the same type of cell organelle (mitochondria), which appears

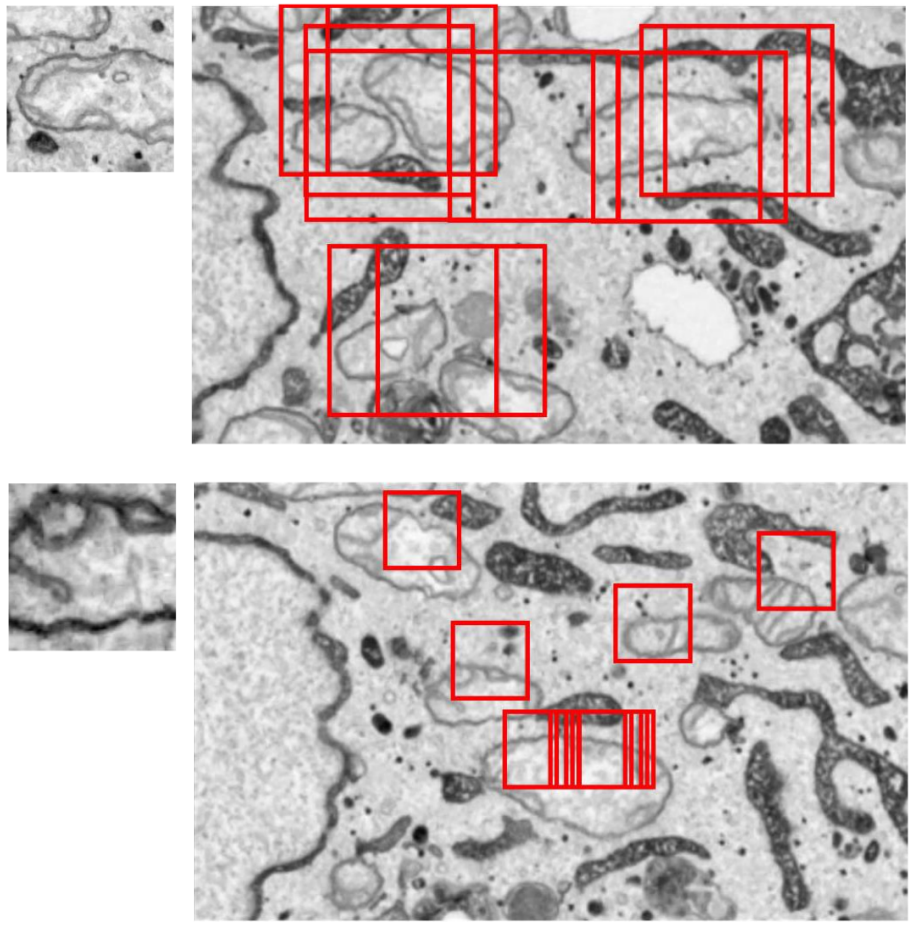


Figure 6.9: Content-based patch retrieval. Query patch is shown on the left, and retrieved patches are marked with red. Both the query patch and the retrieved patches contain mitochondria.

in the query.

We believe that this method can be useful to locate potentially interesting areas in large datasets (e.g., as input to further verification and precise localisation of the objects of interest). This method can also be employed to semi-automatically label large datasets based on a relatively few labelled patches.

6.6 Conclusion

In this chapter, we proposed a novel method for self-supervised deep image hashing based on twin-bottleneck hashing and variational autoencoders, that we call VAE-TBH. We improved the performance of the state-of-the-art TBH approach by designing the bottleneck structure inspired by some recent insights in the field of variational autoencoders. We adapted the generation of hash codes to promote learning disentangled representations and we expanded the continuous bottleneck to a variational autoencoder to improve the models' ability to fit to input data. The experimental evaluation on different datasets and different hash-code sizes showed that our method improves upon the state of the art in all cases. The research in this chapter has led to one conference paper [Verwilst 21].

7

Conclusions and future work

Well, here at last, dear friends, on the shores of the Sea comes the end of our fellowship in Middle-earth. Go in peace! I will not say: do not weep; for not all tears are an evil.
—Gandalf, from *The Lord of the Rings: The Return of the King*

7.1 Conclusions

It is estimated that around 45 thousand images are captured every second, resulting in around 1.5 trillion images per year [RAR 22]. With the right image processing, this data could be highly valuable and can be used to solve complicated problems such as teaching machines to recognise and track objects, or retrieving the most relevant data related to an image. For this reason, there is a significant need to extract the most important information from images and to represent those images in lower-dimensional space. This need has, in turn, lead to an increase in the demand for effective image dimensionality reduction methods. In this thesis, we have focused on two important types of image dimensionality reduction: local image descriptors and image hashing.

Local image descriptors play a crucial role in many image processing tasks, such as object tracking and recognition, panorama image stitching, and image retrieval. They allow us to match different image patches based on their similarity. In chapters 3-5, we research learning local image descriptors in a self-supervised fashion, using autoencoders.

Image hashing is an fundamental tool for content-based image retrieval, where the most similar images are retrieved for a given query image. In Chapter 6, we develop a state-of-the-art self-supervised deep image hashing method using variational autoencoders.

Over the course of this thesis, we have made several contributions to the fields of learning local image descriptors and of image hashing, which are summarised in the following paragraphs. At the end of this section, we present a critical reflection of the work performed in this thesis.

Deeper understanding of autoencoders for learning local image

descriptors. While conducting the research in this thesis, it came to our attention that there were several gaps in the literature regarding how to effectively train and evaluate both classical and variational autoencoders for learning local image descriptors. To address these knowledge gaps, in Chapter 3 we performed a thorough comparative analysis of these two types of autoencoders alongside an in-depth analysis of the most relevant hyperparameters. Our research in this area provided insights into how to select an appropriate evaluation technique during supervised learning of local image descriptors and led to the development of a fast and efficient evaluation technique that is correlated with established benchmarks and fast enough to be used during training. Furthermore, we studied the invertibility property of local image descriptors and proposed a descriptor designed specifically for being invertible.

Local image descriptors designed for memory-efficient storing of patch encodings. Some image processing tasks involve performing a large number of patch comparisons within an image. To perform these comparisons, the descriptors of these patches must either be calculated on-the-fly or cached in memory, with the former entailing a significant computation load and the latter requiring a significant amount of memory. In Chapter 4, we proposed an autoencoder architecture for learning descriptors designed for enabling memory-efficient storing of patch encodings. Our autoencoder architecture created a special, intermediate representation of the image which served as a compact means of storing the descriptors of the patches. Storing the patches in this way is both computationally and memory efficient since the patches are overlapping in the intermediate representation and can be extracted using a single max-pooling operation. We evaluated this descriptor on the HPatches benchmark and observed significant memory savings without any degradation in performance. Furthermore, we conducted a qualitative evaluation by integrating our descriptor into an inpainting algorithm and verified that it performs as expected when applied to the virtual restoration of master paintings.

Framework for the transfer of knowledge from hand-crafted to self-supervised-learning–based descriptors. In Chapter 5, we developed a framework for transferring knowledge from hand-crafted local image descriptors to learned descriptors. This framework was motivated by the following observations. First and foremost, learned-descriptors often fail to consider the broader image processing task and are not properly customised to the needs of specific applications. Secondly, most learning-based approaches are supervised methods which require labelled datasets. These datasets, however, are often costly and hard to come by. Finally, hand-crafted methods are more interpretable, and therefore usually preferred in high-stakes applications. The framework presented in this thesis, however, offers a solution to the shortcomings of learning-based approaches by enabling the transfer of knowledge from hand-crafted descriptors to learning-based descriptors. When using this framework, there was no need for labelled datasets and the resulting descriptor was both more explainable and tunable to image processing applications. We put our proposed framework to the test by creating the learned BRIEF descriptor by transferring knowledge from the BRIEF hand-crafted descriptor [Calonder 10]. To that end, we proposed an elegant implementation of BRIEF as a

convolutional neural network.

Deep image hashing based on twin-bottleneck hashing with variational autoencoders. Deep image hashing is a technique for mapping images onto a lower-dimensional binary codes (hash codes) to enable content-based image retrieval. In Chapter 6, we proposed a novel deep hashing method called twin-bottleneck hashing with variational autoencoders – VAE-TBH. Our method employed variational autoencoders in both bottlenecks of the original twin-bottleneck hashing approach. In the binary bottleneck we encouraged the learning of disentangled variables by changing the generation of hash codes to be based on a variational autoencoder. In the continuous bottleneck we employed a variational autoencoder trained using a constrained optimisation setup, which enabled better control over the trade-off between compression and reconstruction quality of generated samples. Both of the modified bottlenecks resulted in improved hashing performance separately, and an even better result when applied together. We rigorously validated our method in comparison with state-of-the-art self-supervised deep hashing methods on different datasets (CIFAR and MS-COCO) and on different hash sizes (16-bit, 32-bit and 64-bit) – our method showed the best performance across all datasets and hash sizes.

Application to efficient search of self-similar structures in electron microscopy images. Our developed image dimensionality reduction methods can be utilised to search for regions of images that contain biological structures similar to the structure on the query image patch. We demonstrated this application in Chapter 6 by using image dimensionality reduction to encode the patches and accelerate the search process. The first steps that have been taken for this application show promising results.

Critical reflection on the work. In this thesis, we have proposed several novel methods for image dimensionality reduction and contributed to the knowledge about the autoencoder-based techniques in this field. Here we present a critical reflection on this work. Firstly, interesting avenues have been opened for valorisation (specifically in the domains of virtual art restoration and in biomedicine), but they have not yet been explored sufficiently. The valorisation in the biomedical domain (search for self-similar biological structures in EM images) is a topic of the ongoing work within the scope of the Flanders AI Research Initiative, where further experiments are ongoing at this point within our research group. Secondly, it needs to be admitted that some of the results presented in this work lack statistical significance testing. We inherited this limitation from the HPatches benchmark, which, despite evaluating descriptors on around 900k patches, does not provide standard deviation for its results. Furthermore, related literature in the field of deep learning also adopts this way of presenting the results, most likely due to the long training times of neural networks. In hindsight, if we were to perform this work again, we would include statistical significance testing for our experiments. Lastly, there are many research directions that we wanted to explore but did not have the time or resources to do it. In the next section, we outline some of the possibilities for the future work.

7.2 Future work

Insights into training and evaluation of autoencoders in general. In Chapter 3 we provided valuable insights about the training and evaluation autoencoders for learning local image descriptors. It would be very interesting to provide such insights for autoencoders used for other applications, and find out whether the same conclusions hold in general. Specifically, the research questions of interest would be: What hyperparameters are the most important for training autoencoders (and variational autoencoders) in general? How can we efficiently evaluate AEs and VAEs? Both questions could be answered by using triplet distance on labelled datasets (e.g. ImageNet) as an 'oracle benchmark' (similarly to how we used HPatches benchmark in this thesis), and then comparing different hyperparameter settings and different approximate evaluation metrics to the score returned by the oracle.

Comparison between other types of autoencoders. While we compare classical and variational autoencoders for the task of learning local image descriptors, it would be interesting to see how they compare to other types of autoencoders, especially some recent ones such as adversarial AEs [Makhzani 15], Wasserstein AEs [Tolstikhin 17] or adversarial latent AEs [Pidhorskyi 20]. To the best of our knowledge, such a comparison does not exist, and we intend on incorporating it into our future work.

Using our proposed knowledge-transfer framework on SIFT descriptor (or other famous local image descriptors). In Chapter 5 we proposed a framework for transferring the knowledge from hand-crafted to self-supervised-learning–based descriptors. We have utilised this framework to create *learned BRIEF* descriptor based on the BRIEF hand-crafted descriptor [Calonder 10]. While BRIEF is arguably the most important binary hand-crafted local image descriptor, other (non-binary) descriptors such as SIFT [Lowe 99], HOG [Dalal 05], SURF [Bay 08] or DAISY [Tola 09] have been used more extensively in the literature (and in practice), and it would be interesting to implement the learned versions of these using our proposed framework.

Further improvements on our proposed VAE-TBH deep image hashing method. In Chapter 6, we proposed a novel deep image hashing method based on twin-bottleneck hashing with two important improvements in both bottlenecks. We observed a significant boost in performance by expanding the continuous bottleneck to employ a variational autoencoder trained using a constrained optimisation setup. However, any architecture that can effectively help train this shared encoder could achieve improvements in the same way. Specifically, it would be interesting to see the performance of other hybrid networks, where the shared encoder would be trained by a general adversarial network (GAN) or another deep encoder architecture. The second aspect that could lead to further potential performance boosts is the dataset's feature extraction method. We have used the pretrained AlexNet [Krizhevsky 09] to extract features from a dataset. In recent years, many more advanced pretrained networks have become available (such as ResNet [He 16] and EfficientNet [Tan 19]) and could potentially boost the performance of our method.

Bibliography

[Aanæs 12] Henrik Aanæs, Anders Lindbjerg Dahl & Kim Steenstrup Pedersen. *Interesting interest points.* International Journal of Computer Vision, vol. 97, no. 1, pages 18–35, 2012.

[Abdel-Hamid 14] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn & Dong Yu. *Convolutional Neural Networks for Speech Recognition.* IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 22, no. 10, pages 1533–1545, 2014.

[Alahi 12] Alexandre Alahi, Raphael Ortiz & Pierre Vandergheynst. *FREAK: Fast Retina Keypoint.* In 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 510–517. IEEE, 2012.

[Alain 14] Guillaume Alain & Yoshua Bengio. *What regularized auto-encoders learn from the data-generating distribution.* The Journal of Machine Learning Research, vol. 15, no. 1, pages 3563–3593, 2014.

[Andoni 06] Alexandr Andoni & Piotr Indyk. *Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions.* In 2006 47th annual IEEE symposium on foundations of computer science (FOCS'06), pages 459–468. IEEE, 2006.

[Arandjelović 12] Relja Arandjelović & Andrew Zisserman. *Three things everyone should know to improve object retrieval.* In 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2911–2918. IEEE, 2012.

[Arganda-Carreras 15] Ignacio Arganda-Carreras, Srinivas C Turaga, Daniel R Berger, Dan Cireşan, Alessandro Giusti, Luca M Gambardella, Jürgen Schmidhuber, Dmitry Laptev, Sarvesh Dwivedi, Joachim M Buhmann*et al.* *Crowdsourcing the creation of image segmentation algorithms for connectomics.* Frontiers in neuroanatomy, page 142, 2015.

[Balntas 16] Vassileios Balntas, Edgar Riba, Daniel Ponsa & Krystian Mikolajczyk. *Learning local feature descriptors with triplets and shallow convolutional neural networks*. In Proceedings of the British Machine Vision Conference (BMVC), 2016.

[Balntas 17] Vassileios Balntas, Karel Lenc, Andrea Vedaldi & Krystian Mikolajczyk. *HPatches: A benchmark and evaluation of handcrafted and learned local descriptors*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5173–5182, 2017.

[Bawa 05] Mayank Bawa, Tyson Condie & Prasanna Ganesan. *LSH forest: self-tuning indexes for similarity search*. In Proceedings of the 14th international conference on World Wide Web, pages 651–660, 2005.

[Bay 08] Herbert Bay, Andreas Ess, Tinne Tuytelaars & Luc Van Gool. *Speeded-up robust features (SURF)*. Computer Vision and Image Understanding, vol. 110, no. 3, pages 346–359, 2008.

[Bayard 19] David S Bayard, Dylan T Conway, Roland Brockers, Jeff H Delaune, Larry H Matthies, Håvard F Grip, Gene B Merewether, Travis L Brown & Alejandro M San Martin. *Vision-Based Navigation for the NASA Mars Helicopter*. In AIAA Scitech 2019 Forum, page 1411, 2019.

[Belkin 03] Mikhail Belkin & Partha Niyogi. *Laplacian eigenmaps for dimensionality reduction and data representation*. Neural Computation, vol. 15, no. 6, pages 1373–1396, 2003.

[Bengio 13] Yoshua Bengio, Li Yao, Guillaume Alain & Pascal Vincent. *Generalized Denoising Auto-Encoders as Generative Models*. In NIPS, 2013.

[Biswas 21] Koushik Biswas, Sandeep Kumar, Shilpak Banerjee & Ashish Kumar Pandey. *SAU: Smooth activation function using convolution with approximate identities*. arXiv preprint arXiv:2109.13210, 2021.

[Brand 03] Matthew Brand. *Charting a manifold*. In Proceedings of the Advances in Neural Information Processing Systems (NIPS). Cambridge, MA, MIT Press, 2003.

[Brown 07] Matthew Brown & David G Lowe. *Automatic panoramic image stitching using invariant features*.

International Journal of Computer Vision, vol. 74, no. 1, pages 59–73, 2007.

[Calonder 10] Michael Calonder, Vincent Lepetit, Christoph Strecha & Pascal Fua. *BRIEF: Binary robust independent elementary features.* In European Conference on Computer Vision (ECCV), pages 778–792. Springer, 2010.

[Cao 16] Yue Cao, Mingsheng Long, Jianmin Wang & Han Zhu. *Correlation autoencoder hashing for supervised cross-modal search.* In Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval, pages 197–204, 2016.

[Cao 18] Yue Cao, Bin Liu, Mingsheng Long & Jianmin Wang. *HashGAN: Deep Learning to Hash with Pair Conditional Wasserstein GAN.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1287–1296, 2018.

[Carreira-Perpinán 15] Miguel A Carreira-Perpinán & Ramin Raziperchikolaei. *Hashing with binary autoencoders.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 557–566, 2015.

[Cayton 05] Lawrence Cayton. *Algorithms for manifold learning.* Univ. of California at San Diego Tech. Rep, vol. 12, no. 1-17, page 1, 2005.

[Charikar 02] Moses S Charikar. *Similarity estimation techniques from rounding algorithms.* In Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, pages 380–388, 2002.

[Chen 15] Lin Chen, Franz Rottensteiner & Christian Heipke. *Feature descriptor by convolution and pooling autoencoders.* International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 40, no. 3W2, pages 31–38, 2015.

[Chen 18] Ricky TQ Chen, Xuechen Li, Roger Grosse & David Duvenaud. *Isolating Sources of Disentanglement in Variational Autoencoders.* arXiv preprint arXiv:1802.04942, 2018.

[Cordes 13] Kai Cordes, Bodo Rosenhahn & Jörn Ostermann. *High-resolution feature evaluation benchmark.* In International Conference on Computer Analysis of Images and Patterns, pages 327–334. Springer, 2013.

[Criminisi 04] Antonio Criminisi, Patrick Pérez & Kentaro Toyama. *Region filling and object removal by exemplar-based image inpainting.* IEEE Transactions on Image Processing, vol. 13, no. 9, pages 1200–1212, 2004.

[Dadaneh 20] Siamak Zamani Dadaneh, Shahin Boluki, Mingzhang Yin, Mingyuan Zhou & Xiaoning Qian. *Pairwise supervised hashing with Bernoulli variational auto-encoder and self-control gradient estimator.* In Conference on Uncertainty in Artificial Intelligence, pages 540–549. PMLR, 2020.

[Dai 17] Bo Dai, Ruiqi Guo, Sanjiv Kumar, Niao He & Le Song. *Stochastic generative hashing.* In International Conference on Machine Learning (ICML), pages 913–922. PMLR, 2017.

[Dalal 05] Navneet Dalal & Bill Triggs. *Histograms of oriented gradients for human detection.* In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 1, pages 886–893. IEEE, 2005.

[d'Angelo 12] Emmanuel d'Angelo, Alexandre Alahi & Pierre Vandergheynst. *Beyond bits: Reconstructing images from local binary descriptors.* In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), pages 935–938. IEEE, 2012.

[Dasgupta 11] Anirban Dasgupta, Ravi Kumar & Tamás Sarlós. *Fast locality-sensitive hashing.* In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1073–1081, 2011.

[Daubechies 04] Ingrid Daubechies, Michel Defrise & Christine De Mol. *An iterative thresholding algorithm for linear inverse problems with a sparsity constraint.* Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences, vol. 57, no. 11, pages 1413–1457, 2004.

[Deng 09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li & Fei-Fei Li. *ImageNet: A large-scale hierarchical image database.* In 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 248–255, 2009.

[Deng 13] Li Deng, Ossama Abdel-Hamid & Dong Yu. *A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with*

phonetic confusion. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pages 6669–6673. IEEE, 2013.

[Dhaene 22] Zeno Dhaene, Nina Žižakić, Shaoguang Huang, Xian Li & Aleksandra Pižurica. *HSIToolbox: a web-based application for the classification of hyperspectral images*. SoftwareX (in preparation), 2022.

[Dizaji 18] Kamran Ghasedi Dizaji, Feng Zheng, Najmeh Sadoughi, Yanhua Yang, Cheng Deng & Heng Huang. *Unsupervised deep generative adversarial hashing network*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3664–3673, 2018.

[Donoho 03] David L Donoho & Carrie Grimes. *Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data*. Proceedings of the National Academy of Sciences, vol. 100, no. 10, pages 5591–5596, 2003.

[En 17] Sovann En, Bruno Crémilleux & Frédéric Jurie. *Unsupervised deep hashing with stacked convolutional autoencoders*. In 2017 IEEE International Conference on Image Processing (ICIP), pages 3420–3424. IEEE, 2017.

[Erin Liong 15] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin & Jie Zhou. *Deep Hashing for Compact Binary Codes Learning*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2475–2483, June 2015.

[Fefferman 16] Charles Fefferman, Sanjoy Mitter & Hariharan Narayanan. *Testing the manifold hypothesis*. Journal of the American Mathematical Society, vol. 29, no. 4, pages 983–1049, 2016.

[Fischer 14a] Philipp Fischer, Alexey Dosovitskiy & Thomas Brox. *Descriptor matching with convolutional neural networks: a comparison to SIFT*. arXiv preprint arXiv:1405.5769, 2014.

[Fischer 14b] Philipp Fischer, Alexey Dosovitskiy & Thomas Brox. *Descriptor matching with convolutional neural networks: a comparison to SIFT*. arXiv preprint arXiv:1405.5769, 2014.

[Frey 10] Brendan Frey. *Frey Faces dataset*, 2010.

[Ghorai 16] Mrinmoy Ghorai, Sekhar Mandal & Bhabatosh Chanda. *Patch sparsity based image inpainting using local patch statistics and steering kernel descriptor*. In 2016 23rd International Conference on Pattern Recognition (ICPR), pages 781–786. IEEE, 2016.

[Gong 13] Y. Gong, S. Lazebnik, A. Gordo & F. Perronnin. *Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 12, pages 2916–2929, 2013.

[Goodfellow 16a] Ian Goodfellow, Yoshua Bengio & Aaron Courville. Deep learning. MIT Press, 2016. http://www.deeplearningbook.org.

[Goodfellow 16b] Ian Goodfellow, Yoshua Bengio & Aaron Courville. Deep learning. MIT Press, 2016. http://www.deeplearningbook.org.

[Goulekas 01] Karen Goulekas. Visual Effects in a Digital World: A Comprehensive Glossary of Over 7000 Visual Effects Terms. Elsevier, 2001.

[Gregor 10] Karol Gregor & Yann LeCun. *Learning fast approximations of sparse coding*. In Proceedings of the 27th International Conference on Machine Learning, pages 399–406, 2010.

[Han 15] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar & Alexander C Berg. *Matchnet: Unifying feature and metric learning for patch-based matching*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3279–3286, 2015.

[Hansen 20] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Stephen Alstrup & Christina Lioma. *Unsupervised Semantic Hashing with Pairwise Reconstruction*. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 2009–2012, 2020.

[He 16] Kaiming He, Xiangyu Zhang, Shaoqing Ren & Jian Sun. *Deep residual learning for image recognition*. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.

[He 18] Kun He, Yan Lu & Stan Sclaroff. *Local descriptors optimized for average precision.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 596–605, 2018.

[Hendrycks 16] Dan Hendrycks & Kevin Gimpel. *Gaussian Error Linear Units (GELUs).* arXiv preprint arXiv:1606.08415, 2016.

[Higgins 17] Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed & Alexander Lerchner. *β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework.* In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017.

[Hinton 94] Geoffrey E Hinton & Richard S Zemel. *Autoencoders, minimum description length and Helmholtz free energy.* In Advances in Neural Information Processing Systems, pages 3–10, 1994.

[Hinton 95] Geoffrey E Hinton, Peter Dayan, Brendan J Frey & Radford M Neal. *The" wake-sleep" algorithm for unsupervised neural networks.* Science, vol. 268, no. 5214, pages 1158–1161, 1995.

[Hinton 06] Geoffrey E Hinton & Ruslan R Salakhutdinov. *Reducing the dimensionality of data with neural networks.* Science, vol. 313, no. 5786, pages 504–507, 2006.

[Hosu 20] Vlad Hosu, Hanhe Lin, Tamas Sziranyi & Dietmar Saupe. *KonIQ-10k: An ecologically valid database for deep learning of blind image quality assessment.* IEEE Transactions on Image Processing, vol. 29, pages 4041–4056, 2020.

[Hu 17] Qinghao Hu, Jiaxiang Wu, Jian Cheng, Lifang Wu & Hanqing Lu. *Pseudo Label Based Unsupervised Deep Discriminative Hashing for Image Retrieval.* In Proceedings of the 25th ACM international conference on Multimedia, pages 1584–1590, 2017.

[Indyk 98] Piotr Indyk & Rajeev Motwani. *Approximate nearest neighbors: towards removing the curse of dimensionality.* In Proceedings of the thirtieth annual ACM symposium on Theory of computing, pages 604–613, 1998.

[Jacobs 85] Paul S Jacobs. *A knowledge-based approach to language production.* Rapport technique, California University Berkeley Dept. of Electrical Engineering and Computer Sciences, 1985.

[Janocha 17] Katarzyna Janocha & Wojciech Marian Czarnecki. *On loss functions for deep neural networks in classification.* arXiv preprint arXiv:1702.05659, 2017.

[Ji 12] Jianqiu Ji, Jianmin Li, Shuicheng Yan, Bo Zhang & Qi Tian. *Super-bit locality-sensitive hashing.* In Advances in Neural Information Processing Systems, pages 108–116. Citeseer, 2012.

[Jin 05] Hongliang Jin, Qingshan Liu, Xiaoou Tang & Hanqing Lu. *Learning local descriptors for face detection.* In 2005 IEEE International Conference on Multimedia and Expo, pages 928–931. IEEE, 2005.

[Joshi 20] Khushbu Joshi & Manish I Patel. *Recent advances in local feature detector and descriptor: a literature survey.* International Journal of Multimedia Information Retrieval, vol. 9, no. 4, pages 231–247, 2020.

[Kaggle 10] Kaggle. *Titanic - Machine Learning from Disaster.* https://rr.noordstar.me/kaggle-challenge-titanic-01632e81, 2010.

[Kaplan 19] Andreas Kaplan & Michael Haenlein. *Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence.* Business Horizons, vol. 62, no. 1, pages 15–25, 2019.

[Ke 04] Yan Ke & Rahul Sukthankar. *PCA-SIFT: A more distinctive representation for local image descriptors.* In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, pages 506–513. IEEE, 2004.

[KIK-IRPA 10] KIK-IRPA. *Closer to Van Eyck.* http://closertovaneyck.kikirpa.be/, 2010.

[Kingma 13] Diederik P Kingma & Max Welling. *Auto-Encoding Variational Bayes.* arXiv preprint arXiv:1312.6114, 2013.

[Kingma 14] Diederik P Kingma & Jimmy Ba. *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.

[Kingma 19] Diederik P Kingma & Max Welling. *An introduction to variational autoencoders.* arXiv preprint arXiv:1906.02691, 2019.

[Komodakis 07] Nikos Komodakis & Georgios Tziritas. *Image completion using efficient belief propagation via priority scheduling and dynamic pruning.* IEEE Transactions on Image Processing, vol. 16, no. 11, pages 2649–2661, 2007.

[Kramer 91] Mark A Kramer. *Nonlinear principal component analysis using autoassociative neural networks.* AIChE journal, vol. 37, no. 2, pages 233–243, 1991.

[Krishna 17] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma*et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations.* International Journal of Computer Vision, vol. 123, no. 1, pages 32–73, 2017.

[Krizhevsky 09] Alex Krizhevsky & Geoffrey Hinton. *Learning multiple layers of features from tiny images.* Technical report, University of Toronto, 2009.

[Kulis 11] Brian Kulis & Kristen Grauman. *Kernelized locality-sensitive hashing.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 6, pages 1092–1104, 2011.

[Le Meur 11] Olivier Le Meur, Josselin Gautier & Christine Guillemot. *Examplar-based inpainting based on local geometry.* In 2011 18th IEEE International Conference on Image Processing (ICIP), pages 3401–3404. IEEE, 2011.

[LeCun 10] Yann LeCun, Corinna Cortes & Chris Burges. *MNIST handwritten digit database*, 2010.

[Lenat 89] Douglas B Lenat & Ramanathan V Guha. Building large knowledge-based systems; representation and inference in the Cyc project. Addison-Wesley Longman Publishing Co., Inc., 1989.

[Leutenegger 11] Stefan Leutenegger, Margarita Chli & Roland Y Siegwart. *BRISK: Binary Robust Invariant Scalable Keypoints.* In 2011 International Conference on Computer Vision (ICCV), pages 2548–2555. IEEE, 2011.

[Lin 13] Yue Lin, Rong Jin, Deng Cai, Shuicheng Yan & Xuelong Li. *Compressed hashing.* In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), pages 446–451, 2013.

[Liu 11] Wei Liu, Jun Wang, Sanjiv Kumar & Shih-Fu Chang. *Hashing with graphs.* In International Conference on Machine Learning (ICML), 2011.

[Liu 16] Haomiao Liu, Ruiping Wang, Shiguang Shan & Xilin Chen. *Deep supervised hashing for fast image retrieval.* In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), pages 2064–2072, 2016.

[Lovelace 42] Ada Lovelace. Notes upon L. F. Menabrea's "Sketch of the Analytical Engine invented by Charles Babbage". 1842.

[Lowe 99] David G Lowe. *Object recognition from local scale-invariant features.* In Proceedings of the Seventh IEEE International Conference on Computer Vision (ICCV), page 1150. IEEE, 1999.

[Luo 20] Xiao Luo, Chong Chen, Huasong Zhong, Hao Zhang, Minghua Deng, Jianqiang Huang & Xiansheng Hua. *A survey on deep hashing methods.* arXiv preprint arXiv:2003.03369, 2020.

[Mahendran 15] Aravindh Mahendran & Andrea Vedaldi. *Understanding deep image representations by inverting them.* In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5188–5196, 2015.

[Makhzani 15] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow & Brendan Frey. *Adversarial autoencoders.* arXiv preprint arXiv:1511.05644, 2015.

[McCann 17] Michael T McCann, Kyong Hwan Jin & Michael Unser. *Convolutional neural networks for inverse problems in imaging: A review.* IEEE Signal Processing Magazine, vol. 34, no. 6, pages 85–95, 2017.

[McCorduck 04] Pamela McCorduck. Machines who think (2nd ed.). A. K. Peters, Ltd., 2004.

[Meeus 19] Laurens Meeus, Shaoguang Huang, Bart Devolder, Hélène Dubois & Aleksandra Pižurica. *Deep learning for paint loss detection with a multiscale, translation*

invariant network. In Proceedings of the 11th International Symposium on Image and Signal Processing and Analysis (ISPA 2019), page 5, 2019.

[Meeus 20] Laurens Meeus, Shaoguang Huang, Nina Žižakić, Xianghui Xie, Bart Devolder, Hélène Dubois, Maximiliaan Martens & Aleksandra Pižurica. *Assisting classical paintings restoration: efficient paint loss detection and descriptor-based inpainting using shared pretraining*. In Optics, Photonics and Digital Technologies for Imaging Applications VI, volume 11353, page 113530H. International Society for Optics and Photonics, 2020.

[Meilă 07] Marina Meilă. *Comparing clusterings – an information based distance*. Journal of Multivariate Analysis, vol. 98, no. 5, pages 873–895, 2007.

[Mikolajczyk 05] Krystian Mikolajczyk & Cordelia Schmid. *A performance evaluation of local descriptors*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 10, pages 1615–1630, 2005.

[Mishkin 15] Dmytro Mishkin, Jiri Matas, Michal Perdoch & Karel Lenc. *WxBS: Wide baseline stereo generalizations*. arXiv preprint arXiv:1504.06603, 2015.

[Misra 19] Diganta Misra. *Mish: A self regularized non-monotonic neural activation function*. arXiv preprint arXiv:1908.08681, vol. 4, no. 2, pages 10–48550, 2019.

[Monga 21] Vishal Monga, Yuelong Li & Yonina C Eldar. *Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing*. IEEE Signal Processing Magazine, vol. 38, no. 2, pages 18–44, 2021.

[Nai 18] Ke Nai, Zhiyong Li, Guiji Li & Shanquan Wang. *Robust object tracking via local sparse appearance model*. IEEE Transactions on Image Processing, vol. 27, no. 10, pages 4958–4970, 2018.

[Narayanan 10] Hariharan Narayanan & Sanjoy Mitter. *Sample complexity of testing the manifold hypothesis*. Advances in neural information processing systems, vol. 23, 2010.

[Newson 17] Alasdair Newson, Andrés Almansa, Yann Gousseau & Patrick Pérez. *Non-local patch-based image inpainting*. Image Processing On Line, vol. 7, pages 373–385, 2017.

[Ng 11] Andrew Ng. *Sparse autoencoder*. CS294A Lecture notes, vol. 72, no. 2011, pages 1–19, 2011.

[Ono 18] Yuki Ono, Eduard Trulls, Pascal Fua & Kwang Moo Yi. *LF-Net: learning local features from images*. In Advances in Neural Information Processing Systems, pages 6234–6244, 2018.

[Ovid 04] Ovid & Charles Martin. Metamorphoses. W. W. Norton, 2004.

[Pan 12] Jia Pan & Dinesh Manocha. *Bi-level locality sensitive hashing for k-nearest neighbor computation*. In 2012 IEEE 28th International Conference on Data Engineering, pages 378–389, 2012.

[Panigrahy 05] Rina Panigrahy. *Entropy based nearest neighbor search in high dimensions*. arXiv preprint cs/0510019, 2005.

[Papernot 16] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow & Kunal Talwar. *Semi-supervised knowledge transfer for deep learning from private training data*. arXiv preprint arXiv:1610.05755, 2016.

[Patel 16] Manish I Patel, Vishvjit K Thakar & Shishir K Shah. *Image registration of satellite images with varying illumination level using HOG descriptor based SURF*. Procedia Computer Science, vol. 93, pages 382–388, 2016.

[Pernici 13] Federico Pernici & Alberto Del Bimbo. *Object tracking by oversampling local features*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 12, pages 2538–2551, 2013.

[Pidhorskyi 20] Stanislav Pidhorskyi, Donald A Adjeroh & Gianfranco Doretto. *Adversarial latent autoencoders*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 14104–14113, 2020.

[Pihlgren 20] Gustav Grund Pihlgren, Fredrik Sandin & Marcus Liwicki. *Improving image autoencoder embeddings with perceptual loss*. In 2020 International Joint Conference on Neural Networks (IJCNN), pages 1–7. IEEE, 2020.

[Pižurica 15] Aleksandra Pižurica, Ljiljana Platiša, Tijana Ružić, Bruno Cornelis, Ann Dooms, Maximiliaan Martens,

Hélène Dubois, Bart Devolder, Marc De Mey & Ingrid Daubechies. *Digital Image Processing of The Ghent Altarpiece: Supporting the painting's study and conservation treatment.* IEEE Signal Processing Magazine, vol. 32, no. 4, pages 112–122, 2015.

[Poddar 19] Shashi Poddar, Rahul Kottath & Vinod Karar. *Motion Estimation Made Easy: Evolution and Trends in Visual Odometry.* In Recent Advances in Computer Vision, pages 305–331. Springer, 2019.

[Qiu 17] Zhaofan Qiu, Yingwei Pan, Ting Yao & Tao Mei. *Deep semantic hashing with generative adversarial networks.* In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 225–234, 2017.

[RAR 22] Rise Above Research RAR. *2022 Worldwide Image Capture Forecast: 2021 - 2026.* https://riseaboveresearch.com/rar-reports/2022-worldwide-image-capture-forecast-2021-2026/, 2022.

[Rezende 18] Danilo Jimenez Rezende & Fabio Viola. *Taming VAEs.* arXiv preprint arXiv:1810.00597, 2018.

[Rosten 06] Edward Rosten & Tom Drummond. *Machine learning for high-speed corner detection.* In European Conference on Computer Vision (ECCV), pages 430–443. Springer, 2006.

[Rublee 11] Ethan Rublee, Vincent Rabaud, Kurt Konolige & Gary Bradski. *ORB: An efficient alternative to SIFT or SURF.* In 2011 International Conference on Computer Vision (ICCV), pages 2564–2571. IEEE, 2011.

[Rumelhart 85] David E Rumelhart, Geoffrey E Hinton & Ronald J Williams. *Learning internal representations by error propagation.* Rapport technique, California Univ., San Diego, La Jolla Inst. for Cognitive Science, 1985.

[Ružić 15] Tijana Ružić & Aleksandra Pižurica. *Context-aware patch-based image inpainting using Markov random field modeling.* IEEE Transactions on Image Processing, vol. 24, no. 1, pages 444–456, 2015.

[Sainath 15] Tara N Sainath, Brian Kingsbury, George Saon, Hagen Soltau, Abdel-rahman Mohamed, George Dahl & Bhuvana Ramabhadran. *Deep convolutional neural networks for large-scale speech tasks.* Neural Networks, vol. 64, pages 39–48, 2015.

[Schmidt 21] Robin M Schmidt, Frank Schneider & Philipp Hennig. *Descending through a Crowded Valley – Benchmarking Deep Learning Optimizers.* In International Conference on Machine Learning (ICML), pages 9367–9376. PMLR, 2021.

[Schonberger 17] Johannes L Schonberger, Hans Hardmeier, Torsten Sattler & Marc Pollefeys. *Comparative evaluation of hand-crafted and learned local features.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1482–1491, 2017.

[Sheikh 06] Hamid R Sheikh & Alan C Bovik. *Image information and visual quality.* IEEE Transactions on Image Processing, vol. 15, no. 2, pages 430–444, 2006.

[Shen 15] Fumin Shen, Chunhua Shen, Wei Liu & Heng Tao Shen. *Supervised discrete hashing.* In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR), pages 37–45, 2015.

[Shen 20] Yuming Shen, Jie Qin, Jiaxin Chen, Mengyang Yu, Li Liu, Fan Zhu, Fumin Shen & Ling Shao. *Auto-Encoding Twin-Bottleneck Hashing.* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 2815–2824, 2020.

[Shi 16] Xiaoshuang Shi, Fuyong Xing, Jinzheng Cai, Zizhao Zhang, Yuanpu Xie & Lin Yang. *Kernel-based supervised discrete hashing for image retrieval.* In European Conference on Computer Vision (ECCV), pages 419–433. Springer, 2016.

[Simo-Serra 15] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua & Francesc Moreno-Noguer. *Discriminative learning of deep convolutional feature point descriptors.* In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 118–126, 2015.

[Simonyan 14] Karen Simonyan & Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition.* arXiv preprint arXiv:1409.1556, 2014.

[Sizyakin 20] Roman Sizyakin, Bruno Cornelis, Laurens Meeus, Hélène Dubois, Maximiliaan Martens, Viacheslav Voronin & Aleksandra Pižurica. *Crack detection in paintings using convolutional neural networks.* IEEE Access, vol. 8, pages 74535–74552, 2020.

[Solomonoff 56] Ray Solomonoff. *Ray's notes on his Thinking Machine ideas.* http://raysolomonoff.com/dartmouth/boxbdart/dart56ray812825who.pdf, 1956. Accessed: 2022-03-12.

[Song 18] Jingkuan Song, Tao He, Lianli Gao, Xing Xu, Alan Hanjalic & Heng Tao Shen. *Binary generative adversarial networks for image retrieval.* In Proceedings of the AAAI Conference on Artificial Intelligence, volume 32, 2018.

[Sparkes 96] Brian A. Sparkes. The Red and the Black: Studies in Greek Pottery. Routledge, 1996.

[Strecha 11] Christoph Strecha, Alex Bronstein, Michael Bronstein & Pascal Fua. *LDAHash: Improved matching with smaller descriptors.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 1, pages 66–78, 2011.

[Tan 19] Mingxing Tan & Quoc Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.* In International Conference on Machine Learning (ICML), pages 6105–6114. PMLR, 2019.

[Tandy 97] David W. Tandy. Works and days: A translation and commentary for the social sciences. University of California Press, 1997.

[Tian 19] Yurun Tian, Xin Yu, Bin Fan, Fuchao Wu, Huub Heijnen & Vassileios Balntas. *SOSNet: Second order similarity regularization for local descriptor learning.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 11016–11025, 2019.

[Tobias-Springenberg 14] Jost Tobias-Springenberg, Alexey Dosovitskiy, Thomas Brox & Martin Riedmiller. *Striving for simplicity: The all convolutional net.* arXiv preprint arXiv:1412.6806, 2014.

[Tola 09] Engin Tola, Vincent Lepetit & Pascal Fua. *DAISY: An efficient dense descriptor applied to wide-baseline stereo.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 5, pages 815–830, 2009.

[Tolstikhin 17] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly & Bernhard Schoelkopf. *Wasserstein auto-encoders.* arXiv preprint arXiv:1711.01558, 2017.

[Tošić 11] Ivana Tošić & Pascal Frossard. *Dictionary learning: What is the right representation for my signal?* IEEE Signal Processing Magazine, vol. 28, no. 2, pages 27–38, 2011.

[Turing 50] Alan M Turing. *Computing Machinery and Intelligence.* Mind, vol. LIX, no. 236, pages 433–460, 10 1950.

[Tuytelaars 08] Tinne Tuytelaars & Krystian Mikolajczyk. Local Invariant Feature Detectors: A Survey. Now Publishers Inc, 2008.

[Verwilst 21] Maxim Verwilst, Nina Žižakić, Lingchen Gu & Aleksandra Pižurica. *Deep image hashing based on twin-bottleneck hashing with variational autoencoders.* In 2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP). IEEE, 2021.

[Vondrick 13] Carl Vondrick, Aditya Khosla, Tomasz Malisiewicz & Antonio Torralba. *Hoggles: Visualizing object detection features.* In Proceedings of the IEEE International Conference on Computer Vision, pages 1–8, 2013.

[Voronin 14] Viacheslav V Voronin, Vladimir I Marchuk, Nikolay V Gapon, Roman A Sizyakin, Alexander I Sherstobitov & Karen O Egiazarian. *Exemplar-based inpainting using local binary patterns.* In Image Processing: Algorithms and Systems XII, volume 9019, page 901907. International Society for Optics and Photonics, 2014.

[Wald 00] Lucien Wald. *Quality of high resolution synthesised images: Is there a simple criterion?* In Third conference "Fusion of Earth data: merging point measurements, raster maps and remotely sensed images", pages 99–103. SEE/URISCA, January 2000.

[Wang 02] Zhou Wang & Alan C Bovik. *A universal image quality index.* IEEE Signal Processing Letters, vol. 9, no. 3, pages 81–84, 2002.

[Wang 03] Zhou Wang, Eero P Simoncelli & Alan C Bovik. *Multiscale structural similarity for image quality assessment.* In The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003, volume 2, pages 1398–1402. IEEE, 2003.

[Wang 04] Zhou Wang, Alan C Bovik, Hamid R Sheikh & Eero P Simoncelli. *Image quality assessment: from*

error visibility to structural similarity. IEEE transactions on image processing, vol. 13, no. 4, pages 600–612, 2004.

[Wang 11] Sheng Wang. *A Review of Gradient-Based and Edge-Based Feature Extraction Methods for Object Detection.* In 2011 IEEE 11th International Conference on Computer and Information Technology, pages 277–282. IEEE, 2011.

[Wang 16a] Xiaofang Wang, Yi Shi & Kris M Kitani. *Deep supervised hashing with triplet labels.* In Asian conference on computer vision, pages 70–84. Springer, 2016.

[Wang 16b] Xiaojuan Wang, Ting Zhang, Guo-Jun Qi, Jinhui Tang & Jingdong Wang. *Supervised quantization for similarity search.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2018–2026, 2016.

[Wang 17] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen*et al. A survey on learning to hash.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 4, pages 769–790, 2017.

[Wang 18] Guan'an Wang, Qinghao Hu, Jian Cheng & Zengguang Hou. *Semi-supervised generative adversarial hashing for image retrieval.* In Proceedings of the European Conference on Computer Vision (ECCV), pages 469–485, 2018.

[Wang 20] Qianqian Wang, Xiaowei Zhou, Bharath Hariharan & Noah Snavely. *Learning feature descriptors using camera pose supervision.* In European Conference on Computer Vision (ECCV), pages 757–774. Springer, 2020.

[Weinberger 06] Kilian Q Weinberger & Lawrence K Saul. *Unsupervised learning of image manifolds by semidefinite programming.* International Journal of Computer Vision, vol. 70, no. 1, pages 77–90, 2006.

[Weinzaepfel 11] Philippe Weinzaepfel, Hervé Jégou & Patrick Pérez. *Reconstructing an image from its local descriptors.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011), pages 337–344. IEEE, 2011.

[Weiss 09] Yair Weiss, Antonio Torralba & Rob Fergus. *Spectral Hashing.* In Advances in Neural Information Processing Systems 21, pages 1753–1760. Curran Associates, Inc., 2009.

[Wikipedia 22] Wikipedia. *Autoencoder – Wikipedia.* https://en.wikipedia.org/wiki/Autoencoder, 2022. Accessed: 2022-09-09.

[Winder 07] Simon AJ Winder & Matthew Brown. *Learning local image descriptors.* In 2007 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–8. IEEE, 2007.

[Xia 14] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu & Shuicheng Yan. *Supervised hashing for image retrieval via image representation learning.* In Proceedings of the AAAI conference on artificial intelligence, volume 28, 2014.

[Xiao 17] Han Xiao, Kashif Rasul & Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*, 2017.

[Yang 17] Huei-Fang Yang, Kevin Lin & Chu-Song Chen. *Supervised learning of semantics-preserving hash via deep convolutional neural networks.* IEEE transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 2, pages 437–451, 2017.

[Yilmaz 06] Alper Yilmaz, Omar Javed & Mubarak Shah. *Object Tracking: A Survey.* ACM computing surveys (CSUR), vol. 38, no. 4, pages 13–es, 2006.

[Yim 10] Changhoon Yim & Alan Conrad Bovik. *Quality assessment of deblocked images.* IEEE Transactions on Image Processing, vol. 20, no. 1, pages 88–98, 2010.

[Zagoruyko 15] Sergey Zagoruyko & Nikos Komodakis. *Learning to compare image patches via convolutional neural networks.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4353–4361, 2015.

[Zeiler 12] Matthew D Zeiler. *ADADELTA: an adaptive learning rate method.* arXiv preprint arXiv:1212.5701, 2012.

[Zhang 19] Na Zhang, Hua Ji, Li Liu & Guanhua Wang. *Exemplar-based image inpainting using angle-aware patch matching.* EURASIP Journal on Image and Video Processing, vol. 2019, no. 1, page 70, 2019.

[Zhou 98] Jie Zhou, Daniel L Civco & JA Silander. *A wavelet transform method to merge Landsat TM and SPOT panchromatic data.* International Journal of Remote Sensing, vol. 19, no. 4, pages 743–757, 1998.

[Zieba 18] Maciej Zieba, Piotr Semberecki, Tarek El-Gaaly & Tomasz Trzcinski. *BinGAN: Learning Compact Binary Descriptors with a Regularized GAN*. Advances in Neural Information Processing Systems, vol. 31, 2018.

[Žižakić 19a] Nina Žižakić, Izumi Ito, Laurens Meeus & Aleksandra Pižurica. *Autoencoder-learned local image descriptor for image inpainting*. In BNAIC/BENELEARN 2019, volume 2491, 2019.

[Žižakić 19b] Nina Žižakić, Izumi Ito & Aleksandra Pižurica. *Learning Local Image Descriptors with Autoencoders*. In International Conference on Image Processing and Communications, pages 214–221. Springer, 2019.

[Žižakić 20a] Nina Žižakić & Aleksandra Pižurica. *Learned BRIEF – transferring the knowledge from hand-crafted to learning-based descriptors*. In 2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP). IEEE, 2020.

[Žižakić 20b] Nina Žižakić & Aleksandra Pižurica. *Invertible local image descriptors learned with variational autoencoders*. In IEICE Information and Communication Technology Forum (ICTF) 2020, Proceedings. IEICE Europe Section, 2020.

[Žižakić 21a] Nina Žižakić & Aleksandra Pižurica. *Efficient Local Image Descriptors Learned with Autoencoders*. IEEE Access, vol. 10, pages 221–235, 2021.

[Žižakić 21b] Nina Žižakić & Aleksandra Pižurica. *β-variational autoencoders for learning invertible local image descriptors*. Image Processing & Communications, vol. 24, no. 1, pages 71–78, 2021.

[Žižakić 22] Nina Žižakić & Aleksandra Pižurica. *How to Efficiently Evaluate Autoencoders for Learning Local Image Descriptors*. In review, European Signal Processing Conference (EUSIPCO), 2022.