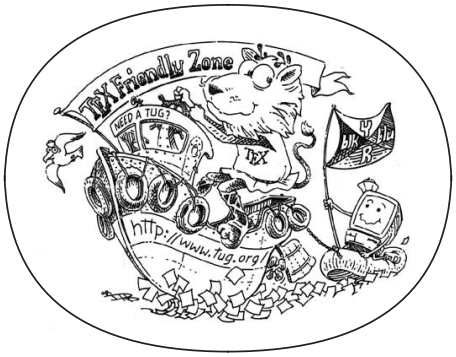
EXPLORING AND LEARNING FROM VISUAL DATA

YANNIS AVRITHIS



A Representation Space Odyssey

February 2020 – version 0.3

"Every revolutionary idea seems to evoke three stages of reaction.
(1) It's completely impossible. (2) It's possible, but it's not worth doing.
(3) I said it was a good idea all along."

— Sir Arthur C. Clarke

Dedicated to Maria, Savvas and Apostolos

ABSTRACT

This manuscript is about a *journey*. The journey of computer vision and machine learning research from the early years of Gabor filters and linear classifiers to surpassing human skills in several tasks today. The journey of the author's own research, designing representations and matching processes *to explore* visual data and exploring visual data *to learn* better representations.

PART I addresses instance-level visual search and clustering, building on shallow visual representations and matching processes. The representation is obtained by a pipeline of local features, hand-crafted descriptors and visual vocabularies. Improvements in the pipeline are introduced, including the construction of large scale *vocabularies* [18], *spatial matching* for geometry verification [20, 433], representations *beyond vocabularies* [436, 437] and nearest neighbor search [213]. Applications to *exploring photo collections* are discussed, including location recognition, landmark recognition and automatic discovery of photos depicting the same scene [19, 215].

PART II addresses instance-level visual search and object discovery, building on deep visual representations and matching processes, focusing on the manifold structure of the feature space. The representation is obtained by deep parametric models learned from visual data. Contributions are made to advancing *manifold search* over global or regional CNN representations. This process is seen as graph filtering, including *spatial* [194] and *spectral* [189]. *Spatial matching* is revisited with local features detected on CNN activations [405]. Finally, *object discovery* from CNN activations over an unlabeled image collection [406, 407] is introduced.

PART III addresses learning deep visual representations by exploring visual data, focusing on limited or no supervision. It progresses from instance-level to category-level tasks and studies the sensitivity of models to their input. It introduces methods for category-level and instance-level tasks using limited supervision, including *unsupervised metric learning* [192], *semi-supervised learning* [193] and a *few-shot learning* [261]. The latter studies activation maps and learns multiple layers to convergence for the first time. Finally, an attack is introduced as an attempt to improve upon the visual quality of *adversarial examples* in terms of imperceptibility [505].

PART IV summarizes more of the author's past and present contributions, reflects on these contributions in the present context and consolidates the ideas exposed in this manuscript. It then attempts to draw a road map of ideas that are likely to come.

PUBLICATIONS

The contributions presented in this manuscript have appeared previously in the following articles. All articles are available online. For easy access, the titles are hyperlinked.

[1] Yannis Avrithis and Yannis Kalantidis. "Approximate Gaussian Mixtures for Large Scale Vocabularies." In: *ECCV*. 2012.

[2] Yannis Avrithis, Yannis Kalantidis, Giorgos Tolias, and Evaggelos Spyrou. "Retrieving Landmark and Non-Landmark Images from Community Photo Collections." In: *ACM Multimedia*. 2010.

[3] Yannis Avrithis and Giorgos Tolias. "Hough Pyramid Matching: Speeded-Up Geometry Re-Ranking for Large Scale Image Retrieval." In: *IJCV* 107.1 (2014), pp. 1–19.

[4] Ahmet Iscen, Yannis Avrithis, Giorgos Tolias, Teddy Furon, and Ondrej Chum. "Fast Spectral Ranking for Similarity Search." In: *CVPR*. 2018.

[5] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. "Mining on Manifolds: Metric Learning without Labels." In: *CVPR*. 2018.

[6] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. "Label Propagation for Deep Semi-Supervised Learning." In: *CVPR*. 2019.

[7] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, Teddy Furon, and Ondrej Chum. "Efficient Diffusion on Region Manifolds: Recovering Small Objects with Compact CNN Representations." In: *CVPR*. 2017.

[8] Yannis Kalantidis and Yannis Avrithis. "Locally Optimized Product Quantization for Approximate Nearest Neighbor Search." In: *CVPR*. 2014.

[9] Yannis Kalantidis, Giorgos Tolias, Yannis Avrithis, Marios Phinikettos, Evaggelos Spyrou, Phivos Mylonas, and Stefanos Kollias. "VIRaL: Visual Image Retrieval and Localization." In: *Multimedia Tools and Applications* 51.2 (2011), pp. 555–592.

[10] Yann Lifchitz, Yannis Avrithis, Sylvaine Picard, and Andrei Bursuc. "Dense Classification and Implanting for Few-shot Learning." In: *CVPR*. 2019.

[11] Oriane Siméoni, Yannis Avrithis, and Ondrej Chum. "Local Features and Visual Words Emerge in Activations." In: *CVPR*. 2019.

[12] Oriane Simeoni, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. "Unsupervised Object Discovery for Instance Recognition." In: *WACV*. 2018.

[13] Oriane Siméoni, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. "Graph-Based Particular Object Discovery." In: *Machine Vision and Applications* 30.2 (Mar. 2019), pp. 243–254.

[14] Giorgos Tolias and Yannis Avrithis. "Speeded-Up, Relaxed Spatial Matching." In: *ICCV*. 2011.

[15] Giorgos Tolias, Yannis Avrithis, and Hervé Jégou. "To Aggregate or not to Aggregate: Selective Match Kernels for Image Search." In: *ICCV*. 2013.

[16] Giorgos Tolias, Yannis Avrithis, and Hervé Jégou. "Image Search with Selective Match Kernels: Aggregation across Single and Multiple Images." In: *IJCV* 116.3 (2016), pp. 247–261.

[17] Hanwei Zhang, Yannis Avrithis, Teddy Furon, and Laurent Amsaleg. "Smooth Adversarial Examples." In: *arXiv preprint arXiv: 1903.11862* (Mar. 2019).

CONTENTS

ACRONYMS

ADC	Asymmetric Distance Computation . . . 41
AGM	Approximate Gaussian Mixture . . . 16
AKM	Approximate k-Means . . . 21
ANN	Approximate Nearest Neighbor . . . 16
AP	Average Precision . . . 19
AQE	Average Query Expansion . . . 73
ASMK	Aggregated Selective Match Kernel . . . 18
BoW	Bag of Words . . . 13
BP	Boundary Projection . . . 133
CAM	Class Activation Mapping . . . 118
CG	Conjugate Gradient . . . 60
CIFAR	Canadian Institute for Advanced Research . . . 113
CKM	Cartesian k-Means . . . 41
CNN	Convolutional Neural Network . . . 5
CroW	Cross-Dimensional Weighting . . . 82
CW	Carlini & Wagner . . . 123
CUB	Caltech-UCSD Birds . . . 106
DC	Dense Classification . . . 101
DELF	DEep Local Features . . . 61
DenseNet	Densely Connected Network . . . 57
DFT	Discrete Fourier Transform . . . 67
DLP	Deep Label Propagation . . . 99
DoF	Degrees of Freedom . . . 17
DoG	Difference of Gaussians . . . 14
DRVQ	Dimensionality-Recursive Vector Quantization . . . 131
DSM	Deep Spatial Matching . . . 62
EGM	Expanding Gaussian Mixture . . . 20
EM	Expectation-Maximization . . . 16
EMA	Exponential Moving Average . . . 65
FC	Fully-Connected . . . 58
FGSM	Fast Gradient Sign Method . . . 122
FLANN	Fast Library for ANN . . . 25
FLOPS	FLoating-point Operations Per Second . . . 143

FMH	Feature Map Hashing . . . 131
FSL	Few-Shot Learning . . . 102
FSM	Fast Spatial Matching . . . 27
FSR	Fast Spectral Ranking . . . 60
FS	Feature Saliency . . . 88
GAP	Global Average Pooling . . . 117
GCN	Graph Convolutional Network . . . 133
GeM	Generalized Mean . . . 59
GHT	Generalized Hough Transform . . . 13
GMM	Gaussian Mixture Model . . . 16
GMP	Generalized Max Pooling . . . 75
GOD	Graph-based Object Discovery . . . 63
GPU	Graphics Processing Unit . . . 57
GSP	Graph Signal Processing . . . 5
HDR	Habilitation á Diriger des Recherches . . . 2
HE	Hamming Embedding . . . 18
HKM	Hierarchical k-Means . . . 21
HPM	Hough Pyramid Matching . . . 17
HOG	Histogram of Oriented Gradients . . . 15
I-FGSM	Iterative FGSM . . . 122
IDF	Inverse Document Frequency . . . 16
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge 57
IoU	Intersection over Union . . . 85
IQM	Inverted-Quantized k-Means . . . 131
IRISA	Institut de Recherche en Informatique et Systèmes Aléatoires . . . 2
IVFADC	Inverted File - ADC . . . 41
k-NN	k-Nearest Neighbor . . . 60
KVQ	Kernel Vector Quantization . . . 19
LIFT	Learned Invariant Feature Transform . . . 61
LoG	Laplacian of Gaussian . . . 14
LO-RANSAC	Locally Optimized RANSAC . . . 48
LOPQ	Locally Optimized Product Quantization . . . 18
LP	Label Propagation . . . 69
MA	Multiple Assignment . . . 17
MAC	Maximum Activation of Convolutions . . . 59
mAP	mean Average Precision . . . 19

MDS	Multidimensional Scaling 97
MFD	Medial Feature Detector 130
ML	Maximum Likelihood 22
MoM	Mining on Manifolds 98
mP	mean Precision 19
MSER	Maximally Stable Extremal Regions 83
MT	Mean Teacher 113
Multi-LOPQ	Multi-Index LOPQ 43
Multi-D-ADC	Multi-Index ADC 41
NAS	Neural Architecture Search 96
NCA	Neighborhood Component Analysis 98
NCM	Nearest Class Mean 100
NMI	Normalized Mutual Information 106
NMS	Non-Maximum Suppression 84
NSOD	Nano-Supervised Object Detection 132
OMulti-D-OADC	Optimized Multi-Index ADC 41
OPQ	Optimized Product Quantization 18
OS	Object Saliency 90
PN	Prototypical Networks 116
PCA	Principal Component Analysis 18
PMK	Pyramid Match Kernel 17
PQ	Product Quantization 18
QE	Query Expansion 49
R-CNN	Regions with CNN features 58
R-MAC	Regional Maximum Activation of Convolutions . . . 59
R-Match	Regional Matching 78
RAKM	Robust Approximate k-Means 21
RANSAC	RANdom SAmple Consensus 17
ReLU	Rectified Linear Unit 57
ResNet	Residual Network 57
RevOP	Revisited Oxford and Paris 20
RoI	Region of Interest 130
RPN	Region Proposal Network 58
RWR	Random Walk with Restart 60
sCW	smooth Carlini & Wagner 125
SfM	Structure-from-Motion 19
SGD	Stochastic Gradient Descent 5

SIFT	Scale-Invariant Feature Transform 13
SPM	Spatial Pyramid Matching 134
SSL	Semi-Supervised Learning 102
STIP	Space-Time Interest Point 130
SURF	Speeded-up Robust Features 14
SVM	Support Vector Machine 16
TILDE	Temporally Invariant Learned DEtector 61
TF	Term Frequency 16
VGG	Visual Geometry Group 57
VIRaL	Visual Image Retrieval and Localization 19
UAP	Universal Adversarial Perturbation 101
VLAD	Vector of Locally Aggregated Descriptors 18
WGC	Weak Geometric Consistency 31
WSOD	Weakly-Supervised Object Detection 132
WTA	Winner-Take-All 62

1 ORIENTATION

This manuscript is about designing representations and matching processes to explore *visual data and exploring visual data* to learn *better representations. This chapter begins by motivating such an endeavor in seeking to replicate the functionalities of the human visual system. It sets the goals of this manuscript and walks through the subjects studied in its different parts, while providing relevant background from the fields of* computer vision *and* machine learning. *Finally, it includes a more detailed structure of the manuscript, a list of chapter dependencies and a reading guide.*

1.1 MOTIVATION

The impatient reader wondering how to read this manuscript may want to jump to Section 1.4 right away.

We are born into this world and we open our eyes. Everything is blurry but we can discriminate colors from white and recognize the face of our mother within days. We develop a perception of depth a few weeks later, long before we begin crawling. Although we can focus on particular objects at two months, it is only at about six months that we can perceive sharp images.

As our *visual system* develops, we begin to encode, store and retrieve detailed mental images of vast amounts of objects, places or people. This occurs over a time range spanning from eye movements to years. As we grow up, little recollection remains from the first few years of our life. We maintain an up-to-date mental image of beloved persons and we are at times surprised to find out in photographs what they looked like several years ago.

How do we perceive the world around us? What is the form of mental representations stored in our *visual memory*? How do we organize objects into abstract categories, while at the same time recognizing particular ones instantly? Is there a precise location in the brain where a particular object, place or person is stored, or is the information distributed? What mechanisms limit the capacity of our memory and dictate the evolution of stored representations over time? What is the role of other senses in the development of *vision*?

What would be the impact if we could *unlock* the mysteries of the human visual system and visual memory and if we had access to hardware powerful enough to *replicate* its functionalities? What if we could afford such an artificial system for every person suffering from impaired vision or memory disorders? What if such systems could be networked into a collective memory of gigantic capacity, still able to recognize instantly?

While the unlocking part is the subject of *neuroscience*, the replicating part is primarily the subject of *computer vision* and *machine learning*. Often taking inspiration from the former over decades, the latter two are currently undergoing a period of spectacular growth.

1.2 ABOUT THIS MANUSCRIPT

A journey. This manuscript is about a *journey*. The journey of computer vision and machine learning research from the early years of Gabor filters and linear classifiers to surpassing human skills in several tasks today. The journey of my own research, designing representations and matching processes *to explore* visual data and exploring visual data *to learn* better representations. My personal journey, exploring this vibrant scientific field and learning from my own experience and the experience of colleagues over time.

Contributions. This manuscript summarizes the research I have been conducting with collaborators during roughly the last 10 years, focusing more on recent work, and draws perspectives on future directions. Clearly, putting together work of such broad thematic and temporal range is challenging in terms of size, coherence and terminology. For this reason, I am focusing on a small number of selected articles that I expose to some length. A complete and updated list of my publications and professional activities is maintained online[1].

HDR. This manuscript is a requirement for the qualification of *Habilitation á Diriger des Recherches* (HDR) by University of Rennes 1. According to a recommendation by *Institut de Recherche en Informatique et Systèmes Aléatoires* (IRISA), an HDR manuscript is a "synthetic document of 30 to 50 pages (excluding attached articles and appendices)."

Design. On one hand, a document of such length requires significant effort to prepare, without giving the reader a good understanding of the presented methods. On the other hand, attaching articles has little value since articles are too detailed to be read in large quantity. Besides, they are available online.

For this reason, I choose to also include a self-contained summary of each selected article, mostly one per chapter. These technical chapters are intermixed with non-technical chapters corresponding to the "synthetic document." The latter provide context and background as well as summary and positioning of the contributions.

This design, discussed in Section 1.4, serves a dual purpose: meeting the HDR expectations and providing a succinct, unified presentation of my research contributions in the broad context of developments in *computer vision*, *machine learning* and other fields of study over a period of several decades. It is thus my hope that the manuscript will be read by more people than my jury.

Structure. The manuscript consists of three technical parts, each containing a collection of contributions corresponding to a different period or subject; and a fourth part consolidating the contributions and drawing perspectives on future work. Section 1.3 discusses in detail the contents of the different parts.

This manuscript studies solutions to similar problems in two different periods: before and after the establishment of *deep learning* as the dominant paradigm in representing and understanding visual data.

1 https://avrithis.net

This provides an excellent opportunity to appreciate the impact of deep learning on the relative importance given to different research directions by the scientific community.

"We".

Even though I am the author of this manuscript, hence take full responsibility for its content, most contributions are the result of my collaboration with colleagues. For this reason, I use first-person plural when referring to own contributions from this point onward.

1.3 EXPLORING AND LEARNING

This section serves as a mild introduction to the subjects studied in this manuscript. It introduces a road map and terminology, then guides through the different parts. Aspiring to be read easily by non-experts, it is kept abstract and does not include references other than to our own contributions. It is essentially a high-level summary of Chapters 2, 8, 13, 18 and 19. As explained in Section 1.4, this sequence of chapters provides a non-technical synthesis of our contributions and is recommended to follow, at least at first reading.

ROAD MAP We begin with representations and matching processes for exploring visual data: first based on shallow representations in PART I, then based on deep learned representations in PART II. We then turn to learning visual representations by exploring visual data in PART III, progressing towards category-level tasks. A discussion of past, present and future ideas is attempted in PART IV.

Exploring.

By *exploring* we refer to processes that are linear in the data size, like *searching* for (or *retrieving*) the most similar images within a dataset to a given image; and processes that are quadratic, like *clustering* or *discovering objects* within a dataset. These processes rely on the definition of *visual similarity* between two images.

Representations.

By *representations* we refer to simple, compact visual representations, like a vector or a set of vectors per image. A representation may be *learned* on some training data. We focus on explicit mappings from the input image to the representation space, such that extending to images unseen during training is straightforward.

Matching processes.

By *matching processes* we refer to processes for evaluating the visual similarity of two images. This can be based on a simple Euclidean distance or cosine similarity in the case of one vector per image. A set of vectors per image allows for *partial similarity*, *i.e.* searching for a part of an image that is similar to a part of the other.

Instance-level and category-level tasks.

In *instance-level* tasks, two images are visually similar if they depict two views of the same instance, *e.g.* object or scene; in *category-level* tasks, they are similar if they depict different instances of the same semantic category, *e.g.* birds or cars. The distinction refers mainly to the amount of appearance variability to which the representation should be invariant to.

EXPLORING PART I addresses instance-level visual search and clus-

This is a summary of Chapter 2.

tering, building on shallow visual representations and matching processes. The representation is obtained by a pipeline of local features, hand-crafted descriptors and visual vocabularies.

Local features. By *local features* we refer to a large set of small geometric primitives per image, like points, circles or ellipses. Those features are stable to *lighting* and *viewpoint* changes, in the sense that a geometric transformation of the image causes the same transformation on the features. Splitting images into many small pieces and finding which pieces they have in common is an efficient way to measure partial similarity, providing robustness to *occlusion* and background *clutter*.

Visual descriptors. By *visual descriptor* we refer to a vector representation of an image or part of an image. *Local descriptors* are obtained on rectangular patches aligned with local features. Descriptors are stable to *lighting* changes and *discriminative* in the sense that proximity in the descriptor space reliably indicates correspondence of the associated features. Hand-crafted descriptors commonly take the form of *orientation histograms*, inspired by biological visual systems. *Hand-crafted*, introduced in retrospect, means not *learned*.

Visual vocabularies. By *visual vocabulary* we refer to a set of descriptors learned without supervision from visual data, which allows the formation of a second layer of representation on top of visual descriptors. This representation takes again the form of *histograms* in the high-dimensional descriptor space. It is a compact vector representation of regions or entire images, *invariant* to geometric configuration, but not very discriminative. While vocabularies are learned, this representation is still referred to as *shallow*, a term introduced in retrospect as opposed to *deep*, *i.e.*, consisting of more layers.

Improving representations and matching. We introduce improvements to this pipeline: a mechanism to learn *large-scale vocabularies* that automatically adjusts the size of the vocabulary to the data [18]; an extremely fast *spatial matching* process to identify common objects between two views, allowing for multiple objects or deformation [20, 433]; and a high-dimensional representation *beyond vocabularies* that is very discriminative [436, 437].

Exploring visual data. Using the same representation pipeline, we also introduce improvements in *exploring* visual data: a *compression* and *indexing* scheme that adapts to the distribution in the descriptor space and allows for fast *search* in the compressed domain at a very large scale [213]; and a *clustering* scheme that finds multiple views of the same scene, builds a joint representation without supervision and uses this representation to improve search [19]. The latter is applied to *exploring photo collections*, supporting location and landmark recognition [215].

This is a summary of Chapter 8. EXPLORING DEEPER PART II addresses instance-level visual search and object discovery, building on deep visual representations and matching processes, focusing on the manifold structure of the feature space. The representation is obtained by deep parametric models learned from visual data.

Deep parametric models. By *deep parametric models* we refer to functions mapping images to a representation space, composed of several linear and nonlinear

operations, determined by a significant amount of parameters. The most common is a *Convolutional Neural Network* (CNN), where each operation is called a *layer*. Operations apply densely, resulting in 3-dimensional *activation tensors* that can be seen as one visual descriptor per location. Gradually, the dimension of the representation increases, while the spatial resolution decreases, introducing invariance to deformation. This is reminiscent of the *encoding* and *pooling* steps of histograms in shallow models.

Learning and optimization.

By *learning* we refer to the process of determining the parameters of a deep model, by jointly *optimizing* an objective function on a target task on some training data. The most common task is *supervised classification*, where the model maps directly to a vector of class scores and the objective is optimal when the maximum score is attained for a given class label per image. At large scale, the most common form of optimization is *Stochastic Gradient Descent* (SGD), relying on first-order derivatives only. These derivatives are computed efficiently by *automatic differentiation*, a dynamic programming approach known as *back-propagation* in this context. The difficulties lie in optimizing in the absence of higher-order derivatives and in designing operations such that propagated derivatives are "stable".

Learning for instance-level search.

One of the interesting aspects of deep models is that, regardless of the initial task, representations obtained at intermediate layers may be re-used for or adapted to new tasks, where less training data is available; the latter is called *transfer learning*. For *instance-level search*, it is common to obtain one or more *feature* vectors[2] per image by *spatial pooling* of activation maps, globally or over rectangular regions respectively. Alternatively, one may obtain a large set of *local descriptors* by some form of sampling.

Features.

The new target task may still be supervised classification at instance level: All views of an object belong to the same class. Importantly, the classes are different at training and testing; hence at least the last layer, a parametric classifier, is discarded. Alternatively, it may be a form of *metric learning*, where the objective function is defined on two or more images and is optimal when the *ranking* obtained by the representations is correct. Most importantly, the *supervision* may be provided by existing hand-crafted mechanisms.

Manifold search.

The power of deep representations lies in that one or few vectors per image are discriminative enough to outperform shallow representations consisting of thousands of vectors per image. This allows us to explore the global manifold structure of the feature space by means of *nearest-neighbor graphs* and search beyond Euclidean distance, according to *manifold similarity*. We introduce a number of improvements that make this search process practical at large scale and make connections to *Graph Signal Processing* (GSP), casting search as *graph filtering* [189, 190, 194].

2 *Feature* means representation here. In shallow representations, *feature* is a geometric primitive, while representation is referred to as (appearance) *descriptor*. To avoid confusion, we use *local feature* for the geometric primitive unless obvious from context, and *descriptor* more often than feature for the representation.

Spatial matching revisited.

When searching for small objects in large images, *partial similarity* is still needed. In this case, a representation consisting of thousands of deep learned local descriptors is even more discriminative and allows for *spatial matching* much like shallow counterparts. We observe that *local features* may be obtained simply at local maxima over the 3d activation maps without training and we use such local features for spatial matching, even without descriptors [405].

Object discovery.

An alternative approach to partial similarity is to identify which regions of an image may correspond to objects of interest and focus attention to those regions only, suppressing background clutter. This is only feasible using the entire image collection to be searched. We introduce a nonparametric *attention mechanism* that is learned without supervision on the target collection, capturing discriminative and frequent patterns according to activation maps [406, 407].

This is a summary of Chapter 13.

LEARNING PART III addresses learning visual representations by exploring visual data, focusing on solutions assuming only limited or no supervision. It progresses from instance-level to category-level tasks and studies the sensitivity of models to their input.

Research trends.

The better deep models and learning processes are understood, the more interesting things happen. Learning with *less supervision* becomes a priority: Representations are learned by observing visual data without human involvement, or with human supervision on tiny fractions of the data, or with noisy supervision from different sources on the Internet. *Category-level* and *instance-level* tasks converge in the sense that they are mostly treated the same way, the only difference lying in the data and annotations defining the task. *New tasks* appear; or, tasks of marginal interest become very popular. The research landscape changes within months.

Metric learning.

One of the tasks resisting recent trends is *metric learning*. Originating in unsupervised *nonlinear dimension reduction* or *manifold learning* from pairs of "similar" and "dissimilar" examples, *supervised* metric learning arises as a task where such pairs are specified by humans. In early *deep* metric learning, supervision appears as a natural choice. In most cases, two examples are defined as "similar" if they belong to the same class. But then, one wonders what is the difference from supervised classification.

Unsupervised metric learning.

Building on our work on searching by manifold similarity, we introduce one of the first *unsupervised* deep metric learning solutions [192]. We treat manifold similarity as a *teacher* model and we use it to learn a representation by a *student* model. The goal is that the ranking obtained by the representation of the student is in agreement with the ranking obtained by manifold similarity in the representation space of the teacher. This approach is equally effective in *fine-grained* classification and *instance-level* search.

Semi-supervised learning.

Obtaining large amounts of training data annotated by humans for every single task is both impractical and error prone. Unsupervised solutions are still inferior to solutions on data laboriously annotated by humans. *Semi-supervised learning* becomes increasingly important

because it can combine limited data carefully labeled by humans with abundant unlabeled data.

Inductive label propagation.

Searching by manifold similarity is a well-known solution for semi-supervised learning, known as *label propagation*: In short, unlabeled examples receive class scores according to average manifold similarity to labeled examples of each class. This is a *transductive* solution, *i.e*. only makes predictions on the training set. We introduce an *inductive* version [193] whereby a deep model is learned according to supervision provided by label propagation. This model can then make predictions on unseen data without accessing the training set.

Few-shot learning.

What if both annotations *and* raw data are scarce? *Few-shot learning*, *i.e*. learning from few examples, is challenging because learning a deep representation undoubtedly needs a lot of data. It is often treated by nearest neighbor classifiers and indeed many solutions rely on metric learning. However, modern benchmarks require generalizing way beyond the few given examples to recognize examples of entirely different appearance. This is harder than standard category-level tasks in terms of within-class variability.

Implicit data augmentation, increased capacity.

We study activation maps for the first time in the context of few-shot learning [261]. Inspired by partial similarity, we introduce a classification loss that implicitly acts as a *data augmentation* mechanism, essentially searching over all possible locations in an image. At the same time, inspired by mechanisms of increasing model capacity in *incremental learning*, we widen the last layers of the model to accommodate for new, task-specific features with reduced risk of overfitting. These solutions are complementary to some extent.

Adversarial examples.

With the impressive performance of deep models and the resulting routine application to safety-critical tasks like driving cars, it becomes imperative to study and understand the sensitivity of these models to their input. *Adversarial examples*, *i.e*., failure cases obtained by imperceptible perturbations on legitimate input images, become extremely important in this respect.

Smooth adversarial examples.

Our *smooth adversarial examples* [505] are an attempt to improve upon the visual quality of adversarial examples in terms of imperceptibility by an image-dependent smoothing process. This work lies at the heart of studying the manifold structure of visual data because natural images are piecewise smooth. *Robustness* to adversarial examples on this manifold is essentially *generalization*. Hence this study is important in learning better representations regardless of the standard threat model of adversarial examples.

BEYOND PART IV summarizes more of our past and present contributions, reflects on our contributions in the present context and consolidates the ideas exposed in this manuscript. It then attempts to draw a road map of ideas that are likely to come.

From representing to exploring and back.

On one hand, improvements in visual representation and matching processes yield improved *search* by visual similarity and improved *discovery of global structure*. On the other hand, improving the quality of global structure yields improved *representations*, assuming access to

the target image collection. In addition, deep parametric models can learn new representations based on the global structure of the data and *generalize* to unseen images.

> *The recurrent theme is that exploring data and learning the representation are mutually beneficial.*

Memory mechanisms.

Learning deep parametric models has undergone a number of challenges before establishing its power. The most recent challenge has been to convince the scientific community in investing the required amounts of data, supervision and computing power. One remaining challenge may be to invest in *storage capacity* and to find *memory mechanisms* translating this capacity to better performance.

We put forth a vision for future research whereby *data becomes a first-class citizen* in visual recognition tasks. Around this vision, we build a number of research directions, all on learning visual representations from data with limited supervision and applying them to visual recognition tasks. The most challenging is *learning while memorizing*, a form of instance-level incremental learning.

1.4 HOW TO READ

Assumed background and structure.

We assume a basic *background* on computer vision and machine learning. Other than that, this manuscript can be read by a non-expert. We include the necessary background to make the text as self-consistent as possible. The text is organized in a modular structure, allowing reading in different ways. Figure 1.1 shows the chapter dependencies. For instance, apart from top-down, one may read PART I, PART II or PART III in any order, as long as a few chapters are read first.

'Lazy way': non-technical synthesis.

It is recommended, at least at first reading, to follow the sequence of chapters indicated in green. Apart from the current introductory chapter, this comprises one *'outline'* chapter at the beginning of each of the first three parts, including context, background, our contributions and the structure of the text; and the two chapters of PART IV, providing a general synthesis and drawing perspectives beyond this work. This content is high-level and non-technical. In fact, it corresponds to the "synthetic document of 30 to 50 pages (excluding attached articles and appendices)" that is recommended by IRISA.

The background material included in the 'outline' chapters is by no means complete; it is selected according to importance, personal preference and relevance to the subjects studied in each part.

Technical chapters and publications.

The remaining chapters assume reading of the 'outline' chapter of the corresponding part. Each of these chapters—except Chapter 9—is essentially a short version of one or two clearly indicated publications; the association of chapters to publications also appears at the end of the corresponding 'outline' chapters. This text is technical but includes enough background to be self-contained. The material is presented in a unified treatment and unified notation to some extent. The intent is that in this way, the reader may get a good understanding of

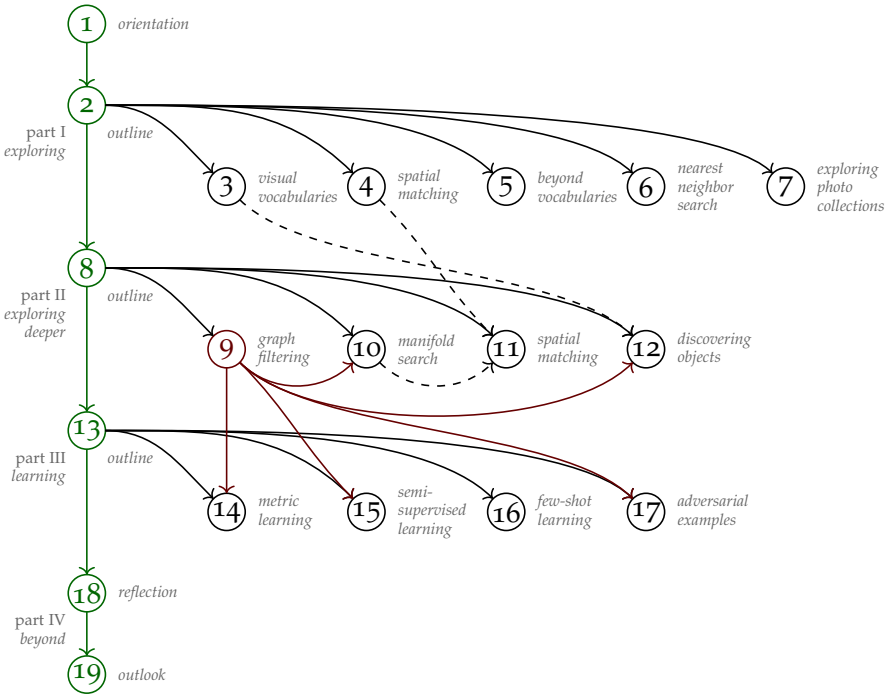


Figure 1.1: *Chapter dependencies*. Solid lines: One chapter is assumed to be read before another. Dashed: One chapter would help to read before another, but it is not necessary. Green: Recommended order of reading. Red: Background on graph filtering.

the methods exposed without having to go through all details of the original publications.

Experiments and related work.

In particular, although these chapters include *experimental results*, these are only a tiny part of the original results and are meant only to illustrate the main achievements of the corresponding methods. Also, these chapters do not include a *related work* section. Instead, the 'outline' chapters of the corresponding part provide a much broader historical and more recent background and context to allow positioning of our contributions in light of the current state of affairs.

Dependencies.

These chapters may in general be read in any order, with the exception of dependencies indicated in Figure 1.1. There is only one strong dependency: Chapter 9 provides background, notation and definitions on *graph filtering* that is required by five other chapters. The remaining dependencies are weak: Chapter 3 presents a method initially introduced for *clustering* that is used by Chapter 12 for *region detection*; Chapters 4 and 11 are both on different aspects of *spatial matching*; in the experiments of Chapter 11 we are using *manifold search* as introduced in Chapter 10. In these cases, reading the chapter indicated as a dependency helps, but is not necessary.

Also, the graph defined in Section 10.2 is used in Chapters 12, 14 and 15.

Margin paragraphs.

Throughout the text, *margin paragraphs* are used to provide

1. the topic discussed in the corresponding paragraph(s);
2. references to other chapters;

3. definitions of notation;
4. examples; or
5. information, explanations or discussion.

In particular, apart from Chapter 9, definitions of notation used in more than one chapters are repeated in each chapter to keep it self-contained. Margin paragraphs in red highlight our contributions.

Part I

EXPLORING

Building on hand-crafted local descriptors and visual vocabularies, we study visual representations and matching processes for exploring visual data, including instance-level visual search and clustering, as well applications to exploring photo collections.

2 OUTLINE

This chapter serves as an outline or road map of Part I. *We present historical and more recent background on hand-crafted local features and descriptors as well as visual vocabularies developed in the 2000s. In this context, we position our own contributions developed in 2008-2014. Building on such methods, our work addresses visual representations and matching processes for exploring visual data, including instance-level visual search and clustering, as well as applications to exploring photo collections. We discuss common evaluation measures and outline the structure of* Part I *in terms of methods, key publications and chapters.*

2.1 CONTEXT

The work of Lowe known as *Scale-Invariant Feature Transform* (SIFT) [274, 276] in 1999 is arguably a landmark of computer vision research in the 2000s. Even if none of the ideas is entirely new, Lowe puts together three elements that make visual recognition "really" work under occlusion, clutter, lighting and viewpoint changes:

1. a sparse local feature *detector*, building on *scale-space* theory [484] and Lindeberg's *automatic scale selection* [264];
2. a local *descriptor* in the form of orientation histograms as an approximation of Gabor filter banks [88, 447], followed by spatial pooling [285, 324]; and
3. a *spatial matching* process using a variant of the *Generalized Hough Transform* (GHT) [28] where each feature correspondence between two views casts a single vote.

Considering local descriptors as a first layer in a visual representation pipeline, the *Bag of Words* (BoW) model [83, 413] builds a second layer that simplifies and accelerates several visual recognition tasks. With few adaptations in terms of local feature sampling and the size of visual vocabularies, this pipeline is applied to tasks including both *instance-level* (different views of the same object or scene) and *category-level* (different objects or scenes in the same category).

Our contributions.

In this context, Part I presents part of our work carried out in the period 2008-2014, which addresses visual representations and matching processes for exploring visual data, including instance-level visual search and clustering. Our contributions consist of:

1. improvements in the pipeline, including the construction of large scale visual vocabularies [18], spatial matching for geometry verification [20, 433], representations beyond vocabularies [436, 437] and nearest neighbor search [213]; and

2. applications to exploring photo collections, including location recognition, landmark recognition, automatic discovery and visualization of photos depicting the same scene [19, 215].

Importantly, as a result of discovering photos depicting the same scene, we improve the representation of each photo itself.

2.2 BACKGROUND AND CONTRIBUTIONS

Why local features?

LOCAL FEATURES *Dense image registration*, studied by Lucas and Kanade in 1981 [278], is a low-level vision task, where for each location in an image, the objective is to find a displacement with respect to another reference image. It is appropriate for small displacements, *e.g.* in stereopsis or optical flow. However, not all locations are equally good for estimating displacement, which is known as the *aperture problem*. By formulating registration as a least squares problem, Tomasi and Kanade find in 1991 [440] that it is more reliable to detect and track a sparse set of *point features*. The criteria are similar to *corner detection*, as studied by Harris and Stephens in 1988 [156].

In *wide-baseline matching*, studied by Pritchett and Zisserman in 1998 [343], an object or scene is seen from two arbitrary viewpoints, so every part of one view may be seen anywhere in the other. Objects may be largely occluded or seen at very different scales. In this more challenging problem, it makes even more sense to focus on a sparse set of *local features*. In establishing correspondences between the two views, it also makes sense to attach to local features more geometric information like scale and orientation, as well appearance information like visual descriptors.

How to detect them?

The *Laplacian of Gaussian* (LoG) kernel plays a central role in the theory of *edge detection* by Marr and Hildredth in 1980 [288]. It serves as a model of retinal cells, with a hypothesized biological implementation as a *Difference of Gaussians* (DoG) having a scale ratio close to 1.6. Following studies by Witkin in 1987 [484], Lindeberg shows in 1994 [266] that the Gaussian kernel and its derivatives are the only possible smoothing kernels for *scale-space* analysis and uses them in 1998 [264] to detect local features with *automatic scale selection*. In this context, the LoG is an idealized blob-like pattern that is searched for exhaustively over all possible locations and scales.

The SIFT detector by Lowe in 1999 [274, 276] is such an LoG blob detector, approximated by a DoG with a scale ratio of exactly 1.6. SIFT features are equipped with coordinates, scale and orientation. On one hand, further approximations target faster implementations, for instance *Speeded-up Robust Features* (SURF) [31] approximate Gaussian derivatives by *integral images* [462]. On the other hand, improved *repeatability* under affine transforms is sought, giving rise to *affine local features* [296]. Based on iterative affine shape adaptation [265], the *Hessian affine* detector [294] becomes the most popular.

VISUAL DESCRIPTORS Following the experiments by Hubel and Wiesel in 1959 [185] and subsequent neuroscience results [287, 307], *2d Gabor filters* are introduced by Daugman in 1985 [87] and confirmed in 1987 [210] as a mathematical model for the receptive fields of simple cortical cells. *Gabor filter banks* are used thereafter for texture analysis [88, 447] and generic visual representation [285, 324].

Gabor filters and local scale/orientation histograms.

Because Gabor filters are orientation-sensitive, a filter bank acts as an *encoding*, effectively assigning each location in an image to an orientation at that location (and similarly for scale). When followed by spatial *pooling* (or *aggregation*), this is roughly equivalent to a histogram over orientations (or scales). Pooling, hence histograms, can be *global* over the image [285, 324] or *regional* [323].

The SIFT descriptor by Lowe in 1999 [274, 276] is essentially a shortcut, where the orientation at each location is determined directly by the local gradient and histograms are computed directly, without Gabor filters. Despite several competitors [295], the SIFT descriptor dominates in most vision tasks until learned representations take over. In modern terminology, all such histogram-like descriptors are called *hand-crafted* or *shallow* in retrospect (as opposed to *learned* or *deep*) and are only the *first layer* in a visual representation pipeline.

Instance-level and category-level tasks.

Local descriptors, *i.e.* descriptors computed at patches aligned with local features at the appropriate scale and orientation, provide a powerful mechanism for establishing correspondences between two views of the same object or scene. This mechanism is robust to occlusion, clutter, lighting and viewpoint changes, and opens the door to solving several *instance-level* tasks including object recognition, retrieval and 3d reconstruction.

The *Histogram of Oriented Gradients* (HOG) [86] is another variant, where the descriptor is computed densely over a grid. As confirmed independently using visual vocabularies [113, 212], this choice is superior to using a local feature detector in *category-level* tasks, including scene categorization and object detection. The situation is the same in concurrent models inspired by the visual cortex [311, 395], which also maintain filter banks like modern architectures.

> *We focus on instance-level tasks, adopting local features and hand-crafted descriptors in* PART I *and learned global, regional or local representations in* PART II*. Then,* PART III *addresses learning representations for both instance-level and category-level tasks.*

What are they?

VISUAL VOCABULARIES In modern terminology, visual vocabularies (or *codebooks*) are used to define a *second layer* in a visual representation pipeline on top of, usually hand-crafted, visual descriptors. Learned without supervision by vector quantization of a collection of visual descriptors, a visual vocabulary acts as a set of bins over which histograms may be computed in the visual descriptor space. An image is then represented by *encoding* (assigning) its visual descriptors

to bins, called *visual words*, and then *pooling* (aggregating) them into a global histogram vector, called *Bag of Words* (BoW).

Why do we need them?

The term BoW originates in a linguistic context [157], where words are naturally discrete. In the first layer of visual representation, histograms over scales and orientations are achieved by scalar quantization, resulting in a 2d grid of bins. In fact, the Gabor filter bank corresponds to a log-polar sampling grid in the frequency domain [88]. By contrast, the visual descriptors handled in the second layer are *high-dimensional*. Histograms over this high-dimensional vector space would be intractable without vector quantization. We are essentially learning the distribution of visual descriptors of natural images, such that we are not left with countless empty bins.

E.g. *18d for a Gabor filter bank of 3 scales and 6 orientations or 128d for a SIFT descriptor.*

An early form of visual vocabulary on Gabor filter bank responses sampled densely over a single image is suggested by Daugman in 1988 [88], applied to texture segmentation. The same idea with the same application is given the name *textons* by Malik *et al*. in 1999 [282], defining precisely what was described by Julesz in 1981 [211] as "basic elements of pre-attentive human texture perception."

Vocabularies on local descriptors follow, applied to category-level [83] and instance-level [413] recognition. The former are small (*coarse*) to compensate for in-class variations, resulting in compact representations that can be used *e.g*. by a *Support Vector Machine* (SVM) for object categorization. Descriptors sampled densely over a grid eventually replace local descriptors in this case [113, 212]. The latter are large (*fine*) to maintain discriminative power, resulting in sparse image representations. Borrowing ideas from text retrieval including *Term Frequency* (TF)-*Inverse Document Frequency* (IDF) [26] and *inverted files* [485], instance-level search becomes very efficient [413].

E.g. *few thousands [124].*

E.g. *even a million [341].*

Vocabularies for category-level tasks commonly use a *Gaussian Mixture Model* (GMM) [110, 337], learned by the *Expectation-Maximization* (EM) algorithm [91]. For instance-level tasks, learning at a scale of *e.g*. one million visual words is mostly constrained to variants of k-means [143]. Our *Approximate Gaussian Mixture* (AGM) [18] is the first attempt to apply a GMM at this scale, employing *Approximate Nearest Neighbor* (ANN) search in EM. The size of the vocabulary is dynamically estimated and AGM only needs to run once. This is extremely important because for each size k of vocabulary conventionally tested, one needs to not only learn the vocabulary, but also re-index a very large collection of images to evaluate performance.

Our Approximate Gaussian Mixtures *(AGMs) [18].*

From 'bags' back to local features.

SPATIAL MATCHING The term 'bag' in BoW refers to dropping all geometrical information from local features and keeping only appearance information as represented in visual descriptors. This makes visual representation completely invariant to geometric transformations, but naturally drops discriminative power as well.

Establishing correspondences between two views of the same object or scene can be much more informative than the similarity of two histograms. However, correspondences found by pairwise matching of visual descriptors are noisy, even more so when descriptors

are quantized. True correspondences, called *inliers*, may be found by measuring consistency with a rigid geometric transformation model having a given number of *Degrees of Freedom* (DoF). Unfortunately, establishing a transformation in turn depends on the correspondences and is challenging when the inliers are only a small percentage.

E.g., 2 DoF for translation, 4 for similarity, 6 for affine transformation.

RANdom SAmple Consensus (RANSAC), introduced by Fischler and Bolles in 1981 [117], is a robust estimator that iteratively evaluates transformation hypotheses by sampling the minimum required number of correspondences depending on the DoF, fitting the model and measuring inliers. Alternatively, the *Generalized Hough Transform* (GHT), introduced by Ballard in 1981 [28], finds promising transformations by casting a set of votes for each correspondence and performing mode seeking in the transformation space. Both are problematic when the number of DoF is large and RANSAC is even more so when the percentage of inliers is small.

Single correspondence hypotheses.

The easiest case is when a transformation hypothesis is possible from a single correspondence (for RANSAC), or there is a single vote cast for every correspondence (for GHT). For point correspondences (*e.g.* corners), this happens for 2-DoF translation only. This used *e.g.* for category-level recognition with vocabularies and GHT [254]. More interestingly, it also happens for more DoF when local features are equipped with additional geometric information. For instance, SIFT (resp. Hessian affine) feature correspondences give rise to transformations of 4 (resp. 5) DoF, which is used for instance-level recognition with GHT by Lowe [274, 276] (resp. with RANSAC by Philbin *et al.* [341]). Both solutions require inlier verification and are therefore *quadratic* in the number of correspondences.

In instance-level image retrieval, a query image needs to be matched against millions others. Typically, this happens by a first stage of *filtering* according to BoW using inverted files, followed by a second stage of more expensive *geometry verification*, only on a short list of top-ranking images of the first stage. To accelerate the latter, our *Hough Pyramid Matching* (HPM) [20, 433] uses GHT alone, without inlier verification, and is *linear* in the number of correspondences, reaching thousands of image matches per second. HPM accommodates for multiple surfaces or *flexible* objects, improving accuracy over any rigid motion model, including homography.

Our Hough Pyramid Matching *(HPM) [20, 433].*

BEYOND VOCABULARIES The BoW model is praised for its efficiency, but vector quantization is found detrimental in category-level tasks [46], because the most informative descriptors are the most infrequent, leading to high quantization error. Ideally, we would like to approximate the optimal descriptor correspondences. The *Pyramid Match Kernel* (PMK) [141, 142] attempts this by using a hierarchical partition of the descriptor space. However, using the true descriptors is superior [46].

From 'words' back to descriptors.

Similarly, in instance-level retrieval, there is a progression from hierarchical vocabularies [316] to flat, fine vocabularies [341], then to *Multiple Assignment* (MA) of descriptors to visual words [342], then

to even finer vocabularies equipped with *visual synonyms*, giving rise to visual words of arbitrary shape [298]. However, if one can afford the required space, using actual compressed descriptors, for instance with *Hamming Embedding* (HE) [202], is superior.

Global pooling.

At the same time, pooling (aggregation) mechanisms beyond histograms are sought. The idea is to pool *vectors* rather scalars (frequencies) for each visual word, resulting in a very high-dimensional representation, which is followed by dimension reduction by *Principal Component Analysis* (PCA). Such models are *Fisher vectors* [85, 109, 339] and the *Vector of Locally Aggregated Descriptors* (VLAD) [204, 205], applied to both instance- and category-level tasks. However, performance in instance-level tasks with small vocabularies is poor.

We are the first to explore matching mechanisms like HE and pooling mechanisms VLAD with large vocabularies for instance-level retrieval, developing a common model that incorporates both as special cases. In doing so, we borrow ideas from both models to introduce the *Aggregated Selective Match Kernel* (ASMK) [436, 437]. This model achieves the last known state of the art before the advent of deep learning. It is still used by current state of the art methods using modern learned representations [350, 427].

Our Aggregated Selective Match Kernel *(ASMK) [436, 437].*

NEAREST NEIGHBOR SEARCH With the use of encoded descriptors, the problem of image retrieval boils down to ANN search in high dimensions, where points (descriptors) are compressed. One of the most successful search methods in the compressed domain is *Product Quantization* (PQ) [200], which decomposes the space into a Cartesian product of subspaces and independently applies vector quantization to each. An improvement is *Optimized Product Quantization* (OPQ) [122], which additionally optimizes subspace decomposition via an orthogonal transformation.

However, the distribution of images in the descriptor space is arbitrary. We cannot expect to fit this distribution with a grid of points, not even a rotated one. Our *Locally Optimized Product Quantization* (LOPQ) [213] applies the above ideas locally, finding an optimal subspace decomposition, rotation and quantizer for each region in the descriptor space. Like local PCA [216], this model can fit distributions exhibiting *manifold structure*.

Our Locally Optimized Product Quantization *(LOPQ) [213].*

LOPQ remains for several years the state of art on a challenging dataset of one billion SIFT descriptors. In 2017, using a CNN image representation, Yahoo! Research chooses LOPQ to index and provide a *similar image search*[1] functionality on its entire Flickr[2] collection, consisting of more than 10 billion images.

EXPLORING PHOTO COLLECTIONS An interesting application of methods discussed so far, is to datasets originating from *community photo collections*, in particular depicting outdoor urban scenes. Build-

1 https://yahooresearch.tumblr.com/post/158115871236/introducing-similarity-search-at-flick

2 https://flickr.com/

ings have typically a level of detail for local descriptors to be discriminative enough and non-deformable structure such that different views are related by a rigid transformation model. Such datasets are frequently accompanied by additional, non-visual information, like tags or geo-tags. In many cases they focus on *landmarks* [341, 342], in which case they exhibit high redundancy.

Applications.

Given a query image, search by visual similarity against a dataset accompanied by geo-tags gives rise to *location recognition* [418, 510], which can work reliably up to city scale [387]. Early efforts on *landmark recognition* [82, 220] are mostly based on metadata, with the contribution of visual features being small. *Visual clustering* [76, 408] is a popular way to discover landmark images without supervision. It can be made more efficient by performing geographic clustering first [120, 349], assuming geo-tags. *Structure-from-Motion* (SfM) allows for vision-based reconstruction and navigation of a 3d scene [414], which can again work at city scale [4].

Most such applications are limited to clusters of popular images like landmarks and, even if they use efficient indexing by inverted files, they do not use the mined information to improve the indexing itself. We rather perform visual clustering by *Kernel Vector Quantization* (KVQ) [432], guaranteeing that isolated images are preserved, and that all images in a cluster share at least a rigid object or surface with one particular reference image. By projecting them on that image plane, we then construct a *scene map* [19] for each cluster. Now, indexing scene maps instead of individual images not only saves space, but increases recall performance as well.

Our scene maps *[19].*

Based on these ideas, our application *Visual Image Retrieval and Localization* (VIRaL)[3] [215] supports automated *location estimation* and geo-tagging, *recognition* of landmarks and points of interest, and *visualization* of photo clusters and tourist paths on an online map.

Our application VIRaL [215], which has been online since 2008.

EVALUATION Several tasks including retrieval, metric learning and object detection are evaluated by *mean Average Precision* (mAP). Another evaluation measure for retrieval is *mean Precision* (mP) at k. Because most chapters in PART I and PART II, as well as Chapter 14 in PART III, experiment on image retrieval, we define both here.

Average Precision (AP) [286] evaluates *ranking*: Given a ranked list of items, it evaluates to what extent a set of *positive* items are ranked first. Informally, AP is the area under the precision-recall curve, where *precision* (resp. *recall*) at every position in the list is the ratio of retrieved positive to retrieved (resp. positive) items. Then, mAP is the average AP over a number of queries (for retrieval or metric learning) or classes (for object detection). This means that for every query (or class), we need to know a corresponding set of positive items.

Formally, let $X := \{x_1, \dots, x_n\}$ be an ordered list of items and $X_k := \{x_1, \dots, x_k\}$ for $k \in [n]$, with $|X_k| = k$. Let also $P \subseteq X$ be a

We write $[n] = \{1, \dots, n\}$ *for* $n \in \mathbb{N}$.

3 http://viral.image.ntua.gr/

subset of *positive* elements of X; the remaining elements in $X \setminus P$ are *negative*. For $k \in [n]$, *precision at* k of X relative to P is defined as

$$p_k(X;P) := \frac{|X_k \cap P|}{|X_k|} \tag{2.1}$$

$$= \frac{1}{k} \sum_{i=1}^{k} \mathbb{1}_P(x_i). \tag{2.2}$$

$\mathbb{1}_A : X \to \{0,1\}$ *is the* indicator function *of* $A \subseteq X$*: for* $x \in X$, $\mathbb{1}_A(x)$ *is* 1 *if* $x \in A$ *and* 0 *otherwise.*

Then, the AP of X relative to P is defined as

$$\mathrm{AP}(X;P) := \frac{1}{|P|} \sum_{k=1}^{n} \mathbb{1}_P(x_k) p_k(X;P). \tag{2.3}$$

AP satisfies $0 \leq \mathrm{AP}(X;P) \leq 1$, with $\mathrm{AP}(X;P) = 1$ iff $X_{|P|} = P$, that is, all elements of P are ranked before elements of $X \setminus P$.

Mean Precision (mP) at k, denoted by mP@k, is the average, over a number of queries, of precision at k. This measures the proportion of top k retrieved items that are positive, for small k. For instance, our *Revisited Oxford and Paris* (RevOP) image retrieval benchmark [350] includes mP@10 as a standard evaluation measure along with mAP.

2.3 STRUCTURE

Chapter 3 deals with building large-scale visual vocabularies for BoW models, based on the *Gaussian Mixture Model* (GMM). It presents our *Expanding Gaussian Mixture* (EGM) model and its approximate version *Approximate Gaussian Mixture* (AGM) [18].

Chapter 4 addresses spatial matching for geometry verification, based on the *Generalized Hough Transform* (GHT). It presents our *Hough Pyramid Matching* (HPM) [20, 433].

Chapter 5 revisits visual representations and discusses solutions beyond BoW, based on local descriptors. It builds a common framework for previous models and introduces our *Aggregated Selective Match Kernel* (ASMK) [436, 437].

Chapter 6 goes beyond visual search to the more generic problem of *Approximate Nearest Neighbor* (ANN) search, based on quantization. It discusses our *Locally Optimized Product Quantization* (LOPQ) [213].

Finally, Chapter 7 addresses the more challenging problem of exploring photo collections to automatically discover groups of photos depicting the same scene. It presents our *scene maps* representation [19] and discusses *Visual Image Retrieval and Localization* (VIRaL), our online application for location and landmark recognition [215].

3 VISUAL VOCABULARIES

We introduce a clustering method that combines the flexibility of Gaussian mixtures with the scaling properties needed for visual vocabularies in image retrieval [18]. It is a variant of Expectation-Maximization *(EM) that dynamically estimates the number of components. We employ approximate nearest neighbor search to speed-up the E-step and exploit its iterative nature to make search incremental, boosting speed and precision.*

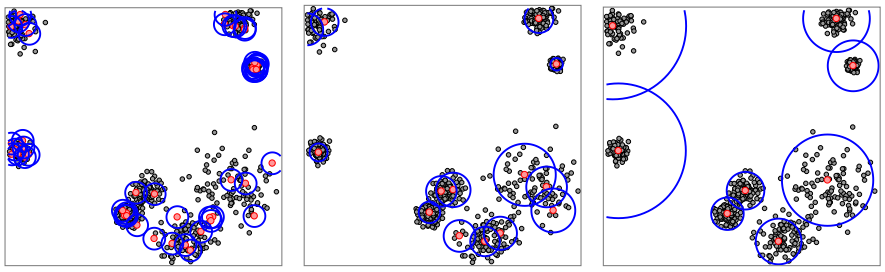
3.1 INTRODUCTION

The *k-means* algorithm is one of the most popular in learning *visual vocabularies*, or *codebooks*, needed by the *Bag of Words* (BoW) model. For image retrieval, fine vocabularies are needed, *e.g.* 10^6 visual words. Clustering options are limited at this scale, with the most common being variants like *Approximate k-Means* (AKM) [341] and *Hierarchical k-Means* (HKM) [316].

In addition to position, GMM models cluster population and shape.

The *Gaussian Mixture Model* (GMM), along with *Expectation-Maximization* (EM) [91] learning, is a generalization of k-means that has been applied to class-level recognition [336]. It assumes pairwise 'interaction' of all points with all clusters and is slower to converge. By contrast, a point is assigned to the nearest cluster via *Approximate Nearest Neighbor* (ANN) search in AKM [341].

In *Robust Approximate k-Means* (RAKM) [255], the nearest neighbor in one iteration is re-used in the next, with less effort being spent for new neighbor search. This motivates us to keep a larger number m of nearest neighbors. Thus, enough information is available for an *Approximate Gaussian Mixture* (AGM) [18] model, whereby each data point 'interacts' only with the m nearest clusters.



iteration 0: 50 clusters iteration 1: 15 clusters iteration 3: 8 clusters

Figure 3.1: Estimating the number, population, position and extent of clusters on an 8-mode 2d Gaussian mixture sampled at 800 points, *in just 3 iterations* (iteration 2 not shown). Red circles: Cluster centers. Blue: Two standard deviations. Clusters are initialized on 50 data points drawn uniformly at random.

As shown in Figure 3.1, we compute the overlap of neighboring clusters and *purge* the ones that appear redundant, after each EM iteration. This algorithm, *Expanding Gaussian Mixture* (EGM) [18], can *dynamically estimate* the number of clusters by starting with a large number and purging along the way. Focusing on spherical Gaussians, we apply its approximate version, AGM, to large scale visual vocabulary learning for image retrieval.

Contrary to typical GMM, indexed descriptors are assigned only to the nearest visual word.

3.2 GAUSSIAN MIXTURES

The density $p(\mathbf{x})$ of a Gaussian mixture distribution is a convex combination of k d-dimensional normal densities or *components*,

$$p(\mathbf{x}) := \sum_{j=1}^{k} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \tag{3.1}$$

for $\mathbf{x} \in \mathbb{R}^d$, where $\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j$ are the *mixing coefficient*, *mean* and *covariance matrix* respectively of the j-th component. Interpreting π_j as the prior probability $p(j)$ of component j, quantity

$$\gamma_j(\mathbf{x}) \leftarrow \frac{\pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{\ell=1}^{k} \pi_\ell \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell)}, \tag{3.2}$$

expresses the posterior probability $p(j|\mathbf{x})$ given observation $\mathbf{x} \in \mathbb{R}^d$. We say that $\gamma_j(\mathbf{x})$ is the *responsibility of j for $\mathbf{x}$*. Now, given a set of i.i.d. observations $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, the *Maximum Likelihood* (ML) estimate for the parameters of each component $j \in [k]$ is [44]

We write $[n] = \{1, \ldots, n\}$ for $n \in \mathbb{N}$.

$$\pi_j \leftarrow \frac{n_j}{n} \tag{3.3}$$

$$\boldsymbol{\mu}_j \leftarrow \frac{1}{n_j} \sum_{i=1}^{n} \gamma_{ij} \mathbf{x}_i \tag{3.4}$$

$$\boldsymbol{\Sigma}_j \leftarrow \frac{1}{n_j} \sum_{i=1}^{n} \gamma_{ij} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^\top, \tag{3.5}$$

where $\gamma_{ij} \leftarrow \gamma_j(\mathbf{x}_i)$ for $i \in [n]$, and $n_j \leftarrow \sum_{i=1}^{n} \gamma_{ij}$ is the *effective number* of points assigned to component j. The EM algorithm is an iterative process: Given an initial set of parameters, compute responsibilities γ_{ij} according to (3.2) (*E-step*); then, re-estimate parameters according to (3.3)-(3.5), keeping responsibilities fixed (*M-step*).

This is still more flexible than k-means, and efficient because σ_j is a scalar.

Here we focus on the particular case of *spherical* (*isotropic*) Gaussians, with covariance matrix $\boldsymbol{\Sigma}_j = \sigma_j^2 \mathbf{I}$. In this case, update equation (3.5) reduces to

$$\sigma_j^2 \leftarrow \frac{1}{d n_j} \sum_{i=1}^{n} \gamma_{ij} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2. \tag{3.6}$$

3.3 EXPANDING GAUSSIAN MIXTURES

Determining the number of components is more critical in GMM than in k-means due to overlapping.

PURGING We devise a method of *purging* components according to an *overlap* measure. Purging is *dynamic* in the sense that it takes place during parameter learning. This idea introduces a *P-step* in EM, to be applied after the E- and M-steps in every iteration.

A component j can be represented by the function p_j determining its contribution to the GMM distribution (3.1),

$$p_j(\mathbf{x}) := \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \tag{3.7}$$

p_j is not normalized unless $\pi_j = 1$.

for $j \in [k]$, $\mathbf{x} \in \mathbb{R}^d$. Now, let

$$\langle p, q \rangle := \int p(\mathbf{x}) q(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{3.8}$$

be the L^2 *inner product* of real-valued, square-integrable functions p, q, where the integral is over $\mathbb{R}^d$. The corresponding L^2 *norm* of function p is given by $\|p\| := \sqrt{\langle p, p \rangle}$. When p, q are normal distributions with $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{a}, \mathbf{A})$ and $q(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{b}, \mathbf{B})$ for $\mathbf{x} \in \mathbb{R}^d$, the integral in (3.8) can be evaluated in *closed form* [18]

$$\langle p, q \rangle = \mathcal{N}(\mathbf{a}|\mathbf{b}, \mathbf{A} + \mathbf{B}). \tag{3.9}$$

Hence, given components represented by p_j, p_ℓ, their *overlap*, as measured by inner product, is

$$\langle p_j, p_\ell \rangle = \pi_j \pi_\ell \mathcal{N}(\boldsymbol{\mu}_j|\boldsymbol{\mu}_\ell, (\sigma_j^2 + \sigma_\ell^2)\mathbf{I}) \tag{3.10}$$

under the spherical Gaussian model, while $\|p_j\|^2 = \pi_j^2 (4\pi\sigma_j^2)^{-d/2}$. Now, if function q represents any component or cluster, (3.10) motivates generalizing (3.2) to

$$\hat{\gamma}_j(q) := \frac{\langle q, p_j \rangle}{\sum_{\ell=1}^k \langle q, p_\ell \rangle}, \tag{3.11}$$

Function p_j is treated here as a generalized data point.

so that $\hat{\gamma}_{j\ell} \leftarrow \hat{\gamma}_\ell(p_j) \in [0, 1]$ is the *generalized responsibility* of component ℓ for *component* j. As illustrated in Figure 3.2, (3.11) is reduced to (3.2) when q collapses to a Dirac delta function, effectively *sampling* component functions p_j.

According to (3.11), $\hat{\gamma}_{jj}$ is the responsibility of component j for *itself*. More generally, given a set K of components with $j \notin K$, let

$$\rho_{j,K} := \frac{\hat{\gamma}_{jj}}{\hat{\gamma}_{jj} + \sum_{\ell \in K} \hat{\gamma}_{j\ell}} = \frac{\|p_j\|^2}{\|p_j\|^2 + \sum_{\ell \in K} \langle p_j, p_\ell \rangle} \in [0, 1] \tag{3.12}$$

be the responsibility of component j for itself *relative to* K. If $\rho_{j,K}$ is large, j can 'explain' itself better than set K *as a whole*; otherwise it is redundant. So, if K holds the components we have decided to *keep* so far, it makes sense to purge component j if $\rho_{j,K}$ drops below *overlap threshold* $\tau \in [0, 1]$, in which case we say that j *overlaps* with K.

$\tau \geq \frac{1}{2}$ ensures we cannot 'keep' two identical components.

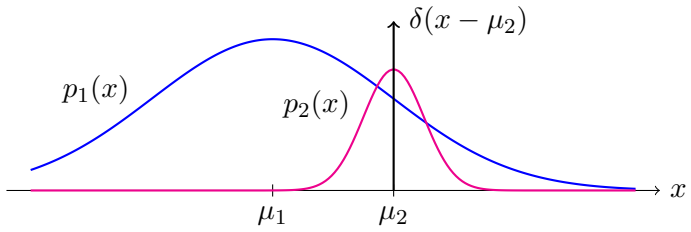


Figure 3.2: '*Sampling*' a large component $p_1(x) = \pi_1 \mathcal{N}(x|\mu_1, \sigma_1^2)$ through a smaller one, $p_2(x) = \pi_2 \mathcal{N}(x|\mu_2, \sigma_2^2)$, in one dimension. When the latter is reduced to a single point, $p_2(x)$ collapses to $\delta(x - \mu_2)$, and inner product $\langle p_1, p_2 \rangle$ is reduced to $p_1(\mu_2)$.

Algorithm 3.1: Component purging (P-step)

input : set of components C at current iteration
output: updated set of components C, after purging

$K \leftarrow \emptyset$ // set of components to keep
sort C such that $j < \ell \rightarrow \pi_j \geq \pi_\ell$ for $j, \ell \in C$ // re-order components
foreach $j \in C$ **do** // ... in descending order of π_j
  if $\rho_{j,K} \geq \tau$ **then** // compute $\rho_{j,K}$ by (3.12)
    $K \leftarrow K \cup j$ // keep j if it does not overlap with K
$C \leftarrow K$ // updated components

This gives rise to the *P-step* is outlined in Algorithm 3.1. We choose to process components in descending order of mixing coefficients, starting from the most populated cluster, which we always keep. We perform the P-step right after E- and M-steps in each EM iteration. We also constrain the E- and M-steps to components in C. Now the number of components $k \leftarrow |C|$ *decreases* in each iteration.

This behavior resembles agglomerative clustering *[159].*

EXPANDING When a component, say j, is purged, data points that were better 'explained' by j prior to purging will have to be assigned to neighboring components that remain. These components will then have to *expand* and cover the space populated by such points. Towards this goal, we modify (3.5) to *overestimate* the extent of each component as much as this does not overlap with its neighboring components. Details can be found in [18]; here we only observe the 'space-filling' behavior of the two clusters on the left in Figure 3.1.

Components will then tend to fill in as much empty space as possible.

INITIALIZING AND TERMINATING We initialize with *all* points as component centers, that is, $k \leftarrow n$. *Mixing coefficients* are uniform initially. *Standard deviations* are initialized to the distance of the nearest neighbor, found approximately. Convergence is never reached in our experiments. What is important is to measure the performance of the resulting vocabulary in the retrieval task *vs.* required processing.

Using ANN search, this choice is still efficient; by the second iteration, few clusters survive anyway.

3.4 APPROXIMATE GAUSSIAN MIXTURES

Counting d-dimensional vector operations, the complexity of the E- and M-steps $O(nk)$, where $k \leq n$ is the current number of components, and the complexity of the P-step (Algorithm 3.1) is $O(k^2)$. This is not practical for large k, especially in the order of n. Similarly to [341], the approximate version of our algorithm involves indexing the entire set of clusters C according to their center $\boldsymbol{\mu}_j$ and performing an ANN query for each data point $\mathbf{x}_i$, prior to the E-step. For typical tree-based ANN search methods, the former is $O(k \log k)$ and the latter $O(n \log k)$. For a query point $\mathbf{x} \in \mathbb{R}^d$, distances to cluster centers are effectively replaced by metric

$$d_m(\mathbf{x}, \boldsymbol{\mu}_j) := \begin{cases} \|\mathbf{x} - \boldsymbol{\mu}_j\|, & \text{if } j \in \mathrm{NN}_m(\mathbf{x}) \\ 0, & \text{otherwise,} \end{cases} \tag{3.13}$$

where $\mathrm{NN}_m(\mathbf{x}) \subseteq C$ denotes the approximate m-nearest neighbors of $\mathbf{x}$. The overall complexity per iteration is then $O(n \log k)$.

Now, similarly to [255], we not only use approximate search to speed up clustering, but we also exploit the iterative nature of the clustering algorithm to enhance the search process itself. To this end, we maintain a list of the m *best neighbors* $\mathcal{B}(\mathbf{x}_i)$ found so far for each data point $\mathbf{x}_i$, and re-use it across iterations. This results in an *incremental m-nearest neighbors* algorithm [18]. It is an *N-step*, to be performed prior to the E-step. As in [255], the rationale is that by keeping track of the best neighbors found so far, we may significantly reduce the effort spent in searching for new ones.

This generalizes RAKM [255], which restricts to $m = 1$ and k-means only.

3.5 EXPERIMENTS

SETUP We compare AGM against AKM [341] and RAKM [255] on large scale vocabulary learning for image retrieval. We use *Oxford buildings*[1] [341] and *world cities*[2] [433] datasets. The distractor set of *world cities* consists of 2M images; we use the first one million as distractors. We use SURF [31] features and descriptors, of dimensionality 64. We learn vocabularies from 6.5M descriptors of an independent set of 15k images depicting urban scenes. At learning, we use *Fast Library for ANN* (FLANN) [309] for all methods. We assign database descriptors to only one visual word. We apply MA [342] on the query side only as in [203], keeping the first 1, 3, or 5 neighbors. We measure mAP.

RESULTS We first choose the best competing method for up to 20k distractors as shown in Table 3.1. The best size for RAKM is 550k. AKM is more or less equivalent; their difference is in speed. AGM is slightly better with its vocabulary size 857k being *automatically inferred*. We

1 http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/
2 http://image.ntua.gr/iva/datasets/wc/

METHOD			RAKM			AKM	AGM
Vocabulary	350k	500k	550k	600k	700k	550k	857k
No distractors	0.471	0.479	**0.486**	0.485	0.476	0.485	**0.492**
20k distractors	0.439	0.440	**0.448**	0.441	0.437	0.447	**0.459**
1M distractors	—	—	**0.250**	—	—	—	**0.280**

Table 3.1: mAP comparisons for vocabularies of different sizes on *Oxford Buildings* with a varying number of distractors, using $m = 100$, 40 iterations for AKM/RAKM, and 15 for AGM.

then extend to 1M *distractors* for the RAKM 550k and AGM 857k vocabularies. AGM maintains a clear overhead.

3.6 DISCUSSION

The main parameter is $\tau = 0.55$.

We manage to get competitive performance on large-scale image retrieval with a set of parameters that work even on our very first two-dimensional example. In most alternatives one needs to tune at least the vocabulary size. Even with spherical components, the added flexibility of Gaussian mixtures appears to boost discriminative power. Yet, learning is as fast as approximate k-means, both in terms of EM iterations and underlying vector operations. Our solution appears to avoid both singularities and overlapping components that are inherent in ML estimation of GMM.

More can be found at our project home page[3], including software.

3 http://image.ntua.gr/iva/research/agm/

4 SPATIAL MATCHING

We develop a simple spatial matching model inspired by Hough voting in the transformation space, where votes arise from single feature correspondences. Using a histogram pyramid, we effectively compute pair-wise affinities of correspondences without ever enumerating all pairs. Our Hough pyramid matching *[20, 433] algorithm is linear in the number of correspondences and allows for multiple matching surfaces or non-rigid objects.*

4.1 INTRODUCTION

Re-ranking is linear in the number of images to match, so its speed is crucial.

Despite the success of the BoW model [413], spatial matching is still needed to boost image retrieval performance. A second stage of *spatial verification* and *geometry re-ranking* is the *de facto* solution of choice, where approximations of RANSAC [117] dominate.

Exploiting the local shape of features (*e.g.* local scale or orientation), it is either possible to generate RANSAC hypotheses by single correspondences [341], or to see correspondences as Hough votes in a transformation space [276]. In the former case one still has to count inliers, so the process is quadratic in the number of (tentative) correspondences. In the latter, voting is linear in the number of correspondences but further inlier count appears unavoidable.

We develop a relaxed spatial matching model, which applies *pyramid matching* [141] to the transformation space. Using local feature shape to generate votes, it is *invariant* to similarity transformations, free of inlier-count verification and *linear* in the number of correspondences. It imposes *one-to-one* mapping and is *flexible*, allowing non-rigid motion and multiple matching surfaces or objects.

Figure 4.1 compares our *Hough Pyramid Matching* (HPM) [20, 433] to *Fast Spatial Matching* (FSM) [341]. Both foreground and background surfaces are matched by HPM, whereas inliers from one surface are only found by FSM. But our major achievement is speed: In a given query time, HPM can re-rank one order of magnitude more images.

4.2 PROBLEM FORMULATION

We restrict discussion to scale and rotation covariant features.

We represent an image by a set P of *local features*. For each feature $p \in P$, its position and local shape is given by the 3×3 matrix

$$F(p) := \begin{bmatrix} M(p) & \mathbf{t}(p) \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{4.1}$$

where $M(p) = \sigma(p)R(p)$ and $\sigma(p), R(p), \mathbf{t}(p)$ stand for isotropic scale, orientation and position, respectively. $R(p)$ is an orthogonal 2×2 matrix with $\det R(p) = 1$, represented by an angle $\theta(p)$.

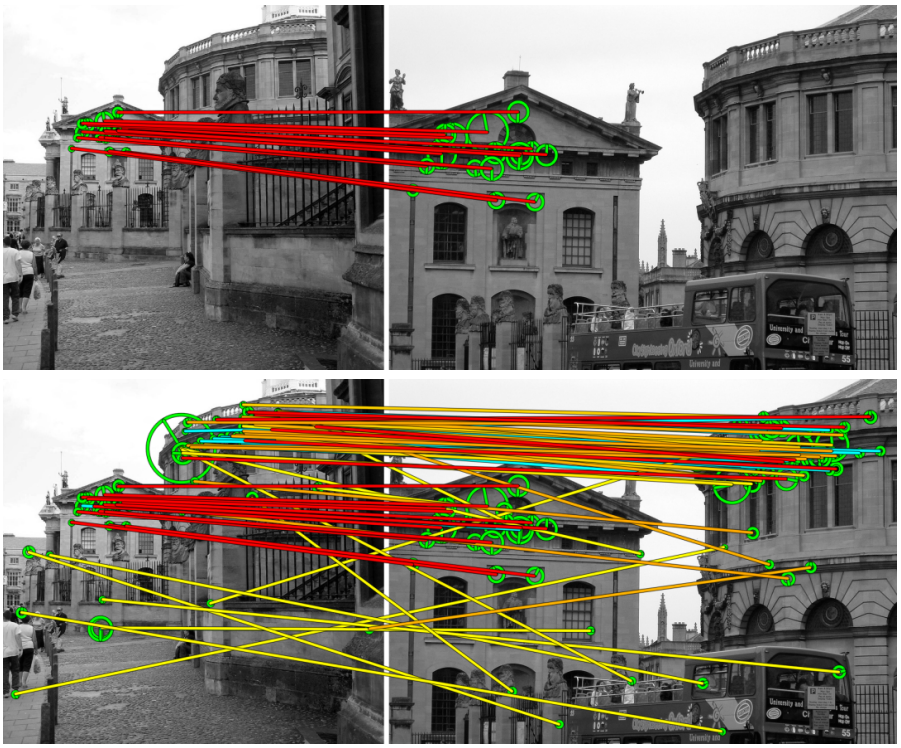


Figure 4.1: (Top) Inliers found by 4-*Degrees of Freedom* (DoF) FSM. (Bottom) HPM matching, with all tentative correspondences shown. The ones in cyan have been erased. The rest are colored according to strength, with red (yellow) being the strongest (weakest).

Given two images P, Q, an *assignment* or *correspondence* $c = (p, q)$ is a pair of features $p \in P, q \in Q$. The relative transformation from p to q is a *similarity transformation* given by

That is, a 4-DoF transformation consisting of translation, rotation and scale.

$$F(c) := F(q)F(p)^{-1} = \begin{bmatrix} M(c) & \mathbf{t}(c) \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{4.2}$$

where $M(c) = \sigma(c)R(c)$, $\mathbf{t}(c) = \mathbf{t}(q) - M(c)\mathbf{t}(p)$; and $\sigma(c) := \sigma(q)/\sigma(p)$, $R(c) := R(q)R(p)^{-1}$ are the relative scale and orientation respectively from p to q. This is a 4-*Degrees of Freedom* (DoF) transformation represented by *parameters*

$$f(c) := (x(c), y(c), \sigma(c), \theta(c)), \tag{4.3}$$

where $[x(c)\ y(c)]^\top := \mathbf{t}(c)$ and $\theta(c) := \theta(q) - \theta(p)$. Hence assignments are points in a d-dimensional *transformation space* $\mathcal{F}$ with $d = 4$.

An alternative is proximity of features in the descriptor space.

We are also given an set C of *tentative* correspondences. Two features correspond when assigned to the same visual word:

$$C := \{(p, q) \in P \times Q : u(p) = u(q)\}, \tag{4.4}$$

We use the IDF of visual words for w.

where $u(p)$ is the *visual word* of p. Each correspondence $c = (p, q) \in C$ is given a weight $w(c)$ measuring its relative importance.

Given a pair of assignments $c, c' \in C$, we assume an *affinity* measure $\alpha(c, c')$ as a non-increasing function of their distance in the transformation space. Two assignments are *conflicting* if they map two features of one image to the same feature of the other.

Our problem is then to identify a subset of non-conflicting assignments that maximizes the sum of affinity over all assignment pairs. This is a *binary quadratic programming* problem [325], but we only target a very fast, approximate solution. In fact, we want to group assignments according to affinity without ever enumerating pairs.

We are looking for one or more transformations that make parts of one image align to parts of the other.

4.3 HOUGH PYRAMID MATCHING

We assume that transformation parameters are mapped to $[0, 1]$, hence $\mathcal{F} := [0, 1]^d$. We construct a hierarchical partition

$$\mathcal{B} := \{B_0, \ldots, B_{L-1}\} \tag{4.5}$$

of $\mathcal{F}$ into L levels. Each $B_\ell \in \mathcal{B}$ is a partition of $\mathcal{F}$ into 2^{kd} *bins* (hypercubes), where $k := L - 1 - \ell$, obtained by partitioning each dimension into 2^k equal intervals of length 2^{-k}. B_0 is at the finest (bottom) level; B_{L-1} is at the coarsest (top), with a single bin. Given bin b, let

Each partition B_ℓ is a refinement of $B_{\ell+1}$.

We define a histogram pyramid *of correspondences into bins.*

$$h(b) := \{c \in C : f(c) \in b\} \tag{4.6}$$

be the set of correspondences with parameters falling into b.

MATCHING PROCESS We recursively split correspondences into bins in a *top-down* fashion, and then group them *bottom-up*. To impose a *one-to-one* mapping constraint, we detect conflicting correspondences at each level, choose one to keep and mark the remaining as *erased*. Let E_ℓ be the set of erased correspondences up to level ℓ. If $b \in B_\ell$ is a bin at level ℓ, we define its *group count* as

We write $[x]_+ := \max(x, 0)$ for the positive part *of $x \in \mathbb{R}$.*

$$g(b) := [|\hat{h}(b)| - 1]_+, \tag{4.7}$$

where $\hat{h}(b) := h(b) \setminus E_\ell$ are the correspondences we have kept in b.

Let $b_0 \subseteq \cdots \subseteq b_\ell$ be the sequence of bins containing a correspondence c up to level ℓ such that $b_k \in B_k$ for $k = 0, \ldots, \ell$. For each k, we approximate the affinity $\alpha(c, c')$ of c to any other correspondence $c' \in b_k$ by a fixed quantity, assumed a non-negative, non-increasing *level affinity* function of k,

λ controls the relative importance of successive levels, i.e. *how* relaxed *matching is.*

$$\alpha(k) := 2^{-\lambda k}. \tag{4.8}$$

Similar to [142], there are $g(b_k) - g(b_{k-1})$ new correspondences in a group with c at level k, giving rise to the *strength* of c up to level ℓ:

The strength of c depends on the size of the groups it participates in and the level at which they are formed.

$$s_\ell(c) := g(b_0) + \sum_{k=1}^{\ell} \alpha(k)\{g(b_k) - g(b_{k-1})\}. \tag{4.9}$$

Then, if $s(c) := s_{L-1}(c)$ is the *strength* of c, and excluding all erased assignments $E := E_{L-1}$, we define the *similarity score* of images P, Q

$$s(C) := \sum_{c \in C \setminus E} w(c)s(c). \tag{4.10}$$

The contributions of c, c' to $s(C)$ may only differ up to level $\ell - 1$.

If c, c' are two conflicting assignments and $b \in B_\ell$ is the first (finest) bin in the hierarchy with $c, c' \in b$, we impose the one-to-one mapping by keeping the strongest one up to level $\ell - 1$ according to (4.9).

The total operations per level are linear in $n := |C|$. Hence the time complexity of HPM is $O(nL)$.

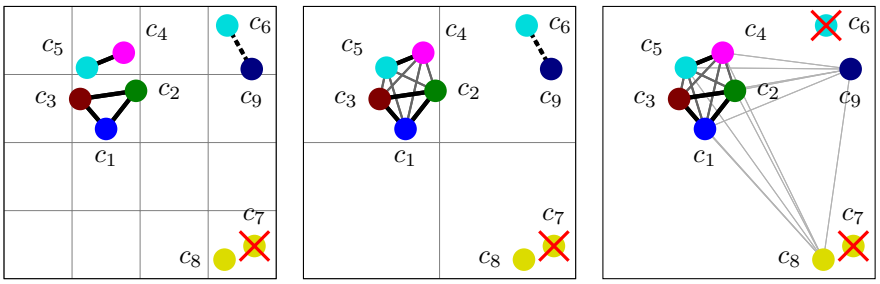


Figure 4.2: *Matching* of nine assignments on a 3-level pyramid in 2d space. Colors denote visual words, and edge strength denotes affinity. The dotted line between c_6, c_9 denotes a group that is formed at level 0 and then broken up at level 2, since c_6 is erased.

	p	q	strength
c_1			$(2 + \frac{1}{2}2 + \frac{1}{4}2)w(c_1)$
c_2			$(2 + \frac{1}{2}2 + \frac{1}{4}2)w(c_2)$
c_3			$(2 + \frac{1}{2}2 + \frac{1}{4}2)w(c_3)$
c_4			$(1 + \frac{1}{2}3 + \frac{1}{4}2)w(c_4)$
c_5			$(1 + \frac{1}{2}3 + \frac{1}{4}2)w(c_5)$
c_6			0
c_7			0
c_8			$\frac{1}{4}6w(c_8)$
c_9			$\frac{1}{4}6w(c_9)$

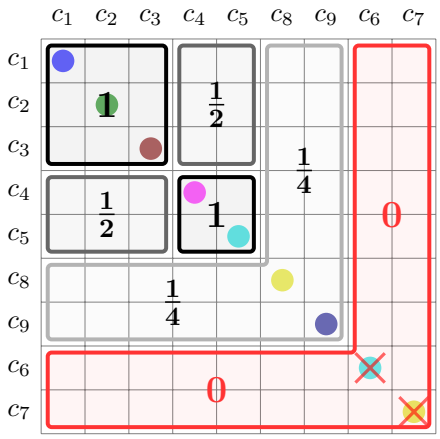


Figure 4.3: Referring to Figure 4.2: (Left) Assignment *labels, features and strength*. Vertices and edges denote features and assignments, respectively. Assignments c_5, c_6 and c_7, c_8 are conflicting, and c_6, c_7 are erased. Assignments $c_1, \ldots, c_5$ join groups at level 0; c_8, c_9 at level 2. (Right) *Affinity matrix* equivalent to strengths (4.9). Assignments have been arranged so that groups are contiguous. Groups formed at levels $0, 1, 2$ have affinity $1, \frac{1}{2}, \frac{1}{4}$ respectively.

EXAMPLE A toy 2d example is used for illustration. Figure 4.2 shows three groups of assignments at level 0: $\{c_1, c_2, c_3\}$, $\{c_4, c_5\}$ and $\{c_6, c_9\}$. The first two are joined at level 1. Assignments c_7, c_8 are conflicting, and c_7 is erased. Assignments c_5, c_6 are also conflicting, but are only compared at level 2; c_5 is stronger, being in a group of 5. Hence $\{c_6, c_9\}$ is broken up, c_6 is erased and c_8, c_9 join $c_1, \ldots, c_5$ in a group of 7 at level 2.

Figure 4.3 illustrates how the similarity score (4.10) is formed, with $\lambda = 1$. For instance, $c_1, \ldots, c_5$ contribute from all 3 levels, while c_8, c_9 only from level 2. These contributions are arranged in an $n \times n$ *affinity matrix* A. Summing affinities over a row of A and multiplying by the corresponding weight yields the assignment strength—the diagonal is excluded from the summation (4.7).

The upper triangular part of A corresponds to the edges of Figure 4.2, with edge strength being proportional to affinity.

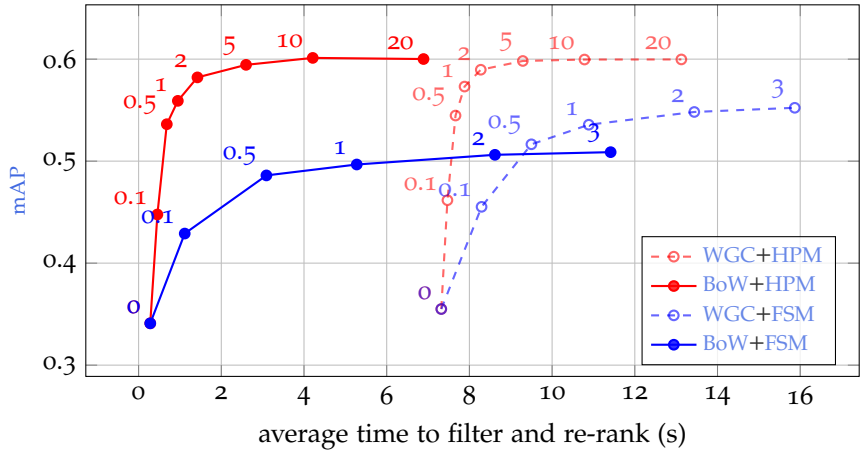


Figure 4.4: mAP *vs.* total (filtering + re-ranking) query time on *Barcelona* with 2M *World Cities* distractors for a varying number of re-ranked images (shown with labels near markers, in thousands).

4.4 EXPERIMENTS

SETUP We compare HPM against 4-DoF FSM [341] in pairwise matching and re-ranking in large scale search. In the latter case, we use two filtering models: BoW [413] and *Weak Geometric Consistency* (WGC) [203]. We use our *World Cities*[1] dataset, comprising a test set referred to as *Barcelona* and a 2M distractors set mostly depicting urban scenes exactly like the test set, but from different cities. We use SURF [31] features and descriptors.

RESULTS Here we only present mAP performance *vs.* query time for the re-ranking task, as shown in Figure 4.4 for varying number of re-ranked images. HPM re-ranks *ten times* more images in less time than FSM. With BoW, its mAP is 10% higher than FSM for the same re-ranking time, on average. At the price of 7 additional seconds for filtering, FSM

1 http://image.ntua.gr/iva/datasets/wc/

eventually benefits from WGC, while HPM is clearly unaffected. After 3.3 seconds, mAP performance of BoW+HPM reaches *saturation* after re-ranking 7k images, while WGC does not help.

4.5 DISCUSSION

HPM is a simple spatial matching algorithm that can be easily integrated in any image retrieval engine. It boosts performance by allowing flexible matching. It is arguably the first time geometry re-ranking reaches saturation in a practical query time. The practice so far has been to stop re-ranking such that queries do not take too long, without studying further improvement as in Figure 4.4.

After its introduction [433], we have extended HPM to use MA [342], applied to the query image only [203]. This extension further boosts performance [20]. In fact, HPM can work perfectly well even when descriptors are not quantized. Hence it can be applied to matching scenaria other than image retrieval.

Finding correspondences from descriptors is much more expensive than matching itself.

More can be found at our project page[2], including real retrieval examples along with comparisons for one query and the entire 2M *World Cities* dataset.

2 http://image.ntua.gr/iva/research/relaxed_spatial_matching/

5 BEYOND VOCABULARIES

We consider a family of metrics to compare images based on their local descriptors. It encompasses aggregated representations like VLAD [204] and selective match kernels like Hamming Embedding [202]. Making the bridge between these approaches leads us to introducing the Aggregated Selective Match Kernel *(ASMK) [436, 437], which takes the best of existing worlds. Finally, the representation underpinning this kernel is approximated, providing precise and scalable image search.*

5.1 INTRODUCTION

To partially recover from the quantization loss incurred by the use of a vocabulary, it is common to assign descriptors to multiple visual words. This is known as *Multiple Assignment* (MA) [342] and is preferably applied to the query image only [203] in an image retrieval context. Alternatively, one can learn a very *fine vocabulary* from data [298], where visual words have arbitrary shape in the descriptor space.

However, at the cost of some additional space, a more precise representation of the individual local descriptors works better than any vocabulary alone. One such solution is *Hamming Embedding* (HE) [203], where descriptors are represented by short binary codes apart from visual word. Such methods exhibit *selectivity*, *i.e.*, a correspondence contributes to the image-level similarity to a different extent, depending on an approximate distance in the descriptor space.

By contrast, *aggregation* or *pooling* operators, such as *Fisher vectors* [109, 339] or the *Vector of Locally Aggregated Descriptors* (VLAD) [204], depart from BoW by pooling vectors (*e.g.*, descriptors) rather than scalars (*e.g.*, frequencies). Compressing the resulting aggregated vectors [204, 338] yields a very compact global image representation, which however is inferior to local image representation.

Here we bridge the gap between *matching* approaches, such as HE, and *aggregated* representations, in particular VLAD. We introduce a class of *match kernels* that encompasses both. Selectivity is only exploited in matching approaches, which however do not use aggregation. We introduce a new representation that exploits the best of both worlds: the *Aggregated Selective Match Kernel* (ASMK) [436, 437].

Aggregation may still be beneficial in light of burstiness *[109, 148].*

5.2 MATCH KERNELS

We represent an image by a set $X := \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ of n d-dimensional local descriptors, quantized by a k-means quantizer $q : \mathbb{R}^d \to C$, where $C := \{\mathbf{c}_1, \dots, \mathbf{c}_k\} \subset \mathbb{R}^d$ is a codebook comprising k visual words. We denote by $X_{\mathbf{c}} := \{\mathbf{x} \in X : q(\mathbf{x}) = \mathbf{c}\}$ the subset of descrip-

tors in X that are assigned to visual word $\mathbf{c}$. Given two images X, Y, we consider a family of image-level similarity functions

$$K(X,Y) := \gamma(X)\,\gamma(Y) \sum_{\mathbf{c}\in C} w_{\mathbf{c}}\; \kappa\left(X_{\mathbf{c}}, Y_{\mathbf{c}}\right), \tag{5.1}$$

For instance, IDF weights.

where scalar $w_{\mathbf{c}}$ is a weight of visual word $\mathbf{c}$, κ is a *match kernel* used as visual word-level similarity function and γ is a normalization function such that the self-similarity of an image is $K(X,X) = 1$. This family encompasses several well known methods.

BAG-OF-WORDS [83, 413] represents each local descriptor $\mathbf{x}$ solely by its visual word. As observed in [45, 203], bag-of-words with cosine similarity can be expressed in terms of (5.1), by defining

$$\kappa_{\mathrm{BoW}}(X_{\mathbf{c}}, Y_{\mathbf{c}}) := |X_{\mathbf{c}}| \times |Y_{\mathbf{c}}| = \sum_{\mathbf{x}\in X_{\mathbf{c}}} \sum_{\mathbf{y}\in Y_{\mathbf{c}}} 1. \tag{5.2}$$

HAMMING EMBEDDING (HE) [202, 203] extends BoW by representing each local descriptor $\mathbf{x}$ with both its quantized value $q(\mathbf{x})$ and a binary code $b_{\mathbf{x}} \in \{-1,+1\}^B$ of B bits. It matches all pairs of descriptors assigned to the same visual word by

$$\kappa_{\mathrm{HE}}\left(X_{\mathbf{c}}, Y_{\mathbf{c}}\right) := \sum_{\mathbf{x}\in X_{\mathbf{c}}} \sum_{\mathbf{y}\in Y_{\mathbf{c}}} w\left(d_H(b_{\mathbf{x}}, b_{\mathbf{y}})\right), \tag{5.3}$$

w can be for instance a step [202] or Gaussian function [148].

where w is a non-increasing similarity function and d_H is the Hamming distance, which can be expressed as $d_H(\mathbf{a},\mathbf{b}) = \frac{B}{2}(1 - \hat{\mathbf{a}}^\top \hat{\mathbf{b}})$. Here $\hat{\mathbf{a}}$ denotes the ℓ_2-normalized counterpart of vector $\mathbf{a}$.

VLAD [204] aggregates the descriptors per visual word into a dk-vector representation $V(X) \propto [v(X_{\mathbf{c}_1}), \ldots, v(X_{\mathbf{c}_k})]$, where

$$v(X_{\mathbf{c}}) := \sum_{\mathbf{x}\in X_{\mathbf{c}}} r(\mathbf{x}) = \sum_{\mathbf{x}\in X_{\mathbf{c}}} \mathbf{x} - q(\mathbf{x}), \tag{5.4}$$

The power-law normalization of Fisher vectors [109] cannot be expanded as (5.5).

and $r(\mathbf{x})$ is the *residual vector* of $\mathbf{x}$. Then, VLAD with cosine similarity can again be expressed in terms of (5.1) by

$$\kappa_{\mathrm{VLAD}}\left(X_{\mathbf{c}}, Y_{\mathbf{c}}\right) := v(X_{\mathbf{c}})^\top v(Y_{\mathbf{c}}) = \sum_{\mathbf{x}\in X_{\mathbf{c}}} \sum_{\mathbf{y}\in Y_{\mathbf{c}}} r(\mathbf{x})^\top r(\mathbf{y}). \tag{5.5}$$

5.3 TOWARDS A COMMON MODEL

The match kernels discussed above can be classified into two kinds.

NON-AGGREGATED KERNELS individually match all the elements occurring in the same vocabulary cell. They are written as

$$\kappa_N(X_{\mathbf{c}}, Y_{\mathbf{c}}) := \sum_{\mathbf{x}\in X_{\mathbf{c}}} \sum_{\mathbf{y}\in Y_{\mathbf{c}}} \sigma\left(\phi(\mathbf{x})^\top \phi(\mathbf{y})\right). \tag{5.6}$$

encompassing all variants discussed so far. Here ϕ is a vector representation function, possibly nonlinear or including normalization, and $\sigma : \mathbb{R} \to \mathbb{R}$ is a scalar *selectivity* function.

AGGREGATED KERNELS, in contrast, are written as

$$\kappa_A(X_\mathbf{c}, Y_\mathbf{c}) := \sigma\left\{\psi\left(\sum_{\mathbf{x}\in X_\mathbf{c}} \phi(\mathbf{x})\right)^\top \psi\left(\sum_{\mathbf{y}\in Y_\mathbf{c}} \phi(\mathbf{y})\right)\right\} \tag{5.7}$$

$$= \sigma\left(\Phi(X_\mathbf{c})^\top \Phi(Y_\mathbf{c})\right), \tag{5.8}$$

where ψ is another vector representation function, again possibly nonlinear or including normalization. $\Phi(X_\mathbf{c})$ is the aggregated vector representation of a set $X_\mathbf{c}$ of descriptors in a cell. In contrast to (5.6), selectivity σ is applied after aggregation.

Aggregation, i.e., computing $\Phi(X_\mathbf{c})$, is an offline operation.

MODEL	$\kappa(X_\mathbf{c}, Y_\mathbf{c})$	$\phi(\mathbf{x})$	$\sigma(u)$	$\psi(\mathbf{z})$	$\Phi(X_\mathbf{c})$
BoW (5.2)	κ_N or κ_A	$\mathbf{1}$	u	$\mathbf{z}$	$\lvert X_\mathbf{c}\rvert$
HE (5.3)	κ_N	$\hat{b}_\mathbf{x}$	$w\left(\frac{B}{2}(1-u)\right)$	–	–
VLAD (5.5)	κ_N or κ_A	$r(\mathbf{x})$	u	$\mathbf{z}$	$V(X_\mathbf{c})$
ASMK (5.11)	κ_A	$r(\mathbf{x})$	$\sigma_\alpha(u)$	$\hat{\mathbf{z}}$	$\hat{V}(X_\mathbf{c})$

Table 5.1: Existing and new solution ASMK for the match kernel κ, classified as aggregated κ_A (5.7), non-aggregated κ_N (5.6), or both. $\phi(\mathbf{x})$: Scalar or vector representation of descriptor $\mathbf{x}$. σ: Scalar selectivity. $\psi(\mathbf{z})$: Vector representation of aggregated descriptor $\mathbf{z}$. $\Phi(X_\mathbf{c})$: Equivalent representation of descriptor set $X_\mathbf{c}$ per cell.

A COMMON MODEL Table 5.1 summarizes the BoW, HE and VLAD kernels and expresses them in a common model. It also identifies in each case options for functions ϕ, σ, ψ and Φ. BoW and VLAD both fit into (5.6) and (5.7), with σ simply being identity. This is not the case for HE matching, which has a nonlinear σ, hence only fits into (5.6). This analysis suggests other potential strategies.

5.4 AGGREGATED SELECTIVE MATCH KERNEL

Our ASMK is motivated observing that VLAD employs a linear combination in (5.5) of the contributions of individual descriptor pairs $(\mathbf{x}, \mathbf{y})$ to κ, while HE applies a nonlinear weighting function σ to the similarity $\phi(\mathbf{x})^\top \phi(\mathbf{y})$ in (5.3), but involves no aggregation.

SELECTIVITY Without loss of generality, we consider a thresholded polynomial selectivity function $\sigma_\alpha : \mathbb{R} \to \mathbb{R}^+$ of the form

$$\sigma_\alpha(u) := \begin{cases} \operatorname{sgn}(u)|u|^\alpha & \text{if } u > \tau \\ 0 & \text{otherwise,} \end{cases} \tag{5.9}$$

and typically set $\alpha = 3$, while $\tau \geq 0$. Figure 5.1 shows the effect of this function σ_α when matching features between two images, for different values of the exponent α. A larger α increases the selectivity and

σ_α plays the same role as w in (5.3), applied to similarities instead of distances.

Figure 5.1: *Matching features* with descriptors assigned to the same visual word and similarity above threshold $\tau = 0.25$. (Top) $\alpha = 1$; (bottom) $\alpha = 3$. Color denotes descriptor similarity $\sigma_\alpha(\hat{r}(\mathbf{x})^\top \hat{r}(\mathbf{y}))$, with yellow (red) being low (high).

drastically down-weights false correspondences. This advantageously replaces hard thresholding as initially proposed in HE [202].

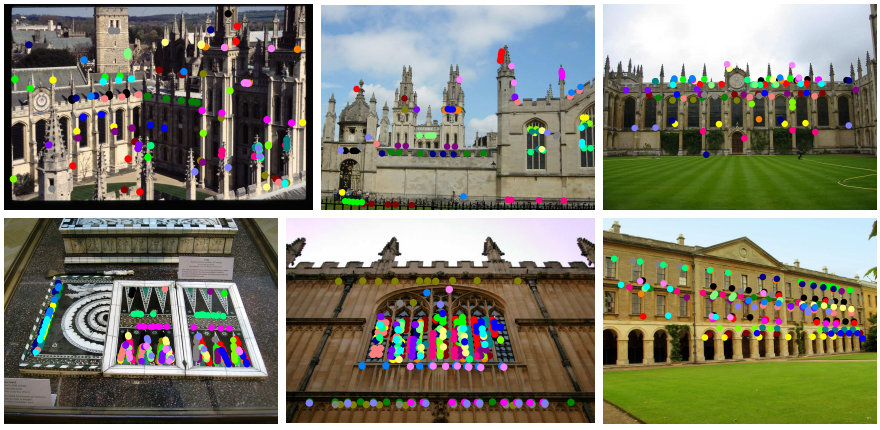


Figure 5.2: Examples of features *mapped to the same visual word*. Top 25 visual words are drawn, each in different color, according the number of features mapped to them.

Such bursty *features are common in repeated structures,* e.g. *in urban images, and textured regions.*

AGGREGATION We propose to apply a selectivity function *after* aggregating descriptors per cell, producing a more compact representation. Figure 5.2 illustrates several examples of features that are aggregated. Their matches usually dominate the image level similarity. To mitigate this effect, it is common to *down-weight* [148] or *normalize* [9]

the contribution per visual word. This resembles *binary* BoW [413] or *max pooling* [48], which account at most one vote per visual word. We follow the same approach but for large vocabularies.

OUR ASMK We aggregate the residual vectors as in VLAD (5.4) and then ℓ_2-normalize the sum:

The database vectors $\hat{v}(X_\mathbf{c})$ are computed offline.

$$\Phi_{\text{ASMK}}(X_\mathbf{c}) := \hat{v}(X_\mathbf{c}) = v(X_\mathbf{c})/\|v(X_\mathbf{c})\|. \tag{5.10}$$

ASMK is an aggregated kernel (5.8) with selectivity function σ_α:

$$\kappa_{\text{ASMK}}(X_\mathbf{c}, Y_\mathbf{c}) := \sigma_\alpha\left(\hat{v}(X_\mathbf{c})^\top \hat{v}(Y_\mathbf{c})\right). \tag{5.11}$$

ASMK is also summarized in Table 5.1. It handles burstiness by keeping only one representative of all bursty descriptors per cell, which is represented by the normalized mean residual.

BINARIZATION For the sake of compactness, HE represents the residual $r(\mathbf{x}) := \mathbf{x} - q(\mathbf{x})$ by a binary vector $b_\mathbf{x}$. We develop ASMK using full d-dimensional descriptors to investigate the upper bound in performance; in practice, we use an approximated version ASMK*, binarizing $v(X_\mathbf{c})$ (5.10) before applying the selectivity function.

DATASET	MA	OXF5K	OXF105K	PARIS6K	HOLIDAYS
HE [203]		51.7	–	–	74.5
HE [203]	✓	56.1	–	–	77.5
HE-burstiness [198]		64.5	–	–	78.0
HE-burstiness [198]	✓	67.4	–	–	79.6
Fine vocabulary [298]	✓	74.2	67.4	74.9	74.9
ASMK*		76.4	69.2	74.4	80.0
ASMK*	✓	80.4	**75.0**	77.0	81.0
ASMK		78.1	–	76.0	81.2
ASMK	✓	**81.7**	–	**78.2**	**82.2**

Table 5.2: mAP comparison of ASMK variants against state-of-the-art with $\alpha = 3$, $\tau = 0$ and $k = 65$k visual words.

5.5 EXPERIMENTS

SETUP We evaluate ASMK against *fine vocabularies* [298] and descriptor-based HE variants [198, 203] on Oxford Buildings [341], Paris [342] and Holidays [203] datasets. We use a modified *Hessian-Affine* feature detector [334] and SIFT descriptors with component-wise *square root* [8, 197]. We use flat k-means to learn visual vocabularies of 65k visual words on an independent dataset. We combine with MA using 5 nearest visual words on query side only [203].

We do not use any post-processing like spatial verification or query expansion.

RESULTS As shown in Table 5.2, ASMK outperforms the binarized ASMK⋆, which outperforms all other methods by a large margin. ASMK relies on full descriptors and does not scale to Oxford105k; only ASMK⋆ does. ASMK⋆ uses less memory and is faster than HE, having less features after aggregation.

5.6 DISCUSSION

By building a common model for match kernels like BoW, HE and VLAD, it becomes evident that different kernels have different properties. ASMK combines the selectivity of HE with the aggregation per visual word of VLAD, successfully combating burstiness.

Aggregation across images [437].

After its introduction [436], we have extended ASMK to operate *across images* [437], identifying local features in different images corresponding to the same physical structure. We do this by pairwise matching descriptors of all images in a dataset per visual word, connecting nearby pairs and forming the connected components of the resulting graph. We aggregate their representation into a single vector per component. We thus further compress the indexing structure and implicitly perform feature augmentation [446].

Early burst detection [400].

Another extension is *early burst detection* [400], where bursts are explicitly detected *before* quantizing to visual words. The simplest and most effective way to do so is to compute all pairwise feature similarities (according to scale, orientation and descriptor), connect features into components and aggregate descriptors per component. This applies to VLAD, ASMK and any aggregated kernel. Performance is on par with the state of the art but with significantly reduced complexity, thanks to the fewer descriptors kept. An interesting finding is the benefit of *asymmetric* aggregation, *i.e.*, aggregating the database descriptors and not those of the query.

More can be found at our project home page[1], including software.

1 http://image.ntua.gr/iva/research/asmk/

6 NEAREST NEIGHBOR SEARCH

We present a simple vector quantizer that combines low distortion with fast search and apply it to Approximate Nearest Neighbor *(ANN) search in high dimensional spaces [213]. Leveraging the very same data structure that is used to provide non-exhaustive search,* i.e.*, inverted lists or a multi-index, the idea is to* locally *optimize over rotation and space decomposition an individual product quantizer per cell and use it to encode residuals.*

6.1 INTRODUCTION

By experimenting with different vocabulary options [298, 342] and representations alternative to BoW [109, 203, 204, 436], it becomes evident that image retrieval boils down to high-quality, *compact* encoding of visual descriptors that allows for *fast* search. This is ANN search in high-dimensional spaces, a recurring problem in computer vision.

This is in contrast to search methods that maintain all data uncompressed [309].

Product Quantization (PQ) [200] is a compact encoding method that can be used for exhaustive or non-exhaustive search through *inverted indexing* or *multi-indexing* [22]. Better fitting to the underlying distribution is critical in search performance, as in *Optimized Product Quantization* (OPQ) [122]. The principle is that *all bits allocated to data points should be used sparingly*. Such methods should be ultimately seen as (lossy) *data compression* targeting minimal distortion.

The same is true for most hashing *methods [162].*

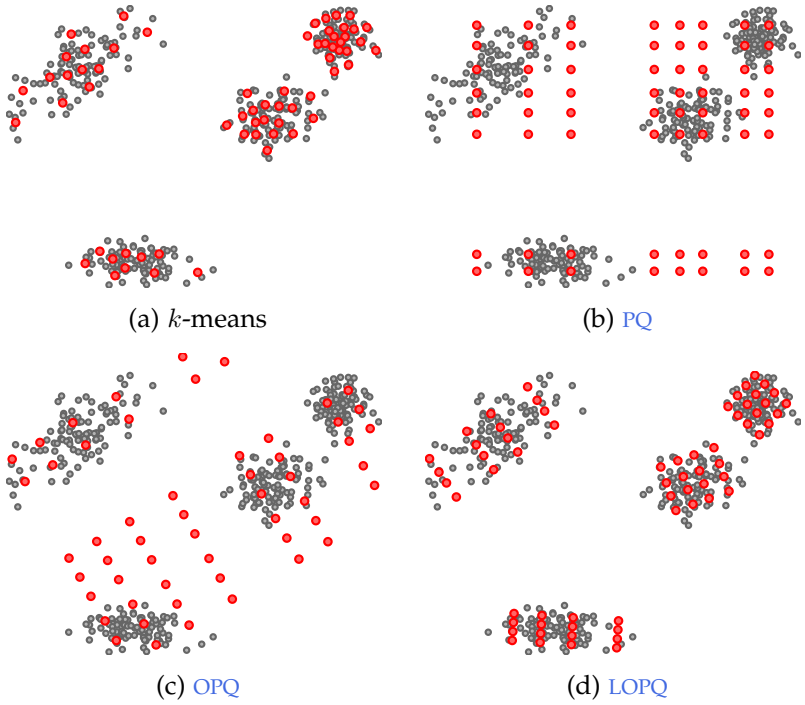
As such, k-means, depicted in Figure 6.1(a), allows $\log_2 k$ bits to represent a data point in $\mathbb{R}^d$ by specifying k centroids. But naïve search is $O(dk)$ and low distortion means very large k. By constraining centroids on an axis-aligned, m-dimensional grid, PQ achieves k^m centroids keeping search at $O(dk)$; but as in Figure 6.1(b), many centroids may remain without data support. OPQ allows arbitrary rotation and re-ordering of dimensions to better align to data and balance variance across subspaces. But as illustrated in Figure 6.1(c), a multimodal distribution may not benefit from such alignment.

Our solution is *Locally Optimized Product Quantization* (LOPQ) [213]. Following [200], a coarse quantizer is used to index data, and residuals between data points and centroids are PQ-encoded. But within-cell distributions are largely unimodal; hence, as in Figure 6.1(d), we locally optimize an individual product quantizer per cell. Under no assumptions on the distribution, all centroids are supported by data, resulting a lower distortion.

We also combine with multi-index [22], which is essential for large scale datasets.

6.2 BACKGROUND

VECTOR QUANTIZATION A *quantizer* is a function q that maps a d-dimensional vector $\mathbf{x} \in \mathbb{R}^d$ to vector $q(\mathbf{x}) \in C$, where $C \subset \mathbb{R}^d$ is a



(a) k-means (b) PQ (c) OPQ (d) LOPQ

Figure 6.1: Four quantizers of 64 centroids (▫) each, trained on a random set of 2D points (▫), following a mixture distribution. (c) and (d) also reorder dimensions, which is not shown in 2D.

codebook with $k := |C|$. Each vector $\mathbf{c} \in C$ is called a *centroid*. Given a set $X := \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of n data points in $\mathbb{R}^d$, q induces *distortion*

$$E := \sum_{\mathbf{x} \in X} \|\mathbf{x} - q(\mathbf{x})\|^2 . \tag{6.1}$$

Hence, an optimal quantizer should minimize distortion E as a function of codebook C alone.

According to *Lloyd's first condition* [143], regardless C, a quantizer that minimizes distortion should map $\mathbf{x} \in \mathbb{R}^d$ to its nearest centroid:

$$q(\mathbf{x}) := \arg\min_{\mathbf{c} \in C} \|\mathbf{x} - \mathbf{c}\| . \tag{6.2}$$

PRODUCT QUANTIZATION Assuming that dimension d is a multiple of m, write any vector $\mathbf{x} \in \mathbb{R}^d$ as a concatenation $(\mathbf{x}^1, \ldots, \mathbf{x}^m)$ of m sub-vectors, each of dimension d/m. If $C^1, \ldots, C^m$ are m sub-codebooks in subspace $\mathbb{R}^{d/m}$, each of k sub-centroids, a *product quantizer* [200] constrains C to the Cartesian product

$$C = C^1 \times \cdots \times C^m, \tag{6.3}$$

We write $[n] = \{1, \ldots, n\}$ for $n \in \mathbb{N}$.

i.e., a codebook of k^m centroids $\mathbf{c} = (\mathbf{c}^1, \ldots, \mathbf{c}^m)$ with each sub-centroid $\mathbf{c}^j \in C^j$ for $j \in [m]$. An optimal product quantizer q should minimize E (6.1) as a function of C, subject to C being of the form (6.3) [122]. In this case, for each $\mathbf{x} \in \mathbb{R}^d$, the nearest centroid in C is

$$q(\mathbf{x}) = (q^1(\mathbf{x}^1), \ldots, q^m(\mathbf{x}^m)), \tag{6.4}$$

where $q^j(\mathbf{x}^j)$ is the nearest sub-centroid of sub-vector $\mathbf{x}^j$ in C^j, for $j \in [m]$ [122]. Hence an optimal product quantizer q in d dimensions

incurs m subproblems of m optimal sub-quantizers $q^j, j \in [m]$, each in d/m dimensions. We write $q = (q^1, \ldots, q^m)$ in this case.

OPTIMIZED PRODUCT QUANTIZATION [122, 320] refers to optimizing the subspace decomposition too. Constraint (6.3) is relaxed to

$$C = \{R\hat{\mathbf{c}} : \hat{\mathbf{c}} \in C^1 \times \cdots \times C^m, R^\top R = I\}, \tag{6.5}$$

where orthogonal $d \times d$ matrix R allows for rotation and permutation of dimensions. Optimization with respect to $C^1, \ldots, C^m$ and R is either joint as in *Cartesian k-Means* (CKM) [320] and in the *non-parametric* OPQ_{NP} of [122], or decoupled, as in the *parametric* OPQ_{P} of [122].

EXHAUSTIVE SEARCH Given a product quantizer $q = (q^1, \ldots, q^m)$, let each encoded data point $q(\mathbf{x})$ be represented by tuple $(i^1, \ldots, i^m)$ of m sub-centroid indices (6.4), each in index set $[k]$. Given a *query* vector $\mathbf{y}$, the (squared) Euclidean distance to $\mathbf{x} \in X$ may be approximated by the *Asymmetric Distance Computation* (ADC) [200]

This PQ-encoding requires $m \log_2 k$ bits per point.

$$\delta_q(\mathbf{y}, \mathbf{x}) := \|\mathbf{y} - q(\mathbf{x})\|^2 = \sum_{j=1}^{m} \left\|\mathbf{y}^j - q^j(\mathbf{x}^j)\right\|^2, \tag{6.6}$$

where $q^j(\mathbf{x}^j) \in C^j := \{\mathbf{c}_1^j, \ldots, \mathbf{c}_k^j\}$ for $j \in [m]$. Distances $\|\mathbf{y}^j - \mathbf{c}_i^j\|^2$ are precomputed for $i \in [k]$ and $j \in [m]$, so (6.6) amounts to only $O(m)$ lookup and add operations.

INDEXING The *residual* vector of $\mathbf{x} \in \mathbb{R}^d$ quantized by q is

$$r_q(\mathbf{x}) := \mathbf{x} - q(\mathbf{x}). \tag{6.7}$$

Non-exhaustive search involves a *coarse quantizer* Q of K centroids, or *cells*. Each point $\mathbf{x} \in X$ is quantized to $Q(\mathbf{x})$, and its residual vector $r_Q(\mathbf{x})$ is quantized by a product quantizer q. A query point $\mathbf{y}$ is quantized to its w nearest cells, and approximate distances to residuals are then computed (6.6) only within those cells. This is referred to as *Inverted File - ADC* (IVFADC) [200].

An inverted list of points and PQ-encoded residuals are maintained per cell.

MULTI-INDEXING applies the idea of PQ to the coarse quantizer used for indexing. A second-order *inverted multi-index* [22] comprises two *subspace quantizers* over $\mathbb{R}^{d/2}$, each of K sub-centroids. A *cell* is now a pair of sub-centroids. There are K^2 cells on a 2-dimensional *grid*, inducing a fine partition of $\mathbb{R}^d$. Quantization and distance computation is done independently per subspace. At query time, cells are traversed in increasing order of distance to the query by the *multi-sequence* algorithm.

A solution involving PQ-encoding of residuals is called *Multi-Index ADC* (Multi-D-ADC) [22]. A solution using OPQ_{NP} to globally optimize both the residuals and the data prior to multi-index construction, is referred to as *Optimized Multi-Index ADC* (OMulti-D-OADC) [123].

6.3 LOCALLY OPTIMIZED PRODUCT QUANTIZATION

We develop our solution for a single coarse quantizer first. We then outline our solution for a multi-index.

SINGLE INDEX Given X, we optimize a coarse quantizer Q, with codebook $B := \{\mathbf{b}_1, \dots, \mathbf{b}_K\}$ of K cells. For $i \in [K]$, we collect the residuals of points quantized to cell $\mathbf{b}_i$

LOPQ incurs $O(K(d^2 + dk))$ space overhead and $O(wd^2)$ query time overhead comparing to IVFADC [200].

$$Z_i := \{\mathbf{x} - \mathbf{b}_i : \mathbf{x} \in X,\ Q(\mathbf{x}) = \mathbf{b}_i\}. \tag{6.8}$$

For each cell $i \in [K]$, we *locally optimize* PQ encoding of residuals in Z_i, yielding an orthogonal matrix R_i and a product quantizer q_i. Residuals are then locally rotated and PQ-encoded as $q_i(R_i^\top \mathbf{z})$ for $\mathbf{z} \in Z_i$. At query time, the query point $\mathbf{y}$ is soft-assigned to its w nearest cells in B. For each such cell $\mathbf{b}_i$, asymmetric distances $\delta_{q_i}(R_i^\top r_Q(\mathbf{y}), R_i^\top \mathbf{z})$ of the rotated query residual $R_i^\top r_Q(\mathbf{y}) = R_i^\top(\mathbf{y} - \mathbf{b}_i)$ to rotated residuals $R_i^\top \mathbf{z}$ for $\mathbf{z} \in Z_i$ are then computed according to (6.6), using the underlying local product quantizer q_i.

The computation is exhaustive within Z_i, but is performed in the compressed domain.

LOCAL OPTIMIZATION Let $Z \in \{Z_1, \dots, Z_K\}$ be the set of residuals of data points quantized to some cell in B. Contrary to [200], we PQ-encode these residuals by locally optimizing both space decomposition and sub-quantizers per cell. Given m and k, this problem is expressed as minimizing distortion as a function of orthogonal matrix $R \in \mathbb{R}^{d \times d}$ and sub-codebooks $C^1, \dots, C^m \subset \mathbb{R}^{d/m}$ per cell,

$$\begin{aligned} \text{minimize} \quad & \sum_{\mathbf{z} \in Z} \min_{\hat{\mathbf{c}} \in \hat{C}} \|\mathbf{z} - R\hat{\mathbf{c}}\|^2 \\ \text{subject to} \quad & \hat{C} = C^1 \times \cdots \times C^m \\ & R^\top R = I, \end{aligned} \tag{6.9}$$

Given solution $R, C^1, \dots, C^m$, codebook C is given by (6.5).

where $|C^j| = k$ for $j \in [m]$. Sub-codebook C^j determines a sub-quantizer q^j by

$$q^j(\mathbf{x}) = \arg\min_{\hat{\mathbf{c}}^j \in C^j} \|\mathbf{x} - \hat{\mathbf{c}}^j\| \tag{6.10}$$

for $j \in [m]$, $\mathbf{x} \in \mathbb{R}^{d/m}$, as in (6.2); collectively, sub-quantizers determine a product quantizer $q = (q^1, \dots, q^m)$ by (6.4). Local optimization can then be seen as a mapping $Z \mapsto (R, q)$. We follow the parametric solution of [122, 320] that we briefly describe here.

PARAMETRIC SOLUTION OPQ_P [122] assumes a zero-mean normal distribution $\mathcal{N}(\mathbf{0}, \Sigma)$ of residual data Z and minimizes the theoretical lower distortion bound as a function of R alone [122]. That is, R is optimized independently prior to codebook optimization, which follows by independent k-means per subspace, exactly as in PQ. Given the covariance matrix Σ, empirically measured on Z, the solution for R is found in closed form, in two steps.

First, rotating data by $\hat{\mathbf{z}} \leftarrow R^\top \mathbf{z}$ should yield a block-diagonal covariance matrix $\hat{\Sigma}$, with the j-th diagonal block being sub-matrix $\hat{\Sigma}_{jj}$ of j-th subspace, for $j \in [m]$. That is, subspace distributions should be pairwise *independent*. This is accomplished by diagonalizing Σ as $U\Lambda U^\top$. Second, determinants $|\hat{\Sigma}_{jj}|$ should be equal for $j \in [m]$, *i.e.*, variance should be *balanced* across subspaces. This is done by *eigenvalue allocation* [122]: Eigenvalues in Λ are traversed in descending order and greedily allocated to the subspace of minimal variance. This yields a permutation π of the set $[d]$ of dimensions. Finally, the solution is $R := UP_\pi^\top$, where P_π is the permutation matrix of π.

The solution for R represents a re-ordering of the eigenvectors of Σ.

MULTI-INDEX In the case of a second-order multi-index, the space overhead is prohibitive to locally optimize per cell. Hence, we separately optimize a rotation and a set of sub-quantizers per cell of the two subspace quantizers and PQ-encode two sub-residuals per data point. At query time, the query needs to be rotated independently for each such cell. Rotations are *lazy-evaluated i.e.* computed on demand by the multi-sequence algorithm and stored for re-use. We call this solution *Multi-Index LOPQ* (Multi-LOPQ).

Multi-LOPQ incurs $O(K(d^2 + dk))$ space overhead comparing to Multi-D-ADC [22]. The query time overhead is $O(Kd^2)$ in the worst case, but much lower in practice.

This solution is more constrained than in the case of a single index: each rotation matrix is constrained to be block-diagonal, keeping rotations within-subspace. By contrast, the rotation matrix in [123] is unconstrained, but it is fixed for all cells.

t	METHOD	$r = 1$	10	100
10k	Multi-D-ADC [22]	0.304	0.665	0.740
	OMulti-D-OADC [123]	0.345	0.725	**0.794**
	Multi-LOPQ	**0.430**	**0.761**	0.782
100k	Multi-D-ADC [22]	0.334	0.793	0.959
	OMulti-D-OADC [123]	0.373	0.841	**0.973**
	Multi-LOPQ	**0.476**	**0.919**	**0.973**

Table 6.1: Recall@r for $r \in \{1, 10, 100\}$ on SIFT1B with 128-bit codes, $k = 256$, *i.e.* 8 bits per sub-quantizer, $m = 16$ subspaces, and subspace quantizers with $K = 2^{14}$; t is the target number of points fetched by multi-sequence.

6.4 EXPERIMENTS

SETUP Here we only include large-scale results of non-exhaustive search with a multi-index. We use the SIFT1B [206][1] dataset, containing 1 billion SIFT vectors and 10K queries. We evaluate Multi-LOPQ against Multi-D-ADC [22] and OMulti-D-OADC [123]. All methods PQ-encode the residuals of the subspace quantizers with 128-bit codes.

1 http://corpus-texmex.irisa.fr/

As in related work [22, 122, 200, 206, 320, 321], we measure search performance via *recall@r*, *i.e.* the proportion of queries having their nearest neighbor ranked in the first r positions. Recall@1 is the most important, and is equivalent to the *precision* of [309].

Alternatively, recall@r is the fraction of queries for which the nearest neighbor would be correctly found if we verified the r top-ranking vectors using exact distances.

RESULTS As shown in Table 6.1, the optimized OMulti-D-OADC [123] outperforms Multi-D-ADC [22]. However, the performance of Multi-LOPQ is unprecedented, enjoying nearly 10% gain over OMulti-D-OADC on the most important measure of precision (recall@1).

Both space and time overhead is constant in data size n.

OVERHEAD With $K = 2^{14}$, the *space overhead* of Multi-LOPQ on top of Multi-D-ADC is 500MB for rotation matrices and 2GB for sub-quantizer centroids, compared to 21GB that is the total SIFT1B index space with 128-bit codes. The *query time overhead* is the time needed to rotate the query for each cell. On average, this is 0.776 and 4.04ms respectively for $t = 10$k and 100k, compared to 7 and 49ms respectively for a Multi-D-ADC query.

6.5 DISCUSSION

Beneath LOPQ lies the very simple idea that no single centroid should be wasted by not representing actual data. Rather, each should contribute to lowering distortion. Hence, to take advantage of PQ, one should attempt to use and optimize product quantizers over parts of the data only.

LOPQ resembles a two-stage fitting of a *mixture distribution*: component means followed by conditional densities via PQ. Joint optimization of coarse and local quantizers might bring further improvement, but its training cost would be prohibitive.

More can be found at our project home page[2], including software.

2 http://image.ntua.gr/iva/research/lopq/

7 EXPLORING PHOTO COLLECTIONS

We introduce an image clustering scheme that compresses a large corpus of images by grouping visually consistent ones, while providing a guaranteed distortion bound. This allows representing thousands of images depicting a landmark, while still being able to retrieve isolated non-landmark images. Starting from a geo-tagged dataset, we group images geographically and then visually. We align all views to a reference image and construct a 2d scene map *[19]. Indexing and retrieval then operates directly on scene maps. We apply to location and landmark recognition and we demonstrate several integrated methods through our online application, VIRaL[1] [215].*

7.1 INTRODUCTION

Billions of images are available online along with metadata such as location, time and tags. Applications are emerging, for instance *location estimation* [160], *virtual tourism* [414], and *landmark recognition* [514]. Here we are interested in *location recognition* given a single image, be it landmark or not. Unfortunately, current solutions either do not scale well, or focus on points of interest like landmarks.

[160] only estimates a geolocation probability map, while [514] only works on landmarks.

Our work lies between generic *image retrieval* and *clustering*. While large image clusters of popular places help in terms of efficiency, a *distortion bound* can guarantee that isolated images are still found as in a generic retrieval engine. For instance, when clustering geo-tagged images by location [82], two images taken 2km apart are unlikely to depict the same building. Likewise, in spatial matching [341], 20 images each having 15 inliers with a reference image, may all depict similar views of a single scene.

We use *Kernel Vector Quantization* (KVQ) [432], along with an appropriate metric to group images by location and then by visual similarity. Contrary to other solutions, this guarantees that no image in a cluster is too "far away" (depending on the metric) from the rest. Given a visual cluster, we align all images to a reference image and construct a 2d *scene map* by grouping local features, giving rise to another application of KVQ. Finally, we extend the entire search pipeline operate on scene maps rather than images. This not only provides memory savings, but increases recall too.

To speed up mining, we apply visual clustering only within each geo-cluster and we use sub-linear indexing for pairwise matching.

7.2 BACKGROUND

It is common to perform *geo-clustering* by location (latitude, longitude) followed by *visual clustering*. The objective is to identify photos depicting *views* of the same scene. For instance, [82, 258] perform geo-

Views of the same scene are not expected in photos taken too far apart, so geo-clustering helps accelerate visual clustering.

1 http://viral.image.ntua.gr/

clustering alone by *mean-shift* [69], while visual clustering follows using *e.g.* *k-means* [220] and *agglomerative clustering* [120, 349, 514]. The drawback of k-means and agglomerative clustering is that there is no control over the maximal intra-cluster distance, while mean-shift [82, 258] requires *seeding* and *fixed tiles* [120, 349] do not adjust to data. We use KVQ [432], which guarantees an upper bound on distortion and adjusts the number of clusters accordingly.

KERNEL VECTOR QUANTIZATION Let $(\mathcal{X}, d)$ be a metric space. Given a set $X := \{x_1, \ldots, x_n\} \subseteq \mathcal{X}$, we are looking for a small subset $Q(X) \subseteq X$ such that no point in X is too far away from some point in Q. Define *kernel function* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ by

$B_r(x)$ is the open ball of radius r centered at x; $\mathbb{1}_A : \mathcal{X} \to \{0, 1\}$ is the indicator function of $A \subseteq \mathcal{X}$.

$$k(x, y) := \mathbb{1}_{B_r(x)}(y), \tag{7.1}$$

to indicate whether points $x, y \in \mathcal{X}$ lie within distance r, where $r > 0$ is a *scale* parameter. The *Gram* matrix is the $n \times n$ matrix K with elements $K_{ij} := k(x_i, x_j)$. For $x \in X$, define *cluster* $C(x) := X \cap B_r(x)$ as the set of points $y \in X$ within distance r from x. If there is a *weight vector* $\mathbf{w} \in \mathbb{R}^n$ such that $K\mathbf{w} > 0$, then all points $x \in X$ lie in $C(x_j)$ for some point $x_j \in X$ with positive weight $w_j > 0$. An ℓ_1 penalty encouraging sparsity yields the problem

A weighted penalty also favors large clusters.

$$\min_{\mathbf{w} \in \mathbb{R}^n} \|\mathbf{w}\|_1 \tag{7.2}$$

$$\text{subject to } K\mathbf{w} \geq 1, \tag{7.3}$$

easily reduced to linear programming. Given the optimal solution $\mathbf{w}^\star$, the *codebook* $Q(X)$ is defined as

$$Q(X) := \{x_j \in X : w_j^\star > 0\}. \tag{7.4}$$

We refer to points in $Q(X)$ as *cluster centers*. Clearly, $Q(X) \subseteq X$, and the collection of clusters $C(x)$ for $x \in Q(X)$ is a *cover* for X[2] but *not* a partition. That is, clusters are *overlapping*. By construction, the *distortion* induced by $Q(X)$ is upper bounded by r. The number of clusters is adjusted accordingly.

Overlap is useful for both geo- and visual clustering, especially in case of gradual view transitions.

7.3 VIEW CLUSTERING

We also refer to photos as images, *or* views.

Now let $\mathcal{X}$ and X refer to *photos*. Each photo $x \in X$ is represented by *location* (latitude and longitude) and a set of local features, including position, local shape and visual word over a vocabulary $\mathcal{W}$.

GEO-CLUSTERING We apply KVQ to X in metric space $(\mathcal{X}, d_g)$ with scale parameter r_g, where metric $d_g : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ is the *geodesic distance* on the Earth surface[3]. Let $Q_g(X)$ be the resulting *geo-codebook*,

In practice, we use spatial bucketing *on a uniform grid and keep one sample per bucket.*

2 A subsequent *pruning* step removes from $Q(X)$ any point such that the cluster collection of the remaining points is still a cover for X.

3 http://en.wikipedia.org/wiki/Great-circle_distance

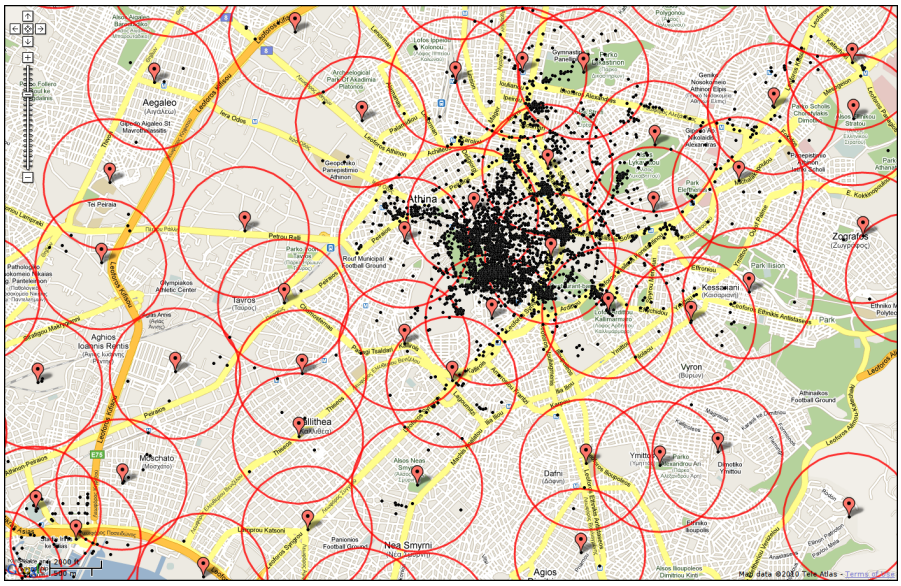


Figure 7.1: Map of Athens illustrating *geo-clusters* for $r_g = 700$m. Black dots, red markers and red circles stand for photos, codebook vectors and cluster boundaries, respectively.

and $C_g(x)$ the *geo-cluster* of $x \in X$. As shown in Figure 7.1, overlapping helps keep dense areas in few clusters for subsequent visual clustering: Photos taken even 1km away from a landmark may be included in the same cluster.

VISUAL CLUSTERING As in [408], we say that any two photos $x, y \in \mathcal{X}$ are *connected* if at least one rigid object is visible in both, possibly under different viewpoints. A *scene* is a subset $S \subseteq X$ of connected photos. We use FSM [341] to match a pair of images x, y under a geometric model. The number of *inliers* is used as *visual similarity*. The visual metric d_v can be any non-increasing function of the similarity, since kernel function k is discrete (7.1). We apply KVQ to each geo-cluster $C_g(x)$ for $x \in Q_g(X)$ in space $(\mathcal{X}, d_v)$ with scale parameter r_v. An example is shown in Figure 7.2. Let $Q_v(G)$ be the *visual codebook* of geo-cluster G, and $C_v(y)$ the *visual cluster* of $y \in G$. The complete codebook $Q(X)$ is the union over all geo-clusters

The model may vary from 3- to 5-DoF.

The objective is not summarization or canonical view selection [408], but to align images in each cluster.

$$Q(X) := \bigcup_{x \in Q_g(X)} Q_v(C_g(x)). \tag{7.5}$$

The bottleneck is the construction of Gram matrix K, which is quadratic in n. The same complexity appears in [120, 408, 514]; other options are to use small spatial tiles of 200m [349] or to *not* use local features at all [220]. Indexing is efficient enough to even work without geo-clustering [76], but then isolated photos are unlikely to be discovered. Our solution is *geo-cluster specific* indexing: We use an inverted file indexed by both visual word and geo-cluster. Given a query image in geo-cluster G, we find all connected images $x \in G$ in constant time. Computing a sparse K is now linear in $|G|$.

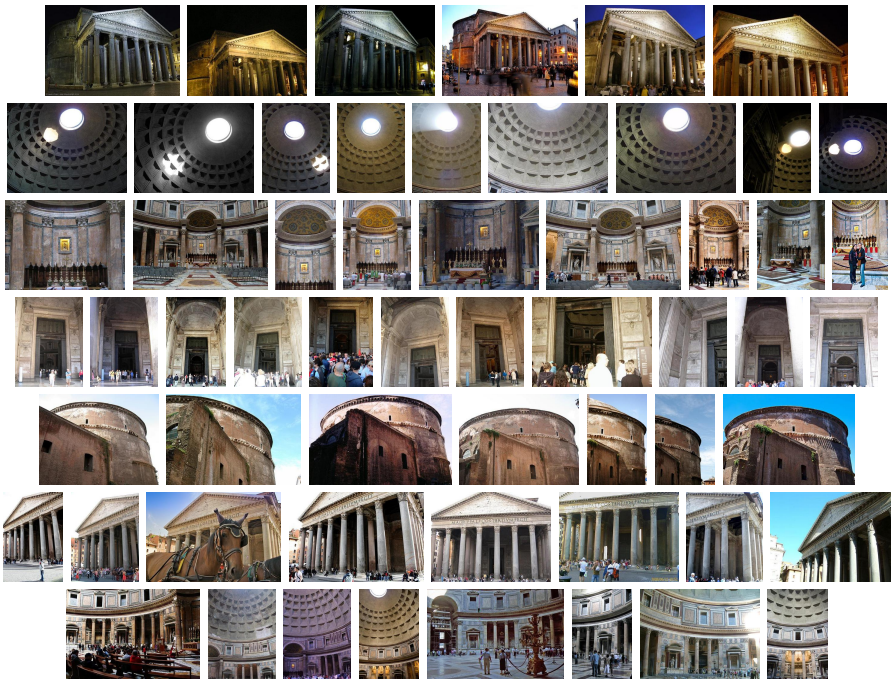


Figure 7.2: Photos in a sample of *visual clusters* from Pantheon, Rome, one cluster per row. The first image in each cluster is its center.

7.4 SCENE MAPS

We align all images in a visual cluster using a homography model and construct a *scene map*, a 2d spatial map of features associated to different views of the same scene. Scene maps are then used directly for retrieval.

VIEW ALIGNMENT During visual clustering, images are geometrically verified by FSM [341]. For each pair of matching images (x, y) in a geo-cluster, we store the best model T_{xy} that transforms y to x. Each image $x \in Q(X)$ is then treated as a *reference* in its visual cluster $C_v(x)$. We now align each image $y \in C_v(x)$ to x and compute a 3×3 homography matrix H_{xy}, starting from the stored model T_{xy} and using the "iterative" method of *Locally Optimized RANSAC* (LO-RANSAC) [74]. An example is shown in Figure 7.3.

SPATIAL CLUSTERING For each reference image $x \in Q(X)$ and corresponding visual cluster $C_v(x)$ we collect a set of features $\mathcal{P}_w(x)$ per visual word $w \in \mathcal{W}$ as the union of features over all images $y \in C_v(x)$, after aligning with the reference

$$\mathcal{P}_w(x) := \bigcup_{y \in C_v(x)} \{H_{xy}F(p) : p \in P_w(y)\}. \tag{7.6}$$

Here, $P_w(y)$ is the set of local features of image y assigned to visual word $w \in \mathcal{W}$, and 3×3 matrix $F(p)$ expresses the position and local shape of feature p, as in (4.1).

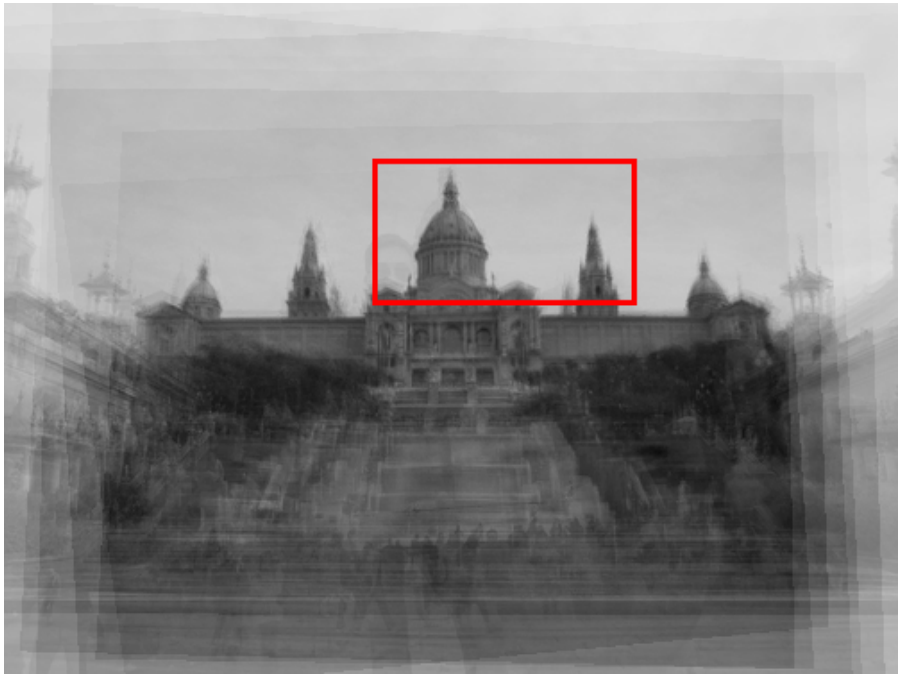


Figure 7.3: *View alignment* of 10 photos of Palau Nacional, Montjuic, Barcelona, in a visual cluster.

This feature collection bears similarities to *view clustering* for 3D object recognition [275], *feature tracks* for extracting *canonical views* [408] of a scene, visual cluster alignment for *landmark detection* [120] and a latent model for *Query Expansion* (QE) [75]. In our case, the objective is a compact representation of all collections $\mathcal{P}_w(x)$ to be used directly for retrieval. Unlike [75], the construction is offline. We want to group local features from all image regions unlike [120], and control the distance between features in a group, unlike [275, 408].

Ideally, a query should match a scene map whenever it matches any single image in the map.

Figure 7.4: *Scene map*: Features within red box of Figure 7.3, before (left) and after (right) spatial clustering. Colored by visual word.

This gives rise to yet another use of KVQ for *spatial clustering*, which we apply separately to the *position* components of each $\mathcal{P}_w(x)$ in the image plane $\mathbb{R}^2$ with scale parameter r_s. The *scene map* $S(x)$ is the collection of the resulting *spatial codebooks* $Q_s(\mathcal{P}_w(x))$ over all visual words $w \in \mathcal{W}$. An example is shown in Figure 7.4.

We use $r_s = \theta$, the error threshold used in spatial matching.

INDEXING AND RETRIEVAL A scene map has exactly the same representation as a single image, *i.e.*, a set of features. We therefore treat

scene maps as images for indexing and retrieval. In particular, $S(x)$ contains $|Q_s(\mathcal{P}_w(x))|$ features assigned to visual word w. We index scene maps by visual word in an inverted file using these cardinalities as a term frequency vector. At query time, all images $y \in C_v(x)$ are ranked at the same position as the corresponding scene map $S(x)$, without verifying them individually.

This makes it possible to retrieve images that would not match by themselves, increasing recall.

7.5 EXPERIMENTS

SETUP We evaluate our method on *European Cities 1M*[4], a one-million urban image dataset that we contribute [19], comprising a test set referred to as *Barcelona* and a 900k distractor set depict urban scenery like the test set. We use SURF features and descriptors [31] and a 75k vocabulary learned by flat k-means [341] on an independent set. We compare against baseline BoW and two QE [75] variants: QE_1, re-querying using the retrieved results and merging for three iterations; and QE_2, creating a scene map using the retrieved results and re-querying once more. We evaluate performance via mAP.

Indexing by scene maps takes 1.2GB instead of 1.61GB for the baseline, a 25% compression.

METHOD	AVG. QUERY TIME	mAP
Baseline BoW	1.03s	0.577
QE_1	20.3s	0.757
QE_2	2.51s	0.620
Scene maps	1.29s	0.807

Table 7.1: Average query time and mAP of the four benchmarked methods on the *European Cities 1M* dataset including all distractors.

RESULTS As shown in Table 7.1, our method outperforms all others, even QE_1: The expanded set comes from entire dataset in QE_1, while scene maps are constructed by querying a single geo-cluster. Scene maps are only slightly slower than the baseline, which is due to spatial matching on more features, while QE is even slower.

7.6 APPLICATION: VIRAL

The method introduced here [19] and others are made accessible through our online application VIRaL[5] [215], a content-based image search engine. Given a single query image, it retrieves visually similar images from its database and estimates where the photo is taken on a map. It suggests tags, identifies known landmarks or points of interest, and provides links to relevant Wikipedia articles.

VIRaL uses a dataset of 2.7M Flickr[6] images from 44 cities around the world, along with their metadata (*i.e.* geographic location, user

The dataset is crawled from Flickr by requesting geo-tagged photos within a bounding box of city centers.

4 http://image.ntua.gr/iva/datasets/ec1m/
5 http://viral.image.ntua.gr/
6 https://www.flickr.com/

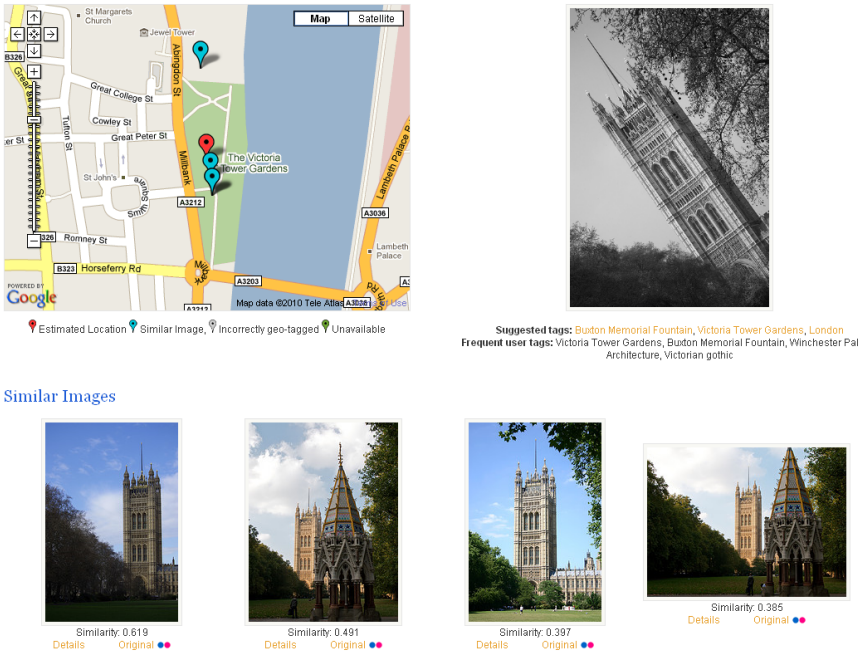


Figure 7.5: VIRaL response to a query image. (Top left) Map depicting locations of similar images (blue markers) and estimated location of the query image (red). (Top right) Query image with sets of frequent and suggested tags, the latter linking to Wikipedia articles. (Bottom) Retrieved visually similar images.

tags, image title and description). It also uses databases of landmarks and points of interest from Wikipedia[7] and GeoNames[8].

Quantitative evaluation of location and landmark recognition is conducted [215].

Figure 7.5 shows the result of a VIRaL query. The suggested tags are all correct and automatically linked to Wikipedia. Landmark recognition is very accurate because it relies on different sources of information. In particular, it compares Flickr metadata and geo-tags to landmark names and coordinates in landmark databases.

VIRaL *Explore*[9] enables browsing of the entire VIRaL image collection on the world map. Starting in a given city, it places icons of grouped photos, along with landmark names and links to Wikipedia, if applicable. The VIRaL collection is processed offline to identify groups of photos depicting the same object, building, or scene, using our *scene maps* [19]. Most popular groups are shown on the map, according to the zoom level.

VIRaL *Routes*[10] offers a unique browsing experience of photo collections. Personal photo sets are processed offline to identify where they were taken and group them by scene. A route is then laid out on the map, showing icons of visited places. Route construction is based

7 http://de.wikipedia.org/wiki/Wikipedia:WikiProjekt_Georeferenzierung/Wikipedia-World/en
8 http://www.geonames.org/export/wikipedia-webservice.html#wikipediaSearch
9 http://viral.image.ntua.gr/?explore
10 http://viral.image.ntua.gr/?routes

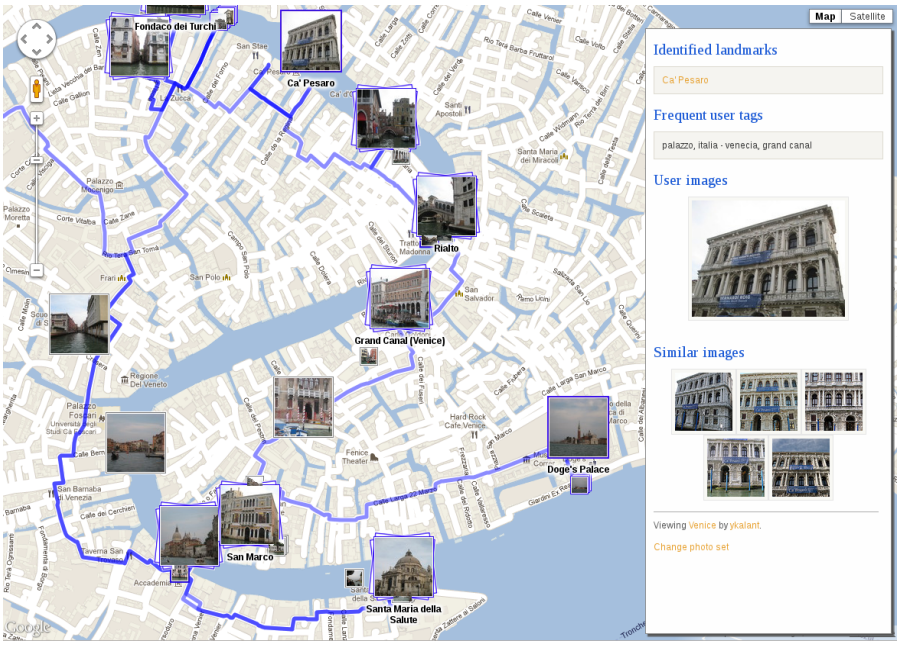


Figure 7.6: VIRaL *Routes*. A personal photo set from a trip to Venice is displayed on a map. The route is automatically inferred.

on geographic coordinates and timestamps. The density of photos is adjusted to the zoom level. Figure 7.6 depicts an example.

7.7 DISCUSSION

Typically, sub-linear indexing is not exploited in landmark recognition applications, while geo-tags are not exploited in large scale 3d reconstruction applications. We combine both here, along with a novel scene representation that is directly encoded in our retrieval engine. The result is significant gain in retrieval performance, even compared to query expansion methods, at the cost of a slight increase in query time. Index space is also reduced. Contrary to landmark recognition applications, we can still retrieve any isolated image, allowing location recognition at any region where geo-tagged photos are available. We also recognize landmarks and points of interest by comparing location, photo title and frequent tags to landmark databases.

More results, both for landmark and non-landmark scenes, can be found online in our project homepage[11].

11 http://www.image.ntua.gr/iva/research/scene_maps/

Part II

EXPLORING DEEPER

Building on *Convolutional Neural Network* (CNN) features, we study visual representations and matching processes for exploring visual data, including instance-level visual search and object discovery, focusing on the manifold structure of the feature space.

8 OUTLINE

This chapter serves as an outline or road map of Part II. *We present historical and more recent background on deep learning for visual representations developed in the 2010s. In this context, we position our own contributions developed in 2017-2019. Building on* Convolutional Neural Network *(CNN) features, our work addresses visual representations and matching processes for exploring visual data, including instance-level visual search and object discovery, focusing on the manifold structure of the feature space. We outline the structure of* Part II *in terms of methods, key publications and corresponding chapters.*

8.1 CONTEXT

The work of Krizhevsky *et al*. known as AlexNet [235] in 2012 is arguably a landmark of machine learning research in the 2010s. Even if none of the ideas is entirely new, Krizhevsky *et al*. put together four elements that make learning visual representations from raw data "really" work at scale:

1. a CNN architecture [118, 250] trained on a supervised classification task using *Stochastic Gradient Descent* (SGD) [371] with *momentum* and *back-propagation* [376];
2. the ILSVRC dataset [377], comprising more than one million images, each with a class label over 1000 classes;
3. a massively parallel implementation on GPUs [67]; and
4. the ReLU non-saturating nonlinearity [313], facilitating backward gradient flow through a depth of 8 layers.

Representations learned on ILSVRC yield astounding performance on a multitude of category-level or instance-level computer vision tasks [100, 398]. In 2015-16, three more elements unleash the full power of deep architectures beyond 100 layers: careful *initialization* [164], activation *normalization* [186] and *skip connections* [165].

In *instance-level search*, different views of the same object should be mapped to similar representations. Shallow representations are quickly outperformed by CNNs trained on ILSVRC and then fine-tuned to new domains, supervised by noisy class labels [25] or by the very same shallow representations [136, 351].

Our contributions.

In this context, Part II presents part of our work carried out in the period 2017-2019, which addresses visual representations and matching processes for exploring visual data, including instance-level visual search and object discovery, focusing on the manifold structure of the feature space.

Our contributions consist of:

1. making advances in *manifold search* over global or regional CNN representations seen as graph filtering, including *spatial* [194], *spectral* [189] and *hybrid* [190];
2. revisiting *spatial matching* with local features detected on CNN activations in the simplest possible way [405]; and
3. *discovering objects* from CNN activations over an unlabeled image collection, seen again as graph filtering [406, 407].

Importantly, as a result of discovering objects in a collection, we improve the representation of each image itself by focusing on objects and suppressing background clutter.

8.2 BACKGROUND AND CONTRIBUTIONS

Due to Minsky and Papert [300], perceptron *is known today as a linear classifier and an algorithm for that classifier.*

NEURAL NETWORKS Rosenblatt introduces the term *perceptron* in the 1960s [375], referring to a wide range of network architectures, learning algorithms and hardware implementations. Although not widely appreciated even today, he lays the foundations of modern neural networks by studying early forms of *multi-layer networks*, continuous activation functions, back-propagating errors, convolution, skip connections, recurrent networks, selective attention, program learning, and multimodality. The *perceptron algorithm* is an instance of (online) *Stochastic Gradient Descent* (SGD), introduced in 1951 by Robbins and Monro [371].

One of its first applications to machine learning is maybe by Werbos [480].

Paramount to the optimization of network parameters is the efficient evaluation of derivatives over arbitrary architectures. Given an arbitrary computational graph and a set of values for the input variables, *automatic differentiation* [32, 479] allows the construction of another graph for the evaluation of derivatives at the given inputs. Its *reverse accumulation mode* becomes widely known as *back-propagation* in 1986 by Rumelhart *et al*. [376], who also advocate for optimization strategies that are standard today, including *random initialization* and *batch* or *online gradient descent* with *momentum*.

Convolutional networks.

Inspired by the findings of Hubel and Wiesel on the visual nervous system in 1959 [185], Fukushima introduces *neocognitron* in 1980 [118], a network consisting of alternating layers of simple and complex cells that learn a feature hierarchy. The former perform *convolution* with parameters that are learned in an unsupervised fashion and the latter spatial average *pooling* and *sub-sampling* without parameters, introducing invariance to deformations. LeCun *et al*. study a similar architecture in the 1990s [250, 251], rather learning the parameters by SGD and back-propagation on a classification loss function, establishing the term *Convolutional Neural Network* (CNN) and advocating end-to-end feature learning from raw data. Serre *et al*. [395] establish *max-pooling* as the operator of choice for spatial pooling in 2005.

DEEP LEARNING During the 2000s, unsupervised layer-wise *pre-training* is used to initialize *e.g.* 4-layer networks [38, 173, 356] and massively parallel CNN implementations appear on *Graphics Processing Units* (GPUs) [67, 79], but experiments are limited to tiny images. In 2012, Krizhevsky *et al.* [235] use a two-GPU implementation to learn an 8-layer CNN from random initialization on the 1000-class classification task of the *ImageNet Large-Scale Visual Recognition Challenge* (ILSVRC) [377], having a training set of 1.2M high-resolution images. This network is now named *AlexNet* after Alex Krizhevsky.

AlexNet *outperforms all previous methods by a large margin on the ILSVRC 2012 competition.*

Among the most influential design choices of AlexNet is that of the *Rectified Linear Unit* (ReLU) nonlinearity [313] over $\tanh$ (or sigmoid). In fact, ReLU is similar to a flipped version of the loss function of three well-known binary classifiers: the perceptron, logistic regression (sigmoid + cross-entropy), and SVM (hinge function) [44]. These functions are easy to optimize because they are not saturating. In retrospect, it becomes clear that saturating nonlinearities like sigmoid would not allow enough backward gradient flow for deep networks to learn, at least without skip connections [315].

ReLU is used at least as early as 1980 by Fukushima [118].

In *deep learning*, "depth" refers to the number of layers. It is often admitted that improvements come from just "stacking more layers" [409]. In 2014, a network named after the *Visual Geometry Group* (VGG) [409] and another after the film *Inception* [423] reach 19 and 22 layers respectively, the latter winning ILSVRC 2014. However, both face difficulties at training. As a workaround, the former uses pre-trained layers of shallow models to initialize deeper ones, while the latter uses auxiliary classifiers at training, attached to intermediate layers.

For the same number of units, deep networks partition the input space into exponentially more linear regions than shallow ones [303, 428].

The difficulties encountered at such depth are related to the problem of *exploding/vanishing* gradients [129]. A solution is careful weight initialization, which is studied in the linear regime of nonlinearities by Glorot and Bengio in 2010 [129] and for ReLU by He *et al.* [164] in 2015, who train a 30-layer network from scratch. A more effective solution is *batch normalization*, an operation introduced by Ioffe and Szegedy also in 2015 [186], which normalizes activations of all layers using mini-batch statistics. Apart from preventing exploding/vanishing gradients and allowing higher learning rates, it reduces the need for regularization, *e.g.* by weight decay or dropout [172].

Learning long-term dependencies in recurrent networks suffers from the same problem [40].

With all problems resolved in training a deep network from scratch, the principle of "stacking more layers" is challenged when *e.g.* 56-layer networks are found inferior to 20-layer networks [165]. Developed by He *et al.* in 2015 [165], reaching 152 layers and winning ILSVRC 2015, the *Residual Network* (ResNet) addresses this issue by introducing *skip connections* to the standard chain-structured network architecture. While skip connections date back at least to Rosenblatt [375], it is with ResNet that they become a key ingredient of modern deep network architectures, including Inception-ResNet [422] and the *Densely Connected Network* (DenseNet) [183].

Skip connections "convexify" the loss landscape [257] and make local minima equally good [315].

Apart from the astounding performance of CNNs, deep learning quickly develops into a "Swiss-army knife" for most computer vision tasks [100, 398], allowing training of different modules on different

datasets and tasks, combining modules into new architectures, fine-tuning in several stages, jointly training modules on several tasks, defining and solving new tasks by proper supervision and loss functions, self-learning on unlabeled data and fictitious tasks, with imagination being the only limit.

Object detection.

As a concrete example, Girshick *et al*. introduce *Regions with CNN features* (R-CNN) [128] for *object detection*. They use AlexNet [235], as pre-trained on ILSVRC and fine-tuned on the target classes with its last *Fully-Connected* (FC) layer removed, to replace shallow representations in an pipeline involving *region proposals* [450, 524] and an SVM classifier. Unfortunately, this uses the network thousands of times, once for each region. He *et al*. [163] rather use the network only once on the entire image and extract regional features by spatial *max-pooling* on projected regions on the activations of the last convolutional layer. Based on this idea, Girshick introduces *Fast* R-CNN [127], replacing the SVM classifier by new FC layers, while Ren *et al*. extend to *Faster* R-CNN [367], replacing hand-crafted proposals by a *Region Proposal Network* (RPN) and training everything end-to-end.

LEARNING FOR IMAGE RETRIEVAL A CNN of *e.g*. 101 layers learned on a category-level classification task can be seen as a linear classifier on top of a representation extracted by the first 100 layers. The representation space can be used for similarity search as already exhibited on few examples for FC layer 7 of AlexNet [235], retrieving images of the same class as the queries.

The term neural codes *[25] indicates that using a representation out of a neural network for similarity search appeared exotic at the time.*

Learning for *instance-level* tasks is possible by treating *e.g*. each building or landmark as a class and using a classification loss like *cross-entropy*. Babenko *et al*. [25] use AlexNet, as pre-trained on ILSVRC, and fine-tune it like that on a Landmarks dataset. They evaluate for the first time instance-level retrieval on a representation learned on raw data and they find that FC layer 6 performs best.

Spatial pooling on convolutional activations treats feature channels as pattern detectors and can be interpreted as a histogram over such patterns, like BoW.

It easily escapes the reader that experiments on convolutional layer 5 by Babenko *et al*. [25] assume *flattening* of the 3d *convolutional activation* tensor into a vector of length 9,216, which is a non-invariant representation. The next obvious attempt by Azizpour *et al*. [21] is to drop FC layers and apply *spatial max-pooling* to the activation tensors of the last convolutional layer. Then, Razavian *et al*. [363, 364] split images into patches on a grid, feed each patch into a CNN and match the resulting regional vector representations exhaustively pairwise. With a lower-dimensional representation based on a modified version of VGG, this solution outperforms for the first time the best known representation based on hand-crafted local descriptors, which happens to be our ASMK [436].

This progression is the same as the progression from R-CNN [128] to Fast R-CNN [127] for object detection.

The matching process by Razavian *et al*. [364] is invariant but expensive. The extraction process is expensive too. With the discriminative power of CNNs, a global representation with a single pass through the network is more appealing. Tolias *et al*. [439] obtain a single activation tensor from the last convolutional layer corresponding to the entire image and, using a similar spatial grid as [364], per-

form a two-level spatial pooling called *Regional Maximum Activation of Convolutions* (R-MAC): max-pooling over cells on the grid followed by average pooling. This is found superior to *Maximum Activation of Convolutions* (MAC), a name given in retrospect to global max-pooling as introduced by Azizpour *et al*. [21].

To adapt representations for the image retrieval task, rather than classifying images like Babenko *et al*. [25], requires learning to *compare* or *rank* them. This is the task of *metric learning*, which is discussed in more detail in Section 13.2. The two most common methods are to consider images in *pairs* or *triplets* with the *contrastive* [72] or *triplet loss* [468], respectively. But how are we to define labels on pairs or triplets of examples? This is certainly a more difficult task than a label per example as in classification.

Gordo *et al*. [136, 137] clean the the Landmark dataset by Babenko *et al*. [25] by pairwise matching using hand-crafted local descriptors and spatial verification, then finding the different profiles of each landmark (*e.g*. inside/outside) as connected components of a graph and keeping the largest one. On this clean dataset, they learn fine-grained similarity with a network pre-trained on ILSVRC by attaching R-MAC pooling [439] and fine-tuning it using the *triplet loss* [468].

Gordo et al. *also use an RPN [367] to define the regions, as discussed below.*

Concurrently, Radenović *et al*. [351, 352] work on an independent unlabeled dataset of urban scenes. They apply a similar pairwise matching process as well as a SfM pipeline, resulting in 3d models and camera positions per image. They use 3d models as labels and select hard positive and negative pairs using camera positions and matching scores. They attach MAC pooling [439] to a pre-trained network and fine-tune it using the *contrastive loss* [72]. They extend to *Generalized Mean* (GeM) pooling [352], which allows for contributions from more than one spatial location.

> *We focus on instance-level tasks, adopting learned CNN representations in* Part II*. The representations are global, regional or local, taking advantage of convolutional activations. Then,* Part III *addresses learning representations for both instance-level and category-level tasks. Our own contribution to metric learning for image retrieval is the subject of Chapter 14.*

SEARCHING ON MANIFOLDS Both Gordo *et al*. [137] and Radenović *et al*. [352], using a global vector representation per image, outperform representations of thousands of hand-crafted local descriptors per image, including our ASMK [436]. This is a game changer. Not only is image retrieval reduced to nearest neighbor search, but searching on image manifolds becomes a reality. To appreciate how, we go back to studies in psychology and social sciences in the 1950s.

E.g., *a vector of length 512 or 2,048 for VGG or ResNet network backbone respectively.*

In 1949, Seeley [393] studies a nonnegative square matrix W with elements representing endorsement between entities, for instance one (zero) meaning "like" ("don't like"). He observes that it is important to be liked by someone who is in turn liked a lot, and so on. Recursively, he defines an index of importance or *centrality* over entities,

given by an *eigenvector* of W. In 1953, Katz [219] uses W to represent a directed graph over entities and a *power series* of W to measure the number of paths between entities, representing recursive endorsements. He introduces a *damping* factor α to ensure convergence of the series and expresses the power series as a *matrix inversion*. In 1965, Hubbell [184] introduces a *boundary condition* or *initial preference* vector to estimate the *similarity* of entities given the graph.

The column sum of this power series is known as Katz centrality.

Vigna [458] puts together all this line of work, calling it *spectral ranking*, connecting the eigenvector and matrix inversion formulations and listing a number of rediscoveries of the same theory in different fields. Two more rediscoveries include Zhou *et al*. [517], who use a symmetric form of W to represent similarity and an iterative process to avoid matrix inversion; and Pan *et al*. [330], who give the name *Random Walk with Restart* (RWR) to the same process. Both treat the initial preference vector as a *query* representation and use the iterative process to estimate the similarity of entities to the query given the graph, which represents a *manifold*.

The most well-known is arguably PageRank by Page et al. *[329].*

Several such iterative processes, called *diffusion processes*, are studied for image retrieval by Donoser and Bischof [101]. To maintain efficiency, graphs need to be small and image representations simple. For large-scale image retrieval using local descriptors, it is more common to apply *Query Expansion* (QE) [75] online, without a graph. In its simplest form, QE is non-iterative and can be thought of as just the first term of the power series formulation [219].

QE originates in text retrieval [53].

With powerful and compact global CNN vector representations, it is easy to construct a *k-Nearest Neighbor* (*k*-NN) graph of a large dataset offline. Following the *regional matching* of Razavian *et al*. [364], one can even construct a *k*-NN graph of image regions, having few (*e.g*. 20) regions per image. This graph represents the image (region) manifold and diffusion explores this manifold online.

Our regional diffusion *[194].*

Our *regional diffusion* [194] is the first method to apply diffusion at large scale on such a representation. We return to the matrix inversion formulation and we observe that it is only needed to solve a linear system. We reveal that the RWR process [330, 517] is an instance of the *Jacobi* [149] solver and we rather use *Conjugate Gradient* (CG) [317], which is also iterative but more efficient.

It turns out that the mapping from the input query vector to the output manifold similarity vector is a linear *graph filtering* operation, *smoothing* in particular, as defined in GSP [383, 401]. We study this analogy between manifold search and smoothing in our *Fast Spectral Ranking* (FSR) [189]. We introduce a scalable offline computation of an approximate Fourier basis of the graph and perform filtering online in the *frequency domain*, which is extremely fast. The basis is a low-dimensional and sparse *explicit embedding* of the diffusion similarity kernel. Alternatively, kernel PCA [49, 389] would need to compute the kernel first, which we do not.

Our Fast Spectral Ranking *(FSR) [189].*

The combination of powerful CNN representations and manifold search nearly solves the *Oxford* [341] and *Paris* [342] image retrieval benchmarks. To facilitate further research, we introduce the *Revis-*

ited Oxford and Paris (RevOP) benchmark [350]. Among other improvements, we create a new set of one million challenging distractors. At this scale, it turns out that CG [194] and FSR [189] take too much time or space respectively. Our *hybrid diffusion* [190] allows full control of the space-time trade-off between the two extremes.

Our RevOP [350].

Our hybrid diffusion *[190].*

LOCAL FEATURES AND SPATIAL MATCHING Convolutional activations on images are 3d tensors, where the two dimensions correspond to spatial dimensions of the input image, though at lower resolution, and the third dimension to features (descriptors). They can be thought of as a set of dense local descriptors per spatial location. Spatial pooling yields global or regional descriptors that can be used to estimate similarity, but, can the original activations be used to estimate accurate correspondences?

Or, as a collection of 2d activation maps, one per channel (see below).

This question is studied and answered in the affirmative by Long *et al*. in 2014 [272]: Although they have large receptive fields, those descriptors carry local information at a fine scale. Hence, pairwise matching can yield dense correspondence and alignment between two views. Importantly, these views are not necessarily of the same object or scene as discussed in Section 2.2. They can be of two different instances of the same category.

However, again as discussed in Section 2.2, we know that not all locations are equally good for establishing correspondences. Sparse local features remain the best choice in handling occlusion and estimating relative pose in wide-baseline matching. In instance-level retrieval, spatial matching is a key ingredient of methods based on hand-crafted local descriptors. CNN representations already encode geometry via interleaved convolution and pooling. Can spatial matching on sparse local features help further, and how?

One of the earliest learned local feature detectors is by Dias *et al*. [95] in 1995, using a 3-layer neural network to detect corners on 8×8 patches after edge detection and thinning. In a more modern setting, Verdie *et al*. [456] introduce in 2014 the *Temporally Invariant Learned DEtector* (TILDE), a piece-wise linear regressor mapping input images to a score map where features are detected at local maxima. The *Learned Invariant Feature Transform* (LIFT) [498] integrates TILDE into a complete pipeline comprising the detector, patch cropping, orientation estimation and descriptor extraction. *SuperPoint* [94] operates on the entire image instead with a single and deeper network backbone encoder and two different upsampling decoders serving as detector and descriptor.

The regressor is essentially a 2-layer CNN with max-out *[134] nonlinearity.*

Most learned detectors need ground truth coordinates or correspondences, which are commonly provided by hand-crafted detectors and matching processes on carefully designed datasets. An exception is *DEep Local Features* (DELF) by Noh *et al*. [318], which is trained with image-level labels only. Following again a single network backbone, an *attention* branch is selecting the spatial locations where descriptors are to be extracted from the activation tensor.

Still, all learned detectors operate on a single 2d score map and detect point features, without local shape. However, Tolias *et al*. [439] illustrate that the spatial locations of maximum activation per channel can yield correspondences, while *Generalized Mean* (GeM) pooling [352] suggests that more locations may be important.

Building on these findings, we observe that local features *emerge* on activation maps without particular effort, *i.e*., without modifying the network architecture and *without training*. Simply put, we see the activation 3d tensor as a collection of 2d maps, one per channel, and we detect local features at local maxima of these 2d maps, independently per channel. By fitting affine regions, we also equip local features with geometric information to allow generation of transformation hypotheses from single correspondences. Our *Deep Spatial Matching* (DSM) [405] applies these ideas to geometry verification for instance-level image retrieval. By treating activation channels as *visual words*, we do not even use local descriptors or vocabularies.

Our Deep Spatial Matching *(DSM) [405].*

VISUAL ATTENTION AND OBJECT DISCOVERY Primates, including human, use attention mechanisms to analyze visual stimuli. In the 1960s, Yarbus [496] uses an eye-tracking device to study the role of *eye movements* in visual perception. In 1980, Treisman and Gelade [444] study *visual search* for targets among distractors. Their findings suggest that in a first *pre-attentive* stage, simple features are processed in parallel, while in a second *attentive* stage, the focus of attention is shifted to different locations in a sequence.

This latter form of attention is independent of eye movements.

Computational models of *selective visual attention* follow. In 1985, Koch and Ullman [229] introduce a model comprising a *saliency map* that combines individual feature maps into a global conspicuity measure and a *Winner-Take-All* (WTA) mechanism that sequentially routes the properties at the most conspicuous location from individual feature maps to a central representation. Itti *et al*. extend this model in 1998 [195] by introducing *competition* among different features, locations and scales in the construction of the saliency map.

WTA is implemented via softmax over spatial locations.

Modern two-stage category-level *object detectors* like the R-CNN family [127, 128] are reminiscent of this approach: *Region proposals* [5] obtained by bottom-up grouping like *selective search* [450] and *edge boxes* [524] are used to focus the attention of a classifier on top of deep convolutional features extracted densely over the entire visual field. RPN [367] is a *learned* attention mechanism, in the form of a class-agnostic detector applied densely. One-stage detectors are applied densely too, but with appropriate weighting [263].

From attention to detection and back.

Compact *global representations* for instance-level tasks can be obtained by pooling over *attended regions* too, replacing the fixed regions of R-MAC [439]: for instance, using selective search [306], RPN as pretrained on ILSVRC [382], or RPN fine-tuned for retrieval [136]. An alternative is weighted average pooling using a *saliency map*. This may be supervised, *e.g*. learning to predict eye fixations [302] or from labeled nearest neighbors [89]; or unsupervised, *e.g*. by focusing on channels

RPN requires ground-truth bounding boxes.

All these solutions are either hand-crafted or require localization or image-level ground-truth.

with sparse activations [214] or implicitly learning an attention layer on image-level labels [318].

The global representation should focus on foreground objects, suppressing clutter and occlusions. Ideally, the definition of foreground should depend on the *target dataset*: Objects appearing frequently are likely to be foreground. In this case, any learning should be *fully unsupervised*. In instance-level tasks using local descriptors, this can be done by pairwise matching and spatial verification over the dataset and selecting inlier features per image [446].

For instance, focusing on animals is not interesting for a target dataset of city scenes.

In category-level tasks, *object discovery* is the task of discovering the categories and localizing foreground objects per category in a fully unsupervised way [412]. Using region proposals, one solution is again pairwise spatial matching, interleaved with region selection [70]. By representing pairwise region interactions by a graph, it makes sense to use a graph centrality measure to identify regions appearing frequently, hence likely to depict foreground [222].

From attention to discovery and back.

Using CNN representations, our *Graph-based Object Discovery* (GOD) [406, 407] is a visual attention mechanism learned in a fully unsupervised way on the target dataset for instance-level retrieval. Beginning with a saliency map capturing discriminative patterns based on convolutional activations alone [214], we discover common patterns by graph centrality on a k-NN region graph. We thus learn a non-parametric model of patterns that are both discriminative and common in the dataset. The result is a global representation that focuses on objects and suppresses background clutter.

Our Graph-based Object Discovery *(GOD) [406, 407].*

We detect region proposals from saliency maps using our EGM [18].

8.3 STRUCTURE

Chapter 9 provides background on *graph filtering*, including notation, definitions and interpretations. This background is then used in Chapters 10, 12, 14, 15 and 17.

Chapter 10 addresses manifold search. It presents two solutions, *regional diffusion* [194] and *Fast Spectral Ranking* (FSR) [189]. It also briefly discusses recent work on revisiting a popular benchmark [350] and a hybrid solution that is more appropriate at large scale [190].

Chapter 11 revisits spatial matching, now equipped with CNN representations. It presents *Deep Spatial Matching* (DSM) [405], which extracts local features and quantized representations directly from CNN activations, without descriptors or vocabularies.

Finally, Chapter 12 addresses unsupervised object discovery from unlabeled image collections. It presents *Graph-based Object Discovery* (GOD) [406, 407], which discovers discriminative and frequent patterns and uses them to improve image representation for retrieval.

9 GRAPH FILTERING

This chapter provides background on graph filtering, *including notation, definitions and interpretations. We define a particular filter that performs smoothing over a graph. This background is used in different ways in Chapters 10, 12, 14, 15 and 17. For instance, we compute a similarity measure between two vertices of a nearest neighbor graph over a dataset. This graph represents a manifold in continuous space, hence we call this measure* manifold similarity. *Alternatively, we compute a separate similarity measure per class and use it for* classification. *Finally, we compute a measure of graph* centrality *and we smooth an image* guided *by another image.*

9.1 INTRODUCTION

Consider an *Exponential Moving Average* (EMA), given by recurrence

$$z_i := \alpha z_{i-1} + (1-\alpha) y_i \tag{9.1}$$

for $i \in \mathbb{Z}$. Here, $\mathbf{y}, \mathbf{z}$ are the input and output respectively. They are discrete-time signals: y_i denotes the sample of $\mathbf{y}$ at time i, and similarly for $\mathbf{z}$. EMA is an example of a *low-pass filter* in *signal processing* [326] and the output $\mathbf{z}$ can be regarded as a *smoothed* version of $\mathbf{y}$. Parameter $\alpha \in [0, 1)$ is a *smoothing factor*: A lower α discounts older observations faster.

What we present in this chapter is a generalization of EMA on graphs, which is the subject of *Graph Signal Processing* (GSP) [383, 401]. The generalization consists in replacing time instances with vertices of a graph and time shift (delay) with an operation called *graph shift*. This operation replaces the sample at a vertex with a weighted linear combination of the samples at its neighbors.

We use this smoothing operation in different contexts for different purposes. For instance, we compute a similarity measure between two vertices of a nearest neighbor graph over a dataset. In turn, this graph represents a manifold in continuous space, hence we call this measure *manifold similarity*. Alternatively, we compute a separate similarity measure per class and use it for *semi-supervised classification*. Finally, we compute a measure of graph *centrality* and we smooth an image *guided* by another image.

The nearest neighbor relation is taken as *reciprocal* (*mutual*), giving rise to an undirected graph with *symmetric* adjacency matrix W. This simplifies the formulation and the implementation because W is diagonalizable and a particular linear system may be solved by the *Conjugate Gradient* (CG) [317] method.

As discussed by Vigna [458] and summarized in Section 8.2, this operation has a long history originating in social sciences in the 1950s

and the most well-known form is PageRank [329]. Here we mainly follow Zhou *et al*. [517] who use a single-input operation to *rank data on manifolds* and Zhou *et al*. [516] who use a multiple-input operation applied to *semi-supervised learning*. We also refer to few elements from frequency analysis in GSP [384] and *spectral graph theory* [77].

9.2 DEFINITIONS

GRAPH We are given a weighted undirected graph G with n vertices represented by its $n \times n$ symmetric nonnegative *adjacency matrix* W. The graph contains no self-loops, *i.e*. W has zero diagonal. We define the $n \times n$ *degree matrix*

$$D := \operatorname{diag}(W\mathbf{1}), \tag{9.2}$$

where $\mathbf{1}$ is the all-ones vector, and the *symmetrically normalized adjacency matrix*

$$\mathcal{W} := D^{-1/2} W D^{-1/2}, \tag{9.3}$$

with the convention $0/0 = 0$. We also define the $n \times n$ *Laplacian* $L := D - W$ and *normalized Laplacian*

$$\mathcal{L} := D^{-1/2} L D^{-1/2} = I - \mathcal{W}. \tag{9.4}$$

Both are singular and positive-semidefinite. The eigenvalues of $\mathcal{L}$ are in the interval $[0, 2]$ [77] and those of $\mathcal{W}$ in $[-1, 1]$. Hence, if $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of $\mathcal{W}$, its *spectral radius*

$$\varrho(\mathcal{W}) := \max_i |\lambda_i| \tag{9.5}$$

is 1. Each eigenvector $\mathbf{u}$ of L associated to eigenvalue 0 is constant within a connected component of G, *e.g*., $L\mathbf{1} = D\mathbf{1} - W\mathbf{1} = \mathbf{0}$ if G is connected. The corresponding eigenvector of $\mathcal{L}$ is $D^{1/2}\mathbf{u}$.

FILTERING We define the $n \times n$ *regularized Laplacian*

$$\mathcal{L}_\alpha := (1 - \alpha)^{-1}(I - \alpha\mathcal{W}), \tag{9.6}$$

where $\alpha \in [0, 1)$ is a parameter. This matrix is positive-definite since $I - \alpha\mathcal{W} = \alpha\mathcal{L} + (1 - \alpha)I \succ \alpha\mathcal{L} \succeq 0$. Then, given an $n \times 1$ *observation vector* $\mathbf{y}$, the linear system

$$\mathcal{L}_\alpha \mathbf{z} = \mathbf{y} \tag{9.7}$$

has a unique solution $\mathbf{z}^\star := h_\alpha(\mathcal{W})\mathbf{y}$, where

$$h_\alpha(\mathcal{W}) := (1 - \alpha)(I - \alpha\mathcal{W})^{-1}. \tag{9.8}$$

The mapping $\mathbf{y} \mapsto h_\alpha(\mathcal{W})\mathbf{y}$ is a linear *graph filtering*, in particular *smoothing* of $\mathbf{y}$ on G and h_α is the *transfer function* of the filter. Similarly, given an $n \times c$ *observation matrix* Y, the linear system

$$\mathcal{L}_\alpha Z = Y \tag{9.9}$$

has a unique solution $Z^\star := h_\alpha(\mathcal{W})Y$, and the mapping $Y \mapsto h_\alpha(\mathcal{W})Y$ is a linear smoothing of Y on G.

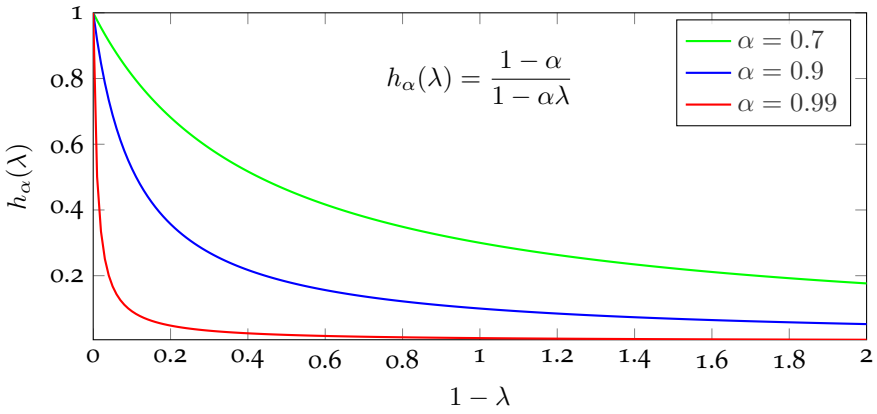


Figure 9.1: Function h_α is a *low-pass filter*: If $\lambda \in [-1, 1]$ is an eigenvalue of $\mathcal{W}$, then $1 - \lambda \in [0, 2]$ is an eigenvalue of the Laplacian $\mathcal{L}$, with values near 0 (2) representing low (high) frequencies.

9.3 INTERPRETATIONS

SPECTRAL FILTERING Being real symmetric, matrix $\mathcal{W}$ is diagonalizable:

This interpretation explains smoothing. *It is used in Section 10.4.*

$$\mathcal{W} = U\Lambda U^\top, \tag{9.10}$$

where $n \times n$ matrices U, Λ hold the eigenvectors and eigenvalues respectively of $\mathcal{W}$. Since U is orthogonal and $I - \alpha\mathcal{W}$ and $I - \alpha\Lambda$ are nonsingular, it follows that

This can be confirmed by direct multiplication of $I - \alpha\mathcal{W}$ and $U(I - \alpha\Lambda)^{-1}U^\top$.

$$h_\alpha(\mathcal{W}) = U h_\alpha(\Lambda) U^\top. \tag{9.11}$$

In fact, because Λ is diagonal, h_α applies element-wise to the eigenvalues as a scalar function

$$h_\alpha(\lambda) := \frac{1-\alpha}{1-\alpha\lambda} \tag{9.12}$$

for $\lambda \in [-1, 1]$. That is, $h_\alpha(\Lambda) = \operatorname{diag}(h_\alpha(\lambda_1), \ldots, h_\alpha(\lambda_n))$. Then, given $\mathbf{y}$, the linear mapping

$$\mathbf{y} \mapsto U h_\alpha(\Lambda) U^\top \mathbf{y} \tag{9.13}$$

has the following interpretation:

> *Seen as a signal in a space domain, $\mathbf{y}$ is mapped by $U^\top$ to the* frequency domain, *multiplied element-wise by $(h_\alpha(\lambda_1), \ldots, h_\alpha(\lambda_n))$, and mapped by U back to the space domain. The columns of U are the* Fourier basis *of the graph.*

This interpretation is understood in the context of *frequency analysis* in GSP [384], where the *Discrete Fourier Transform* (DFT) [326] is generalized by replacing the DFT matrix by $U^\top$ and its inverse by U. Indeed, the columns of the n-point inverse DFT matrix are the eigenvectors of the $n \times n$ *cyclic permutation matrix* C, which is the adjacency

Multiplication by C represents the standard time shift (delay).

matrix of a directed graph having an edge from time i to $i+1$ for $i \in \{0, \dots, n-2\}$ and edge from $n-1$ back to 0 [384].

This reveals that function h_α is a *low-pass filter*, as shown in Figure 9.1. By varying α from 0 to 1, the frequency response varies from all-pass to sharp low-pass. Hence the name *smoothing*.

This interpretation is used in Section 10.3.

RANDOM WALKS Consider the iterating process

$$\mathbf{z}^{(t)} := \alpha \mathcal{W} \mathbf{z}^{(t-1)} + (1-\alpha)\mathbf{y}. \tag{9.14}$$

for $t = 1, 2, \dots$. Regardless of the choice of $\mathbf{z}^{(0)} \in \mathbb{R}^n$, state $\mathbf{z}^{(t)}$ converges to $\mathbf{z}^\star := h_\alpha(\mathcal{W})\mathbf{y}$ as $t \to \infty$ provided $\alpha\mathcal{W}$ has spectral radius $\varrho(\alpha\mathcal{W}) < 1$ [516], which is indeed the case. Again, α controls how much $\mathbf{z}^\star$ is affected by vector $\mathbf{y}$, called *boundary condition* [458] in this context: $\mathbf{z}^\star$ equals $\mathbf{y}$ for $\alpha = 0$, while in the limit $\alpha \to 1$, $\mathbf{z}^\star$ tends to a dominant eigenvector of $\mathcal{W}$. Indeed, for $\alpha = 1$, (9.14) becomes a power iteration.

Recall that $\rho(\mathcal{W}) = 1$ and $|a| < 1$.

Similarly, given an $n \times c$ *observation matrix* Y, the iterating process

$$Z^{(t)} := \alpha \mathcal{W} Z^{(t-1)} + (1-\alpha)Y \tag{9.15}$$

for $t = 1, 2, \dots$ converges to $Z^\star := h_\alpha(\mathcal{W})Y$ as $t \to \infty$.

In the case where $\mathcal{W}$ is a row-stochastic *transition matrix* and $\mathbf{x}^{(0)}, \mathbf{y}$ express distributions over vertices, process (9.14) can be interpreted as a random walk on a (directed) graph: At each iteration a particle moves to a neighboring vertex with probability α or jumps to a vertex according to distribution $\mathbf{y}$ with probability $1-\alpha$. This is referred to as *Markov chain with restart* [47] or RWR [330]. We shall call process (9.14) RWR too, even though $\mathcal{W}$ is symmetric in our case.

This interpretation is used in Sections 15.3 and 17.4.

ENERGY MINIMIZATION The quadratic energy function

$$E_\alpha(\mathbf{z}) := \frac{1}{2}\mathbf{z}^\top \mathcal{L}_\alpha \mathbf{z} - \mathbf{y}^\top \mathbf{z}, \tag{9.16}$$

This is evident by letting the gradient $\nabla_{\mathbf{z}} E_\alpha(\mathbf{z}) = \mathcal{L}_\alpha \mathbf{z} - \mathbf{y}$ vanish.

is minimized at $\mathbf{z}^\star := \mathcal{L}_\alpha^{-1}\mathbf{y}$, that is, the unique solution of (9.7). If we expand $E_\alpha(\mathbf{z})$ using $(1-\alpha)\mathcal{L}_\alpha = \alpha\mathcal{L} + (1-\alpha)I$, we find [516] that it has the same minimizer as

$$Q_\alpha(\mathbf{z}) := \frac{\alpha}{2} \sum_{i,j} w_{ij} \left\| \hat{z}_i - \hat{z}_j \right\|^2 + (1-\alpha) \left\| \mathbf{z} - \mathbf{y} \right\|^2, \tag{9.17}$$

where $\hat{\mathbf{z}} := D^{-1/2}\mathbf{z}$. The first pairwise *smoothness term* encourages $\mathbf{z}$ to vary little across edges of the graph with large weight whereas the second unary *fitness term* to stay close to observation $\mathbf{y}$. Here, α controls the trade-off: $\mathbf{z}^\star$ equals $\mathbf{y}$ for $\alpha = 0$, while for $\alpha \to 1$, it tends to be constant over connected components of G.

9.4 USAGE

IN THE LITERATURE Zhou *et al.* [517] use the RWR process (9.14) to *rank data on manifolds*. They define the vector $\mathbf{y}$ with elements as 1

at queries and 0 elsewhere. After smoothing, the data are ranked by descending order of the elements of the solution $\mathbf{z}$. Similarly, Zhou *et al*. [516] use the matrix version of the RWR process (9.15) for *semi-supervised classification*. They define the *label matrix* Y with rows as one-hot labels at labeled examples and 0 elsewhere. Labels are inferred according to row-wise maximum of the solution Z. We refer to this as *Label Propagation* (LP). Kim *et al*. [224] use the same idea for *interactive segmentation* with pixel-wise label matrix Y defined according to user-specified seeds (strokes) per object.

Label Propagation *(LP)*.

IN THIS MANUSCRIPT We use the definitions of this Chapter as follows. In Chapter 10 we use the linear system solution (9.7) and the frequency-domain solution (9.13) to *rank images on manifolds*. In Chapter 12 we use the linear system (9.7) to compute the *Katz centrality* [219] for *unsupervised object discovery*. In Chapter 14, we use the linear system (9.7) to compute the *manifold similarity* of images for *unsupervised metric learning*. In Chapter 15, we use the matrix version of the linear system (9.9) to build an *inductive version* of LP [516] for *semi-supervised learning* of a CNN classifier. Finally, in Chapter 17, we use the matrix version of the linear system (9.9) to smooth a perturbation image *guided* by an input image in generating an *adversarial example* for a classifier.

In all cases, we solve the linear systems by the *Conjugate Gradient* (CG) [317] method, which applies because $\mathcal{L}_\alpha$ is positive-definite. In all cases except Chapter 17, the observation vector $\mathbf{y}$ or matrix Y is sparse with nonzero elements at queries or labels. In Chapter 17, it is an arbitrary real-valued signal to be smoothed.

HARMONIC SOLUTION For completeness, we mention an alternative version of LP by Zhu and Ghahramani [521] for *semi-supervised classification*, which iteratively propagates labels and clamps the labeled data. As discussed by Zhu *et al*. [522], this is the harmonic solution of a Laplace equation with Dirichlet boundary conditions on the labeled examples. Grady [139] uses the same idea [522] for *interactive segmentation*, with labels defined like Kim *et al*. [224]. Pérez *et al*. [335] use the same formulation for *image interpolation* within some domain with real-values specified on its boundary; they generalize to guided interpolation, giving rise to a Poisson equation.

The version of Zhu and Ghahramani [521] retains the labels on the labeled data, assuming they are noise free. By contrast, in the version of Zhou *et al*. [516], new labels are allowed on labeled examples. This can be useful *e.g*. when classes overlap or labels are noisy. This version is more useful in ranking, where the observation vector originates in distances or similarities in the feature space and is not to be trusted like human supervision. It makes even more sense when the observation vector is an arbitrary signal.

9.5 DETAILED USAGE

In the following, we provide more details on how the definitions of this Chapter apply in each case. This material is given for reference only and it is suggested to skip it at least at first reading.

RANKING DATA ON MANIFOLDS Zhou *et al.* [517] use the RWR iterating process (9.14) to *rank data on manifolds*. They are given a set of points $V := \{\mathbf{v}_1, \dots, \mathbf{v}_n\} \subset \mathbb{R}^d$, a subset of which are queries. They define the adjacency matrix $W = (w_{ij})$ with $w_{ii} = 0$ and w_{ij} according to a decreasing function of the distance $\|\mathbf{v}_i - \mathbf{v}_j\|$ for $i \neq j$. They also define the vector $\mathbf{y} := (y_1, \dots, y_n)$ where $y_i = 1$ if $\mathbf{v}_i$ is a query and $y_i = 0$ otherwise. They perform RWR until convergence and they use the resulting solution $\mathbf{z}^\star$ as a *ranking score*: They rank each point $\mathbf{v}_i$ according to $z_i^\star$, greatest first.

Regional diffusion *[194].*

REGIONAL DIFFUSION In Section 10.3, we use the linear system solution (9.7) to *rank images on manifolds* as above according to a *regional CNN representation*. We are given a collection of *feature vectors* $V := \{\mathbf{v}_1, \dots, \mathbf{v}_n\} \subset \mathbb{R}^d$, with either one *global* or multiple *regional* vectors per image. We define the adjacency matrix W of a nearest neighbor graph as described in Section 10.2. The queries are not assumed to belong to the collection; rather, we compute $\mathbf{y}$ as the sum over queries of the similarities to the k-nearest neighbors in V per query (10.6), where the similarities are defined by (10.1). We solve linear system (10.8) by CG [317] to obtain $\mathbf{z}^\star$. Each image is associated with several elements of $\mathbf{z}^\star$, one per region; we pool these scores by taking a linear combination per image.

FSR [189].

FAST SPECTRAL RANKING In Section 10.4, having the same problem as in Section 10.3, we use instead an approximation of the *frequency-domain* solution (9.13) to obtain $\mathbf{z}^\star$. In particular, we obtain an approximate low-rank Fourier basis by a randomized algorithm [152, 374], approximating $\mathcal{W} \approx U_r \Lambda_r U_r^\top$ with rank $r \ll n$, then filter $\mathbf{y}$ in the frequency domain by (10.11). The pooling operation from regions to images is integrated with U_r offline.

GOD [406, 407].

OBJECT DISCOVERY In Section 12.4, we use the linear system solution (9.7) to compute the *Katz centrality* [219] of image regions, according again to a *CNN representation*, and we apply to *unsupervised object discovery*. In particular, having the same graph as in Section 10.2, we define the all-ones vector $\mathbf{y} := \mathbf{1}$ and we solve the linear system (12.8) by CG to obtain the vector $\mathbf{z}^\star$, where each element $z_i^\star$ expresses the centrality of vector $\mathbf{v}_i$. In turn, $\mathbf{z}^\star$ is used to compute a *saliency map* per image, according to (12.9). Finally, rectangular salient regions are detected as described by Section 12.3.

MoM [192].

METRIC LEARNING In Section 14.2, having the same graph as in Section 10.2, we solve the linear system (9.7) to compute the manifold

similarity of an image to all others in the collection. By forming the *manifold nearest neighbors* for a set of *anchor* images and comparing to *Euclidean nearest neighbors*, we generate positive and negative pairs of images to perform *unsupervised metric learning*. In particular, for each anchor $\mathbf{v}_i$ in V, we set the vector $\mathbf{y} := \mathbf{e}_i$, where $\mathbf{e}_i$ is the standard n-dimensional basis vector and obtain the solution $\mathbf{z}_i^\star$ of linear system (14.1) by CG. Then, for each vector $\mathbf{v}_j$ in V, the manifold similarity of $\mathbf{v}_i, \mathbf{v}_j$ is read off as the j-th element of $\mathbf{z}_i^\star$ (14.2). The *manifold nearest neighbors* of $\mathbf{v}_i$ in V are the elements of V corresponding to the k greatest elements of $\mathbf{z}_i^\star$.

LABEL PROPAGATION (TRANSDUCTIVE) Zhou *et al.* [516] use the matrix version of the RWR iterating process (9.15) and apply it to *transductive semi-supervised classification*. They are given a set of points $V := \{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \subset \mathbb{R}^d$, a subset of which are labeled over c classes. Like Zhou *et al.* [517], they define the adjacency matrix $W = (w_{ij})$ with $w_{ii} = 0$ and w_{ij} according to a decreasing function of the distance $\|\mathbf{v}_i - \mathbf{v}_j\|$ for $i \neq j$. They also define the $n \times c$ *label matrix* $Y = (y_{ij})$ with $y_{ij} = 1$ if $\mathbf{v}_i$ is labeled in class j and $y_{ij} = 0$ otherwise. They perform RWR until convergence, obtaining as solution the $n \times c$ matrix $Z^\star$. They use this matrix to infer a label $\hat{y}_i$ for each point $\mathbf{v}_i$ as the class corresponding to greatest element of the i-th row of $Z^\star$, *i.e.* $\hat{y}_i := \arg\max_j z_{ij}^\star$ (15.5).

LABEL PROPAGATION (INDUCTIVE) In Section 15.4, we use the matrix version of the linear system solution (9.9) to build an *inductive version* of LP [516] above for *semi-supervised learning* of a *parametric CNN classifier*. We are given the same collection of feature vectors V and the same graph as in Section 10.2, only now the CNN representation V is updated as the classifier is trained. We are also given the same label matrix Y as above. We solve the linear system (15.6) using the CG method. Using the solution $Z^\star$, we infer a *pseudo-label* $\hat{y}_i := \arg\max_j z_{ij}^\star$ (15.5) for each unlabeled example $\mathbf{v}_i$ as in LP. We use those pseudo-labels along with the true labels of the labeled examples and certainty weights (15.7) to train the classifier for one epoch, optimizing the weighted cost function (15.8). By doing so, the feature vectors V and the graph are updated, hence we iterate.

DLP [193].

ADVERSARIAL EXAMPLES In Section 17.5, we use the matrix version of the linear system solution (9.9) to smooth a perturbation image $\mathbf{y} \in \mathbb{R}^{n \times d}$, *guided* by an input image $\mathbf{x} \in \mathbb{R}^{n \times d}$, while minimizing a cost function with respect to $\mathbf{y}$ in an attempt to generate an *adversarial example* for a given classifier f, starting at input image $\mathbf{x}$ with class label t. As discussed in Section 17.2, the graph is defined over the pixels of the input image $\mathbf{x}$ according to (17.11). The system is solved by CG and the smooth output, denoted by $s_\alpha(\mathbf{y})$, is row-normalized according to (17.12). The cost function (17.15) consists of a distortion term $\|s_\alpha(\mathbf{y})\|^2$ and a classification loss term (17.9) such that, when the smooth perturbation $s_\alpha(\mathbf{y})$ is added to the input image $\mathbf{x}$, the classi-

Smooth adversarial examples *[505].*

The perturbation is a matrix but denoted by $\mathbf{y}$ *rather than* Y *here.*

fier f makes an incorrect prediction (other than t). Similarly to (9.17), the smooth perturbation $s_\alpha(\mathbf{y})$ is the minimizer of the quadratic cost function $Q_\alpha(\mathbf{z}, \mathbf{y})$ (17.13) over $\mathbf{z} \in \mathbb{R}^{n \times d}$, which encourages $\mathbf{z}$ to be close to $\mathbf{y}$ and to be smooth wherever $\mathbf{x}$ is.

10 SEARCHING ON MANIFOLDS

Using powerful CNN representations, we explore the manifold structure of the feature space, bringing dramatic gains in standard image retrieval benchmarks. We are the first to study a diffusion *mechanism on CNN representations, which can be seen as a recursive form of query expansion. We introduce a number of solutions that differ in the amount of offline pre-processing and space used to accelerate online search. We also introduce a novel view of search as* smoothing *of a sparse signal on a graph.*

10.1 INTRODUCTION

When searching in an image collection, the query is often connected to relevant images by a sequence of images, where pairs of consecutive images are similar. These images form a manifold in the feature space. *Query Expansion* (QE) [75, 438] has been an early method to exploit this idea. *Average Query Expansion* (AQE) is common with CNN representations [136, 214, 439], but only explores a small neighborhood of the query. *Diffusion* [101, 329, 517] is using a *k-Nearest Neighbor* (k-NN) graph of the dataset constructed offline to efficiently search on the manifold online in a principled way.

A workaround is to operate on small sub-graphs *using text [208].*

The use of a k-NN graph has been prohibitive on conventional representations of thousands of local features per image. In this work, we investigate diffusion on CNN representations of one or few features per image for the first time. Using a number of regional features per image effectively recovers small objects, which are a common failure case of CNN-based retrieval. This *regional diffusion* [194] incurs no extra cost compared to diffusion on global features. We introduce a novel mechanism to handle *unseen queries* and we use a closed-form solution that has been avoided so far [101], solving a linear system online by the *Conjugate Gradient* (CG) [317] method.

We thus reduce manifold search to a two-stage Euclidean/dot product search.

We then introduce *Fast Spectral Ranking* (FSR) [189], shifting more computation offline: We exploit a *low-rank* spectral decomposition of the graph adjacency matrix to express the linear system solution as a sequence of matrix multiplications. Equivalently, we treat the query as a signal to be smoothed over the graph in the *frequency domain*, connecting query expansion to *graph signal processing* [383]. We provide a truly scalable solution to computing an approximate *Fourier basis* of the graph offline, accompanied by performance bounds.

10.2 NEAREST NEIGHBOR GRAPH

These definitions are used in Sections 12.4, 14.2 and 15.4.

We are given a set of n *feature vectors* $V := \{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \subset \mathbb{R}^d$, with each $\mathbf{v}_i$ associated to vertex of graph G. An image collection may be

represented by either one *global* or multiple *regional* feature vectors per image. By using a single region per image, global feature vectors are a special case of regional ones.

Graph G is a *k-Nearest Neighbor* (k-NN) similarity graph, where edges are pairs of vectors that are *reciprocal* (*mutual*) nearest neighbors [232]. In particular, we assume a symmetric non-negative *similarity measure* $s : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. Given $\mathbf{v}' \in \mathbb{R}^d$, let

Given $\mathbf{v}, \mathbf{v}' \in \mathbb{R}^d$, we use the monomial kernel $s(\mathbf{v}, \mathbf{v}') = [\mathbf{v}^\top \mathbf{v}']_+^3$ *[436].*

$$s_k(\mathbf{v}|\mathbf{v}') := \begin{cases} s(\mathbf{v}, \mathbf{v}'), & \text{if } \mathbf{v} \in \mathrm{NN}_k(\mathbf{v}') \\ 0, & \text{otherwise} \end{cases} \tag{10.1}$$

be the similarity of $\mathbf{v} \in V$ to $\mathbf{v}'$ if $\mathbf{v}$ is in the k-nearest neighbors $\mathrm{NN}_k(\mathbf{v}')$ of $\mathbf{v}'$ in V, and zero otherwise. Then, if $\mathbf{v}' \in V$,

$\mathrm{NN}_k(\mathbf{v}')$ excludes $\mathbf{v}'$ itself.

$$s_k(\mathbf{v}, \mathbf{v}') := \min\{s_k(\mathbf{v}|\mathbf{v}'), s_k(\mathbf{v}'|\mathbf{v})\} \tag{10.2}$$

equals $s(\mathbf{v}, \mathbf{v}')$ if $\mathbf{v}, \mathbf{v}'$ are the k-nearest neighbors of each other in V, and zero otherwise. The $n \times n$ *adjacency matrix* $W = (w_{ij})$ of G is then defined by $w_{ij} := s_k(\mathbf{v}_i, \mathbf{v}_j)$ for $i, j \in [n]$.

We write $[n] := \{1, \ldots, n\}$ for $n \in \mathbb{N}$.

10.3 REGIONAL DIFFUSION

BACKGROUND: DIFFUSION In the work of Zhou *et al.* [517], an *observation* vector $\mathbf{y} = (y_i) \in \mathbb{R}^n$ specifies a set of *queries* in V, with $y_i = 1$ if $\mathbf{v}_i$ is a query and $y_i = 0$ otherwise. With this definition of $\mathbf{y}$, the iterative process (9.14) is used

$$\mathbf{z}^{(t)} := \alpha \mathcal{W} \mathbf{z}^{(t-1)} + (1 - \alpha)\mathbf{y}, \tag{10.3}$$

where $\mathcal{W}$ is normalized according to (9.3) and $\alpha \in [0, 1)$ is a parameter. We refer to this process as *Random Walk with Restart* (RWR) [330]. Zhou *et al.* [516, 517] show that regardless of the choice of $\mathbf{z}^{(0)}$, vector $\mathbf{z}^{(t)}$ converges to $\mathbf{z}^\star = (z_i^\star)$ defined by

$$\mathbf{z}^\star := \mathcal{L}_\alpha^{-1} \mathbf{y} \tag{10.4}$$

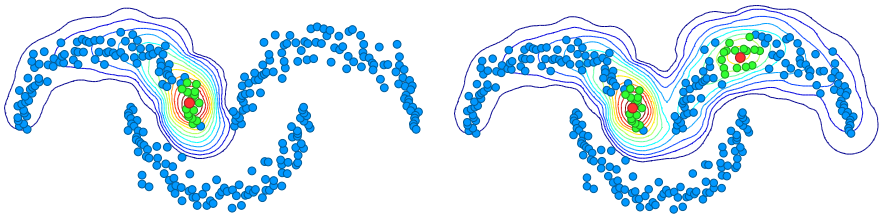
In this work, we use the closed form *solution (10.4) rather than its derivation from (10.3).*

as $t \to \infty$, where $\mathcal{L}_\alpha$ is defined by (9.6). This results in a *ranking score* $z_i^\star$ for each vector $\mathbf{v}_i \in V$, expressing a similarity of $\mathbf{v}_i$ to the set of queries. The benefit is that this similarity captures the intrinsic manifold structure represented by the graph, while multiple queries are combined without additional cost.

HANDLING NEW QUERIES Prior work on diffusion usually assumes a query vector $\mathbf{q} \in \mathbb{R}^d$ to be contained in the dataset V [101, 516]. This does not hold in a retrieval scenario. A query can be included in the graph at query time [509], but this incurs additional cost, especially to maintain reciprocity (10.2) in the presence of $\mathbf{q}$.

Here we introduce an alternative approach which defines observation vector $\mathbf{y}$ in a new way rather than updating the graph. In particular, instead of searching for $\mathbf{q}$, we are searching for its k-nearest neighbors $\mathrm{NN}_k(\mathbf{q})$ in V, weighted by their similarity to $\mathbf{q}$:

$$y_i := s_k(\mathbf{v}_i|\mathbf{q}) \tag{10.5}$$



(a) single query (b) multiple queries

Figure 10.1: Diffusion on a synthetic dataset in $\mathbb{R}^2$. Dataset points, query points and their k-NN are shown in blue, red, and green respectively. The adjacency matrix is defined according to (10.2) with $k = 15$. Diffusion follows by (10.4) with $\alpha = 0.99$ and $\mathbf{y}$ by (10.6). Contour lines correspond to ranking scores after diffusion.

for $i \in [n]$. Assuming a single feature vector per image and defining the observation vector $\mathbf{y}$ by (10.5) is referred to as *global diffusion*.

A query is also represented by a single vector $\mathbf{q} \in \mathbb{R}^d$.

Figure 10.1(a) shows a toy 2-dimensional example, where the k-nearest neighbors of $\mathbf{q}$ taken into account in (10.5) are depicted.

REGIONAL DIFFUSION While global diffusion fits perfectly with the early CNN-based global features [23, 214, 351], it may still fail under severe occlusion or when the object of interest is small. Local CNN features from multiple regions have been investigated for this purpose, either aggregated [132, 439] or represented as a set [363].

Following the latter choice, an image is represented by a set of m feature vectors in $\mathbb{R}^d$, one for each region. Dataset V is the union of such sets over all images; n still denotes its size. The query image is also represented by a set Q of m vectors, and (10.5) becomes

Computing $\mathbf{y}$ involves searching for each query $\mathbf{q}$ individually in V.

$$y_i := \sum_{\mathbf{q} \in Q} s_k(\mathbf{v}_i | \mathbf{q}) \tag{10.6}$$

for $i \in [n]$. Each $\mathbf{v}_i \in V$ is assigned a scalar that is the sum of similarities over all queries $\mathbf{q}$ of which $\mathbf{v}_i$ is a k-nearest neighbor in V, and zero if it is not a k-nearest neighbor of any query. Given $\mathbf{y}$, diffusion (10.4) is now performed jointly for all queries in $\mathbf{q} \in Q$, at no additional cost compared to (10.5). After diffusion, each image is associated with several elements of the ranking score vector $\mathbf{z}^\star$, one for each region. A linear combination of these scores is taken, by *average pooling* or *Generalized Max Pooling* (GMP) [191, 310]. The above process is called *regional diffusion*.

Details are given in [194].

Figure 10.1(b) illustrates diffusion on multiple query points. It is evident that multiple manifolds are captured in this case.

EFFICIENT SOLUTION Iteration (10.3) is slow at large scale. To use the closed-form solution (10.4), one may compute the inverse $\mathcal{L}_\alpha^{-1}$ offline, but this matrix is not sparse like $\mathcal{L}_\alpha$. Connecting (10.3) to linear system solvers, we propose a more efficient solution here.

Diffusion is an iterative solver. Eq. (10.3) is in fact an iteration of the *Jacobi* solver [149]. Given a linear system $A\mathbf{x} = \mathbf{b}$, Jacobi decomposes A as $\Delta + R$, where $\Delta = \mathrm{diag}(A)$ and iterates according to

$$\mathbf{x}^{(t)} := \Delta^{-1}(\mathbf{b} - R\mathbf{x}^{(t-1)}). \tag{10.7}$$

(10.7) then becomes $\mathbf{z}^{(t)} := \alpha\mathcal{W}\mathbf{z}^{(t-1)} + (1-\alpha)\mathbf{y}$.

Substituting $\mathbf{x} \leftarrow \mathbf{z}$, $\mathbf{b} \leftarrow (1-\alpha)\mathbf{y}$, and $A \leftarrow (1-\alpha)\mathcal{L}_\alpha = I - \alpha\mathcal{W}$ (9.6), it follows that $\Delta = I_n$ and $R = -\alpha\mathcal{W}$, re-deriving (10.3).

CG [317] is the method of choice for linear systems like (9.7)

$$\mathcal{L}_\alpha \mathbf{z} = \mathbf{y}, \tag{10.8}$$

where $\mathcal{L}_\alpha$ is positive-definite, in particular for graph-related problems [463]. It has been used for random walk problems [139], but explicitly avoided in diffusion-based retrieval [101].

In fact, CG minimizes the quadratic function $E_\alpha(\mathbf{z})$ (9.16) over a family of Krylov subspaces [443].

Here we argue, as in [246], that it is the solution of (10.4) that we seek, rather than the path followed by iteration (10.3). We use CG to approximate this solution. Contrary to other iterative methods including (10.3), CG terminates in n steps. Remarkably, it provides good approximations in very few steps.

SCALING UP Here we address issues concerning space and online processing at large scale.

This also reduces the redundancy of region features.

Compact representation. To keep the number of region features per image as low as possible, we learn a GMM on the original features of each image and represent the image by the ℓ_2-normalized means.

Truncating the adjacency matrix. We first search through the dataset using global descriptors. We then truncate W, keeping only the rows and columns corresponding to the regions of the top ranked images, and re-normalize it by (9.3). We similarly truncate vector $\mathbf{y}$.

Thus, diffusion only re-ranks the top ranked images.

10.4 FAST SPECTRAL RANKING

As we have seen in Section 9.2, we can write the solution of (10.8) as

$$\mathbf{z}^* := h_\alpha(\mathcal{W})\mathbf{y}, \tag{10.9}$$

where the *transfer function* $h_\alpha(\mathcal{W}) = \mathcal{L}_\alpha^{-1}$ is defined by (9.8). The problem is then to compute $\mathbf{z}^\star$ efficiently, in the sense that $h_\alpha(\mathcal{W})$ is never explicitly computed or stored: $\mathcal{W}$ is given in advance and we can preprocess it *offline*, while $\mathbf{y}$ is given *online*. In particular, we are looking for a more efficient solution than solving linear system (10.8).

We are based on the spectral decomposition (9.11), whereby

$$\mathbf{z}^* = U h_\alpha(\Lambda) U^\top \mathbf{y}. \tag{10.10}$$

Graph spectral filtering is well-known [383, 401], but search as smoothing a sparse signal is a new view.

Here U, Λ represent the eigenvectors and eigenvalues of $\mathcal{W}$, respectively. A low-rank approximation involving the leading eigenvector and eigenvalues can indeed be computed offline. Under the *spectral filtering* interpretation of Section 9.3, (10.10) represents a *smoothing* operation of the sparse signal $\mathbf{y}$ on graph G in the frequency domain, and U is the Fourier basis of the graph. Figure 10.2 depicts 1d and graph miniatures of this interpretation.

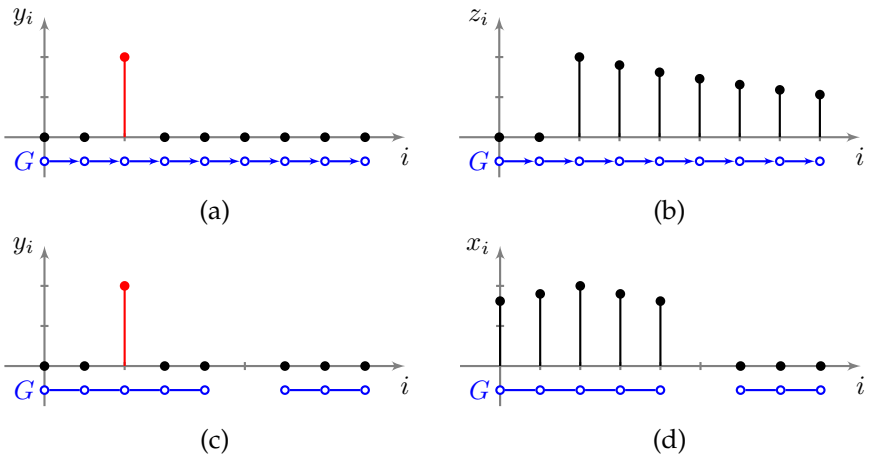


Figure 10.2: *Retrieval as smoothing*. (a) Input signal $\mathbf{y}$. (b) Output of an *Exponential Moving Average* (EMA) filter given by recurrence $z_i := \alpha z_{i-1} + (1-\alpha)y_i$ (9.1). Compare to (10.3). This assumes a directed graph G, in blue. (c), (d) We use an undirected graph G instead. Information "flows" in both directions, controlled by edge weights. The sample in red is the query, and output signal $\mathbf{z}$ is its similarity to all samples.

OFFLINE: FOURIER BASIS Our solution is based on a low-rank approximation of $\mathcal{W}$ computed offline. The approximation is based on a randomized algorithm [374]. In the following, rank $r \ll n$ and number of iterations q are given parameters.

This is similar to Nyström sampling *[105] but with performance guarantees [152, 486].*

1. (*Range basis*) Using *simultaneous iteration* [443, §28], find an $n \times r$ matrix Q with orthonormal columns that represents an approximate basis for the range of $\mathcal{W}$, *i.e.* $QQ^\top\mathcal{W} \approx \mathcal{W}$. In particular [152, §4.5]: Randomly draw an $n \times r$ standard Gaussian matrix $B^{(0)}$ and repeat for $t = 0, \dots, q-1$:
   a) Compute QR factorization $Q^{(t)}R^{(t)} = B^{(t)}$.
   b) Define the $n \times r$ matrix $B^{(t+1)} := \mathcal{W}Q^{(t)}$.

   Finally, set $Q := Q^{(q-1)}$.

   Stage 1 is embarrassingly parallelizable.

2. (*Fourier basis*) Find a rank-r eigenvalue decomposition $U_r\Lambda_r U_r^\top \approx \mathcal{W}$, where $n \times r$ matrix U_r has orthonormal columns and $r \times r$ matrix Λ_r is diagonal. In particular [152, §5.3]:
   a) Form the $r \times r$ matrix $C := Q^\top\mathcal{W}Q$.
   b) Compute its eigendecomposition $V_r\Lambda_r V_r^\top = C$.
   c) Define the matrix $U_r := QV_r$.

   Even if n is very large, r is small enough such that this decomposition is tractable.

An average-case bound on $\left\|QQ^\top\mathcal{W} - \mathcal{W}\right\|$ for the approximation of stage 1 decays to $|\lambda_{r+1}|$ exponentially fast in the number of iterations q [152, §9.3,10.4]. Since $\mathcal{W}$ is symmetric, stage 2 indeed yields $\mathcal{W} \approx QQ^\top\mathcal{W}QQ^\top = QCQ^\top = QV_r\Lambda_r V_r^\top Q^\top = U_r\Lambda_r U_r^\top$ [152, §9.4].

ONLINE: SPECTRAL FILTERING Given $\mathbf{y}$, compute

$$\mathbf{z} := U_r h_\alpha(\Lambda_r) U_r^\top \mathbf{y}. \tag{10.11}$$

A similar situation appears in [441, §3.3].

We are actually approximating $h_\alpha(\mathcal{W})$ by $U_r h_\alpha(\Lambda_r) U_r^\top$. Therefore, it is $|h_\alpha(\lambda_{r+1})|$ that governs the error rather than $|\lambda_{r+1}|$. Since we are using the leading eigenvectors and eigenvalues of $\mathcal{W}$, this approximation makes sense because h_α is *nondecreasing*, as shown in Figure 9.1.

Vector $\mathbf{z} \in \mathbb{R}^n$ contains the *ranking score* z_i of each region feature $\mathbf{v}_i$. To obtain a score per image, we perform a linear pooling operation [194] represented as $\overline{\mathbf{z}} := \Sigma \mathbf{z}$, where Σ is a sparse $N \times n$ *pooling matrix* and N is the number of images. We then directly compute $\overline{\mathbf{z}} := \overline{U}_r h_\alpha(\Lambda_r) U_r^\top \mathbf{y}$ online, where the $N \times r$ matrix $\overline{U}_r := \Sigma U_r$ is computed offline.

The number of nonzero elements of $\mathbf{y}$ and rows of $\overline{U}_r$, whence the cost, are the same for global or regional search.

Computing $\mathbf{y}$ involves Euclidean search in $\mathbb{R}^d$. Then, $U_r^\top$ projects $\mathbf{y}$ onto $\mathbb{R}^r$. With Λ_r being diagonal, $h_\alpha(\Lambda_r)$ is computed element-wise as discussed in Section 9.3. Finally, multiplying by $\overline{U}_r$ and ranking $\mathbf{z}$ amounts to a dot product similarity search in $\mathbb{R}^r$.

> *We thus reduce manifold search to Euclidean followed by dot product search.*

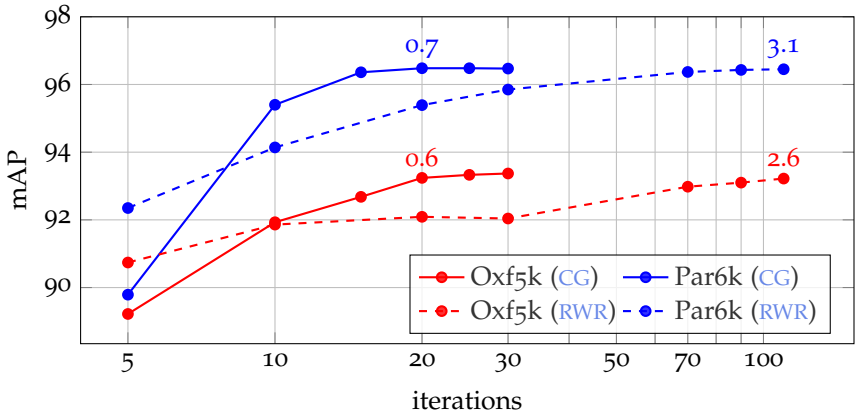


Figure 10.3: mAP of regional diffusion *vs.* number of iterations by solving (10.8) with CG against iterative process RWR (10.3), using VGG features ($d = 512$). Labels indicate diffusion time (s).

10.5 EXPERIMENTS

We apply supervised whitening *[297] to both global and regional features, as in [351].*

SETUP We compare Euclidean to manifold search with global or regional CNN features. We use d-dimensional features by VGG [409] ($d = 512$) and ResNet101 [165] ($d = 2{,}048$), fine-tuned for image retrieval [137, 351]. Global features are R-MAC [439] at 3 scales; regional use the same 21 regions per image without pooling, optionally reduced to 5 by GMM. Regional features are matched pairwise as in *Regional Matching* (R-Match) [363]. The similarity measure is $s(\mathbf{v}, \mathbf{v}') := [\mathbf{v}^\top \mathbf{v}']_+^3$. We use GMP [191, 310] to pool regional diffusion scores per

$[x]_+ := \max(x, 0)$ *is the* positive part *of* $x \in \mathbb{R}$.

image. We set $\alpha = 0.99$, and $k = 50$ (200) for global (regional) diffusion.

We use Oxford5k [341], Paris6k [342] and Instre [469] benchmarks. For large-scale experiments, we add 100k distractor images [341] to Oxford5k and Paris6k, referred to as Oxford105k and Paris106k respectively. We measure performance by mAP. Times exclude construction of the observation vector $\mathbf{y}$.

We introduce a new evaluation protocol for Instre.

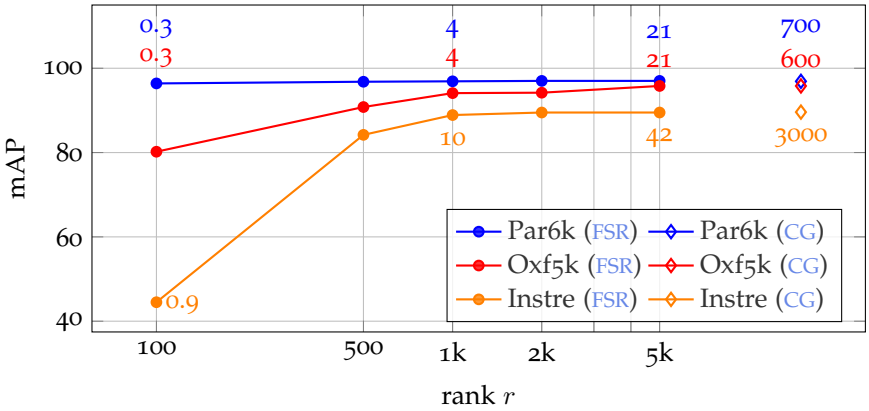


Figure 10.4: mAP of regional diffusion with FSR *vs.* rank r against CG, using VGG features ($d = 512$). Labels indicate diffusion time (ms).

RESULTS Figure 10.3 compares our CG solution to iterative diffusion (10.3). CG converges in as few as 20 iterations, while (10.3) reaches the same performance only after 110 iterations.

Figure 10.4 shows that in all datasets, the optimal FSR performance is already reached at rank $r = 1k$. On Paris6k in particular, this happens as soon as $r = 100$. FSR reaches the same mAP as CG, 150 times faster on Oxford5k and Paris6k and 300 times faster on Instre.

Table 10.1 compares our diffusion Euclidean search and AQE [75]. Regional diffusion significantly outperforms all other methods in all datasets. Global diffusion performs well on Paris because query objects mostly cover the image. This does not hold on Instre, which contains a lot of small objects. FSR performs similarly to CG but at dramatic speed-up, *almost as fast as Euclidean search*: Dataset truncation is no longer needed and this improves mAP.

AQE is common with global representation [136, 214, 439]; we extend it to regional.

10.6 DISCUSSION

The power of CNN features allows representing an image by just one or a few vectors. This, combined with our efficient solutions, allows for the first time to perform large scale diffusion at reasonable query times. Its effect is dramatic, especially when images contain small objects. The closed form solution, approximated by CG, is significantly faster than the iterative process RWR (10.3); but FSR offers a dramatic speed-up, at the expense of space for the eigenvectors U.

METHOD	m	INSTRE	OXF5K	OXF105K	PAR6K	PAR106K
GLOBAL FEATURES: R-MAC [137]						
Euclidean	1	62.6	83.9	80.8	93.8	89.9
AQE	1	70.5	89.6	88.3	95.3	92.7
CG	1	80.5	87.1	87.4	96.5	95.4
FSR	1	80.5	87.5	87.9	96.4	95.3
REGIONAL FEATURES: R-Match [363]						
Euclidean	21	71.0	88.1	85.7	94.9	91.3
AQE	21	77.1	91.0	89.6	95.5	92.5
CG	5	88.4	95.0	90.0	96.4	**95.8**
FSR	5	**88.5**	**95.1**	**93.0**	**96.5**	95.2

Table 10.1: mAP of our CG and FSR diffusion against Euclidean search—R-MAC [137] (R-Match [363]) for global (regional) representation—and AQE, using ResNet101 [137] features ($d = 2{,}048$). Regions per image reduced from $m = 21$ to 5 by GMM. Truncation at top 10k images at large scale. Rank $r = 5$k for FSR.

Revisited Oxford and Paris *(RevOP) [350].*

The excellent performance of our methods motivates us to revisit Oxford [341] and Paris [342], introducing the *Revisited Oxford and Paris* (RevOP) benchmark [350]. We provide new annotation for both datasets, correcting previous errors. We introduce 15 new, more difficult queries per dataset and update the evaluation protocol by introducing three new protocols of varying difficulty. We also create a new set of one million challenging distractors.

Hybrid diffusion [190].

At this scale, it turns out that CG either is too slow or incurs loss by truncation, while FSR either takes too much space or fails at low-rank approximations. We therefore introduce *hybrid diffusion* [190], allowing full control of the space-time trade-off between these two extremes. This approach performs on par with the state of the art, with lower space compared to FSR and faster than CG.

Diffusion code can be found on `github`[1], as well as the revisited Oxford and Paris benchmark[2]. Our version of the Instre dataset is also available online[3].

1 https://github.com/ahmetius/diffusion-retrieval/
2 https://github.com/filipradenovic/revisitop/
3 ftp://ftp.irisa.fr/local/texmex/corpus/instre/instre.tar.gz

11 SPATIAL MATCHING

Observing that convolutional activations of deep networks are sparse and consistent across different views of a scene, we approximate activation tensors by collections of local features, which we robustly match to find the optimal alignment of two views. We thus introduce Deep Spatial Matching *(DSM) [405] for image retrieval. This happens without any network modification, additional layers or training and without using local descriptors or visual vocabularies.*

11.1 INTRODUCTION

A 3d activation tensor can be perceived as a collection of 2d response maps of pattern detectors.

Image retrieval based on hand-crafted local descriptors [316, 413] typically relies on *Bag of Words* (BoW) [413] or aggregated descriptors [204], followed by *spatial verification* [334, 341, 433]. CNN-based retrieval [137, 352] mostly relies on global or regional *spatial pooling* of the 3d activation tensor [23, 214, 352, 439], which yields a compact and invariant representation like BoW, but does not allow spatial verification.

Regional pooling followed by pairwise matching was the first to beat conventional retrieval pipelines [436].

It is known that *dense correspondence* can be recovered by correlating activation tensors [73, 272, 372, 373]. Combined with integral image computation, *regional pooling* allows fast sliding window-style spatial matching [439]. However, the activation tensor is too large to be stored, especially for large-scale applications. *Sparse* local feature representations are possible by imitating conventional detector pipelines [274, 294]. Two dominant paradigms are *detect-then-describe*,

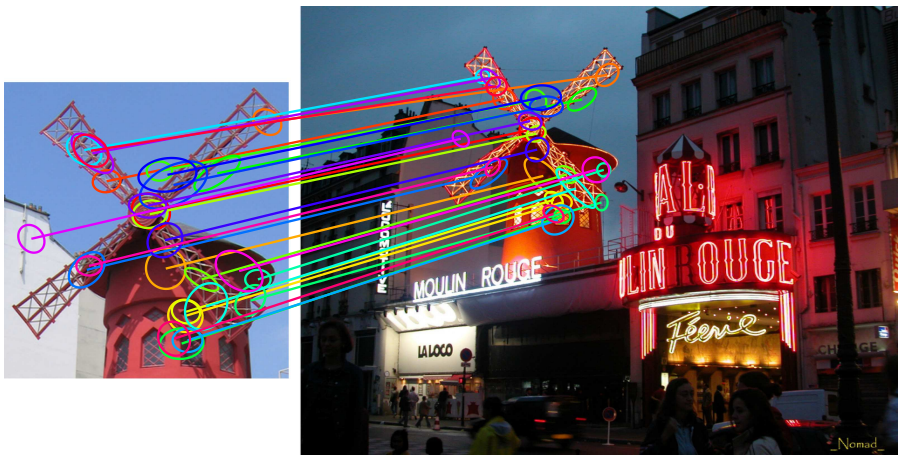


Figure 11.1: *Fast Spatial Matching* (FSM) [341] aligns two views based on a local features. Inlier correspondences shown, colored by "visual word". Are we using (a) Hessian-affine [294] + SIFT [274]? (b) LIFT [498]? (c) DELF [318]? Or (d) local maxima on vanilla activation maps, without descriptors or vocabularies?

as in *Learned Invariant Feature Transform* (LIFT) [498], and *describe-then-detect*, as in *DEep Local Features* (DELF) [318].

We use no local descriptors or vocabularies.

Given enough memory, and using conventional vocabularies, inverted files and spatial verification, DELF is state of the art [350], but not compatible with the global representations used for search. In this work [405], we attempt to reduce this gap by extracting from CNN activations a representation that allows spatial verification, yet it has a trivial relation to global representations: Instead of pooling, we detect *local maxima*. Referring to Figure 11.1, the correct answer is (d): We "read off" information directly from activation maps.

Figure 11.2: Three views (columns) of a scene, overlaid with two feature maps (rows) of the last convolutional layer of VGG16 [409]. All activations are sparse and spatially consistent across views.

11.2 MOTIVATION

Sparsity is studied in Cross-Dimensional Weighting *(CroW) [214], which only provides a coarse saliency map.*

The success of *max-pooling* of convolutional activations (MAC) [439], at least for image retrieval, can be connected to the sparsity of the activations [214]. More interestingly, the locations of maxima can identify correspondences [7, 439]. *Generalized mean pooling* (GeM) [352] is superior, which can be attributed to the fact that it allows for more than one locations contributing to the representation, while still being more selective than average pooling.

We observe all channels, not just those contributing most to image similarity [7, 439].

As illustrated in Figure 11.2, in most cases activation maps in all channels are not just sparse. They also respond at consistent locations with consistent local shape across views, exhibiting translation and scale covariance to some extent. Hence the question:

> *Instead of reducing each channel to a single scalar, why not keep all the peaks of the responses in each channel along with geometric information? Instead of attaching an entire descriptor*

to each such geometric entity, why not just attach the channel it was extracted from, as if it was a visual word?

11.3 DEEP SPATIAL MATCHING

The preceding ideas give rise to the DSM architecture, illustrated in Figure 11.3. We consider a fully convolutional network f that, when followed by spatial pooling *e.g.* MAC [439] or GeM [352], extracts a global feature for retrieval [137, 352]. Two input images x_1, x_2 yield 3d *activation tensors* $A_1 := f(x_1), A_2 := f(x_2)$ at the last convolutional layer of f, where $A_i \in \mathbb{R}^{w_i \times h_i \times c}$, $w_i \times h_i$ is the spatial resolution of A_i for $i = 1, 2$ and c is the number of channels (features).

We see an activation tensor a collection of 2d maps rather then a 2d array or feature vectors, as in image registration [73, 272], optical flow [102] or semantic alignment [223, 372].

From each activation tensor A_1, A_2, feature detector g extracts a sparse collection of *local features* $\mathcal{P}_1 := g(A_1), \mathcal{P}_2 := g(A_2)$ respectively, independently per channel. Then, $\mathcal{P}_1, \mathcal{P}_2$ undergo *spatial matching*, returning a collection of inliers and a geometric transformation, where tentative correspondences are formed again independently per channel. In retrieval, activation tensors are discarded and images are represented by local features alone.

The entire mechanism takes place without adapting the network and without any additional learning.

LOCAL FEATURE DETECTION We use *Maximally Stable Extremal Regions* (MSER) [289] over activation map $A^{(\ell)}$ of tensor A independently for each channel $\ell = 1, \ldots, c$. These are connected regions of arbitrary shape having higher activation than their neighborhood and satisfying a stability criterion.

On images, both maxima (MSER$^+$) and minima (MSER$^-$) are detected. On CNN activations, only maxima are of interest.

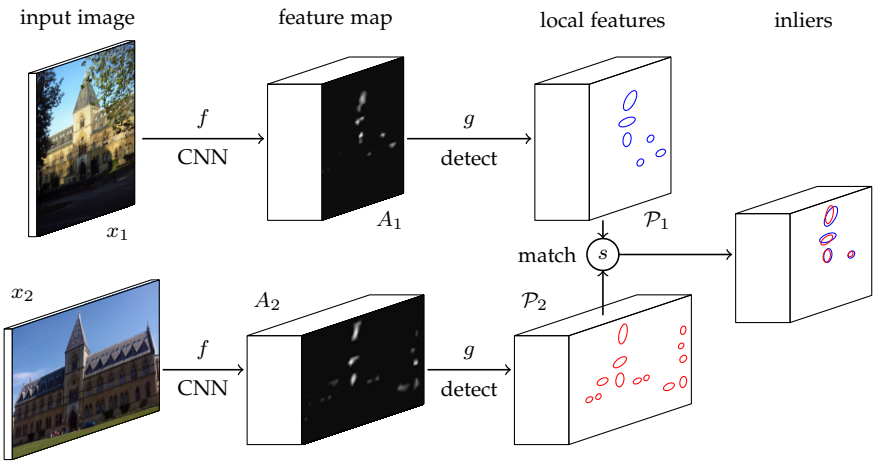


Figure 11.3: DSM architecture. Two input images x_1, x_2 are mapped by network f to activation tensors A_1, A_2 respectively. Sparse *local features* $\mathcal{P}_1, \mathcal{P}_2$ extracted independently per channel by *detector* (g) undergo *spatial matching* (s), resulting in a collection of inliers. In retrieval, only $\mathcal{P}_1, \mathcal{P}_2$ are stored and s applies at re-ranking.

LOCAL FEATURE REPRESENTATION Each MSER detected in channel ℓ gives rise to a local feature p with activation strength $a(p) := \text{pool}_{\mathbf{r}\in R(p)} A^{(\ell)}(\mathbf{r})$, where $R(p)$ is the MSER, pool is a *spatial pooling* operation and $A^{(\ell)}(\mathbf{r})$ the element of $A^{(\ell)}$ at position $\mathbf{r}$. We also fit an ellipse with 2×1 mean (position) vector $\mu(p)$ and 2×2 covariance matrix (local shape) $\Sigma(p)$. We collect local features $\mathcal{P} = (P^{(1)}, \ldots, P^{(c)})$, where $P^{(\ell)}$ contains the local features p found in channel ℓ.

For instance, maximum, mean, or GeM.

CORRESPONDENCES Given the local features $\mathcal{P}_1, \mathcal{P}_2$ of two images x_1, x_2, we form *correspondences*, *i.e.* pairs (p_1, p_2) of local features of the two images. We allow pairs only between local features in the same channel, that is, p_1, p_2 are in $\mathcal{P}_1^{(\ell)}, \mathcal{P}_2^{(\ell)}$ respectively for some channel ℓ. The collection $(\mathcal{P}_1^{(1)} \times \mathcal{P}_2^{(1)}, \ldots, \mathcal{P}_1^{(c)} \times \mathcal{P}_2^{(c)})$ of all such pairs is the set of *tentative correspondences*.

> *We thus treat channels as* visual words*, as if local features were assigned descriptors that were vector-quantized against a vocabulary.*

Feature channels are correlated: One filter may respond to a variety of patterns, while several filters may respond to the same pattern. For this reason we apply *Non-Maximum Suppression* (NMS) over channels on detected regions of database images. We do not apply NMS to the query image to allow matches from any channel.

SPATIAL MATCHING We use FSM [341] to find the geometric transformation between the two images and the subsets of $\mathcal{P}_1, \mathcal{P}_2$ that are consistent with it. FSM is a variant of RANSAC [117] that generates a transformation hypothesis from a single correspondence. A hypothesis is evaluated by the number of *inliers*, that is, correspondences that are consistent with it. All possible hypotheses are enumerated and the transformation with the most inliers is returned.

As in [341], we use LO-RANSAC [74], which iteratively evaluates promising hypotheses by least squares-fitting of a transformation to all inliers.

We adopt the linear 5-DoF transformation allowing for translation, anisotropic scale and vertical shear but no rotation. Images are assumed in "upright" orientation: Given a correspondence of two local features p_1, p_2, one finds from the ellipses defined by $(\mu(p_1), \Sigma(p_1))$, $(\mu(p_2), \Sigma(p_2))$ the transformations T_1, T_2 that map them to the unit circle while maintaining the y-direction, and defines the transformation hypothesis $T = T_2^{-1} T_1$.

RETRIEVAL AND RE-RANKING In image retrieval, n database images $X = \{x_1, \ldots, x_n\}$ are given in advance. For each image x_i with activation tensor A_i, its local features $\mathcal{P}_i := g(A_i)$ are computed along with a global descriptor $\mathbf{v}_i$ spatially pooled directly from A_i, which is then discarded. At query time, given query image x with activation tensor A, local features $\mathcal{P} := g(A)$ and global descriptor $\mathbf{v}$, we first rank $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ by cosine similarity to $\mathbf{v}$, and then the top-ranking images undergo spatial matching against $\mathcal{P}$ and re-ranked according to the number of inliers found.

We apply query-time *diffusion* [194] after spatial re-ranking. The precision of top-ranking images is important for diffusion [350], so spatial re-ranking is expected to help.

11.4 EXPERIMENTS

SETUP We use VGG16 [409] and ResNet101 [165] as trained by [352] with GeM pooling, denoted by VGG (ResNet), or V (R) for short. We also re-train them, denoted by $\star$, with max-pooling (MAC) [439], using the setup of [352]. We upsample ResNet by a factor of 2, denoted by $\uparrow$, without re-training.

We initially rank images by *cosine similarity* on multi-scale global representations extracted at scales related by factors 1, $1/\sqrt{2}$, and $1/2$, and spatially pooled then pooled over scales by MAC [439] or GeM [352]. We use the VLFeat [455] version of MSER, adjusting parameter Δ per network/dataset according to the cumulative distribution of activation values over the dataset. We use a *multi-scale* local representation at the same relative factors as global, keeping the top-ranking 512 (2,048) local features on VGG (ResNet) according to activation, discarding activation maps with more than 20 features. We use an *Intersection over Union* (IoU) threshold 0.2 for NMS.

We apply supervised whitening *[297] to global features, as in [351].*

At re-ranking, we perform spatial matching on all 9 combinations of query and database image scales, keeping the best according to inliers.

METHOD	MEDIUM				HARD			
	$\mathcal{R}$OXF		$\mathcal{R}$OXF+$\mathcal{R}$1M		$\mathcal{R}$OXF		$\mathcal{R}$OXF+$\mathcal{R}$1M	
	mAP	mP@10	mAP	mP@10	mAP	mP@10	mAP	mP@10
DELF-ASMK*+FSM+D [350]	75.0	87.9	68.7	83.6	48.3	64.0	39.4	55.7
V-MAC$\star$	67.7	86.1	56.8	78.6	39.8	51.1	29.4	46.0
V-MAC$\star$+DSM	72.0	90.6	59.2	80.1	43.9	56.0	32.0	47.4
R-MAC$\star\uparrow$	73.9	87.9	61.3	80.6	45.6	62.2	31.9	48.4
R-MAC$\star\uparrow$+DSM	**76.9**	**90.7**	65.7	83.9	**49.4**	**64.7**	35.7	51.3
V-GeM[352]	69.6	84.7	60.4	79.4	41.1	51.1	33.1	49.6
V-GeM[352]+DSM	72.8	89.0	63.2	83.7	45.4	57.1	35.4	53.7
R-GeM[352]$\uparrow$	70.1	84.3	67.5	79.0	41.5	54.4	39.6	53.0
R-GeM[352]$\uparrow$+DSM	75.0	89.6	**70.2**	**84.5**	46.2	60.6	**41.9**	**54.9**

Table 11.1: mAP and mP@10 *state-of-the-art* on benchmark [350]. We use VGG (V) and ResNet (R) with MAC and GeM pooling. $\uparrow$: Upsampling; $\star$: Our re-training. DSM: This work. Top two rows as reported in [350], using DELF [318], ASMK* [437], FSM [341] and diffusion (D) [194] on graph obtained by [137]. Remaining results use diffusion [194] on our global features; GeM as trained by [352].

We perform *spatial verification* of the 100 top-ranked images according to global representations with transfer error threshold of 2 pixels in the activation map, and re-rank according to the number of inliers. We compute the product of the global cosine similarity and number

Diffusion is very powerful but sensitive to the quality of the initial top-ranked results, which we improve.

of inliers on the 10 top-ranking spatially verified images and initiate *global diffusion* [194] on top 5 images according to this product.

We evaluate performance by mAP and mP@10 on the medium and hard setups of the revisited $\mathcal{R}$Oxf and $\mathcal{R}$Par benchmarks [350], as well as the large-scale benchmarks $\mathcal{R}$Oxf+$\mathcal{R}$1M and $\mathcal{R}$Par+$\mathcal{R}$1M, combining a set of 1M distractor images with the two small ones.

RESULTS Table 11.1 only shows results on $\mathcal{R}$Oxf and $\mathcal{R}$Oxf+$\mathcal{R}$1M using diffusion [194]. All baselines are significantly improved by re-ranking, with a gain of up to 5 mAP or 6 mP@10 points.

We also compare to the best performing version of DELF [318] as evaluated by [350], which we outperform in several cases. Apart from spatial verification by [341] on the 100 top-ranking images, this version is using two independent representations. One is our ASMK* [437] on 128-dimensional descriptors of 1000 DELF features per image, used for initial ranking. Another is a global descriptor by ResNet-R-MAC [137], used for diffusion as in this work. By contrast, our global and local representations are obtained from the same activation tensor, and we do not use any local descriptors or vocabularies.

11.5 DISCUSSION

Our representation arises naturally in existing convolutional activations of off-the-shelf or fine-tuned networks, without particular effort to detect local features or extract descriptors on image patches. It does not require network modification or retraining. It is a significant step towards bridging the gap between global descriptors, which are efficient for initial ranking by nearest neighbor search, and local representations, which are compatible with spatial verification.

Of course, the activation channels are not the most appropriate by construction to replace a visual vocabulary. Our representation, while being very compact, is not as powerful as storing *e.g.* hundreds of local descriptors per image. Nonetheless, we demonstrate that it is enough to provide high-quality top-ranking images to initiate diffusion, which then brings excellent results.

DSM code can be found on `github`[1].

1 https://github.com/osimeoni/dsm

12 DISCOVERING OBJECTS

Background clutter is challenging for image retrieval, especially when using global representations. In this work [406, 407], we detect salient regions in an unsupervised manner, capturing discriminative and frequent patterns. Saliency is based on a centrality measure of a nearest neighbor graph of regional CNN representations on an image collection. Pooling features over salient regions improves retrieval performance, mostly for small objects.

12.1 INTRODUCTION

Large objects may be partially occluded, while small may be on background clutter covering most of the image.

Particular object retrieval becomes very challenging when the object of interest is covering a small part of the image. Representations based on *local features* [413] are naturally insensitive to occlusion and background clutter. Locality allows matching small parts of image containing the object of interest, while the incorrect matches are removed by spatial verification [341, 433].

CNN representations have dominated modern image retrieval because they are very discriminative and have excellent performance, while being very compact [23, 25, 439]. Using several regions per image [363, 382], local features [318, 405], or the entire activation tensor [439] can provide robustness to clutter, but is expensive. However, when the object is small, *global* CNN features fail [189, 194].

Ideally, the representation should focus on the relevant part of the image, suppressing clutter and occlusions. Deep networks learn, to some extent, what is discriminative in an image. This can be used for *saliency detection* [207, 214, 302, 318]. Alternatively, methods inspired

Figure 12.1: Saliency map (right) computed for input image (left) based on our salient region detection on Instre dataset. Background clutter and objects not *relevant for Instre* are removed.

by *object detection* are using a number of *region proposals* [136, 306, 382]. In both cases, a global representation is obtained by pooling.

In this work [406, 407], we introduce a pooling mechanism inheriting the properties of both saliency detection and region proposals. It applies to each image *independently*, focusing on *discriminative* parts. We also introduce an *unsupervised Graph-based Object Discovery* (GOD) mechanism, considering the image collection *as a whole* and capturing *frequent objects*. As shown in Figure 12.1, we find what is *relevant for the dataset*, which is not possible by considering images independently. In both cases, we derive a powerful *global* representation.

Our approach outperforms more expensive regional *representations [363], even in the presence of* diffusion *[194].*

12.2 FEATURE SALIENCY

Otherwise, we risk "discovering" uninformative but frequently appearing "stuff"-like patches, e.g. *sky.*

Before looking for frequent patterns in a dataset, we first need to find discriminative regions. Fortunately, this is possible without training or bounding box annotations. Inspired by *Cross-Dimensional Weighting* (CroW) [214], we construct a 2d *Feature Saliency* (FS) map of an image based on its convolutional activation alone.

We write $[n] := \{1, \dots, n\}$ *for* $n \in \mathbb{N}$.

We represent an activation map as a 3d tensor $A \in \mathbb{R}^{w \times h \times c}$, where w, h, c are the height, width and number of feature channels, respectively. Let $\Omega := [w] \times [h]$ be the *spatial domain*, $r := |\Omega| = wh$ the *spatial resolution* and $A^{(\ell)}(\mathbf{r})$ the element of A at position $\mathbf{r} \in \Omega$ and channel $\ell \in [c]$. Following CroW, we use an IDF-like weight

$$b_\ell := -\log\left(\frac{a_\ell + \epsilon}{\sum_{j=1}^{c}(a_k + \epsilon)}\right) \tag{12.1}$$

for $\ell \in [c]$, where $a_\ell := \frac{1}{wh}\sum_{\mathbf{r}\in\Omega} \mathbb{1}[A^{(\ell)}(\mathbf{r}) \neq 0]$ is the average number of nonzero elements of channel $\ell \in [c]$. We then compute a weighted sum over activation channels

$$\widetilde{F}(\mathbf{r}) := \sum_{\ell=1}^{c} b_\ell A^{(\ell)}(\mathbf{r}) \tag{12.2}$$

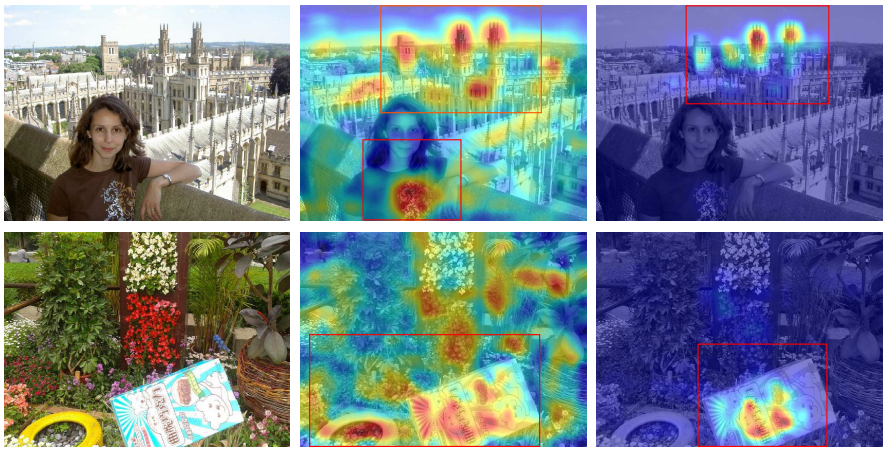
Contrary to CroW, we use the feature channel weights when computing the 2d spatial weights.

for $\mathbf{r} \in \Omega$. Finally, we obtain the 2d FS map $F \in \mathbb{R}^{w \times h}$ by normalizing $\widetilde{F}$ according to [214]. Figure 12.2(b) shows FS examples.

12.3 SALIENT REGIONS

Given a 2d saliency map $S \in \mathbb{R}^{w \times h}$, either FS (Section 12.2) or OS (Section 12.4), the next step is to detect a small set of rectangular regions per image, allowing us to build a region k-NN graph in Section 12.4. Each region is associated with a FS score and a feature vector.

REGION DETECTION We use the *Expanding Gaussian Mixture* (EGM) model [18] discussed in Section 3.3 to detect a number of salient rectangular regions. The original algorithm applies to point sets and isotropic Gaussian components. Here we extend it to *functions*, considering that a saliency map S can be seen as a function $S : \Omega \to \mathbb{R}$.



(a) image (b) FS.EGM (c) OS.EGM

Figure 12.2: Oxford5k (top) and Instre (bottom) images, along with superimposed FS and OS maps and regions detected by EGM, in red.

We fit a number of components, each modeling a region in 2d coordinate space. We also extend it to a *diagonal covariance* model, so that a region is a modeled by an axis-aligned rectangle.

In particular, given 2d saliency map $S \in \mathbb{R}^{w \times h}$, we represent it as a set of Gaussian *sample* functions $q(\mathbf{r}) : \mathbb{R}^2 \to \mathbb{R}$ with

$$q(\mathbf{r})(\mathbf{x}) := S(\mathbf{r})\mathcal{N}(\mathbf{x}|\mathbf{r}, \sigma I) \tag{12.3}$$

σ determines how coarse or fine the representation is.

for $\mathbf{r} \in \Omega$, $\mathbf{x} \in \mathbb{R}^2$, where $\mathcal{N}$ is the normal density and σ is a *scale* parameter. Similarly to (3.7), we represent components as Gaussian functions $p_j : \mathbb{R}^2 \to \mathbb{R}$ with

$$p_j(\mathbf{x}) := \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \tag{12.4}$$

Mean $\boldsymbol{\mu}_j$ represents the center, while the square root of eigenvalues of $\boldsymbol{\Sigma}_j$ represent the height and width of region (component) p_j.

for $j \in [k]$, $\mathbf{x} \in \mathbb{R}^2$, where k is the number of components and $\pi_j \in \mathbb{R}$, $\boldsymbol{\mu}_j \in \mathbb{R}^2$ and $\boldsymbol{\Sigma}_j \in \mathbb{R}^{2\times 2}$ are the mixing coefficient, mean and diagonal covariance matrix respectively of component j. We initialize components as $\{q(\mathbf{r}) : \mathbf{r} \in \Omega\}$, with $k \leftarrow r$. In the *expectation* (E)-step, following (3.11), we find the *generalized responsibility*

$$\hat{\gamma}_j(\mathbf{r}) \leftarrow \frac{\langle q(\mathbf{r}), p_j \rangle}{\sum_{\ell=1}^{k} \langle q(\mathbf{r}), p_\ell \rangle} \tag{12.5}$$

$\langle f, g \rangle$ is the L^2 inner product of $f, g : \mathbb{R}^d \to \mathbb{R}$, given in closed form for Gaussian functions by (3.9).

of component $j \in [k]$ for sample $q(\mathbf{r})$, $\mathbf{r} \in \Omega$. In the *maximization* (M)-step, we update $\pi_j \leftarrow \frac{r_j}{r}$, $\boldsymbol{\mu}_j \leftarrow \frac{1}{r_j} \sum_{\mathbf{r}\in\Omega} \hat{\gamma}_j(\mathbf{r})\mathbf{r}$, and

$$\boldsymbol{\Sigma}_j \leftarrow \frac{1}{r_j} \sum_{\mathbf{r}\in\Omega} \hat{\gamma}_j(\mathbf{r}) \operatorname{diag}\left((\mathbf{r} - \boldsymbol{\mu}_j) \circ (\mathbf{r} - \boldsymbol{\mu}_j)\right), \tag{12.6}$$

where $r_j := \sum_{\mathbf{r}\in\Omega} \hat{\gamma}_j(\mathbf{r})$ is the effective number of points assigned to component j and $\circ$ is the Hadamard product. Finally, in the *purge* (P)-step, similarly to NMS, we process components by Algorithm 3.1 in

k may decrease at each iteration, and overlap is measured similarly to (12.5).

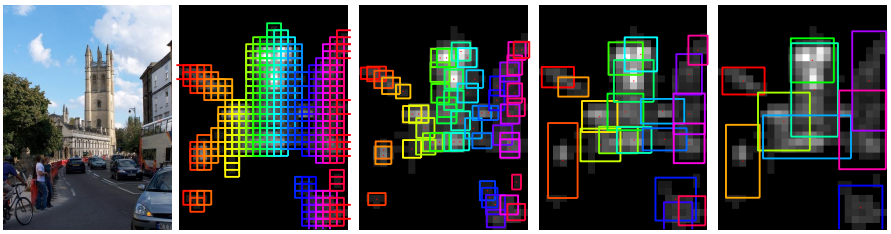


image $i = 0, k = 272$ $i = 1, k = 54$ $i = 4, k = 20$ $i = 14, k = 9$

Figure 12.3: Evolution of regions during EGM iterations on the FS map of the image on the left; i: iteration; m: number of regions.

descending order of mixing coefficient and keep a component unless it overlaps with previously kept ones.

Although less salient, background regions cannot be removed based on FS alone.

Figure 12.3 shows how regions are formed during EGM iterations. We get six clean regions on the ground truth building and five regions on background objects. More examples of region detection on FS and OS maps are shown in Figure 12.2.

REGION REPRESENTATION Given a region R of an image with feature saliency map $F \in \mathbb{R}^{w \times h}$ and activation map $A \in \mathbb{R}^{w \times h \times c}$, we associate it with *feature saliency* $f := \frac{1}{|R|} \sum_{\mathbf{r} \in R} F(\mathbf{r})$ by average pooling of F over R, and with *feature vector* $\mathbf{v} := \phi_A(R) \in \mathbb{R}^d$, where

$A(\mathbf{r}) \in \mathbb{R}^c$ collects elements of all channels of A at position $\mathbf{r}$.

$$\phi_A(R) := \omega \left(\max_{\mathbf{r} \in R} A(\mathbf{r}) \right) \tag{12.7}$$

R-MAC [439] uses uniformly sampled regions at different scales, while ours are based on saliency.

denotes max-pooling [21, 439] of A over R followed by *supervised whitening* $\omega : \mathbb{R}^c \to \mathbb{R}^d$. The latter is performed by [297], as in [351].

12.4 OBJECT SALIENCY

Given an image dataset, we detect a set of regions $\{R_1, \ldots, R_n\}$ from FS maps and we extract a *feature saliency* vector $\mathbf{f} := (f_1, \ldots, f_n) \in \mathbb{R}^n$ from FS maps and a set of feature vectors $V := \{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \subset \mathbb{R}^d$ from activation maps. Based on this information, we construct a k-NN graph of the regions and compute a *centrality* [314] score per region. This allows us to form a dense *Object Saliency* (OS) map, capturing discriminative patterns appearing frequently in the dataset.

Region centrality expresses how often the depicted pattern appears in the dataset.

GRAPH CENTRALITY We form the adjacency matrix W of the k-NN of V according to Section 10.2. Then, following the definitions of Chapter 9, we normalize it into $\mathcal{W}$ by (9.3) and compute the $n \times n$ regularized Laplacian $\mathcal{L}_\alpha$ by (9.6), where $\alpha \in [0, 1)$. The objective is to compute a vector $\mathbf{z} \in \mathbb{R}^n$ where each element z_i represents the significance of vertex $\mathbf{v}_i$ in the graph, for $i \in [n]$. We define this *centrality vector* as the solution $\mathbf{z}^* \in \mathbb{R}^n$ of the linear system

As in Chapter 10, we solve this system by the CG [317] method, but this takes place offline.

$$\mathcal{L}_\alpha \mathbf{z} = \mathbf{1}. \tag{12.8}$$

The solution $\mathbf{z}^*$ is a *graph centrality* measure [314], *Katz centrality* [219] in particular. Its history is summarized in [458].

SALIENCY MAP The problem is then to compute a dense OS map $O \in \mathbb{R}^{h \times w}$ for a new image represented by activation map $A \in \mathbb{R}^{h \times w \times c}$. The value $O(\mathbf{r})$ at $\mathbf{r} \in \Omega$ is found as a linear combination of the centrality values of the k-NN in V of a square patch $P(\mathbf{r})$ centered at $\mathbf{r}$, weighted by similarity and feature saliency

This is a regression problem and we use a non-parametric k-NN solution.

$$O(\mathbf{r}) := F(\mathbf{r})^{\gamma} \sum_{i=1}^{n} s_k(\mathbf{v}_i | \phi_A(P(\mathbf{r}))) f_i^{\gamma'} z_i^*, \tag{12.9}$$

where similarity s_k is defined by (10.1). The sum is weighted by FS, hence OS captures both discriminative and frequent patterns. Exponents γ and γ' control the importance of feature saliency of the current patch and the centrality of neighbors, respectively.

The sum is computed over $NN_k(\phi_A(P(\mathbf{r})))$ only in practice.

Figure 12.2(c) shows OS examples. Looking at the input image and the FS map alone, it is not evident what is of interest and what is clutter. This is only found by discovering other instances of the same object in the dataset, as represented by the graph.

GLOBAL REPRESENTATION Finally, as discussed in Section 12.3, we detect a set of regions $\mathcal{R}$ on a saliency map (FS or OS) in a dataset image with activation map A. For each region $R \in \mathcal{R}$, we apply spatial max-pooling of A and ℓ_2-normalization, denoted by η. We obtain a *global* representation $\phi_A(\mathcal{R})$ by summing over $\mathcal{R}$ and whitening:

Regions detected on OS are more likely to appear in new images than regions detected on FS.

$$\phi_A(\mathcal{R}) := \omega \left(\sum_{R \in \mathcal{R}} \eta \left(\max_{\mathbf{r} \in R} A(\mathbf{r}) \right) \right) \tag{12.10}$$

The difference from (12.7) is that we apply whitening on the aggregated vector and not separately per region.

This is the same as R-MAC [439], but our regions are detected by saliency rather than uniformly distributed.

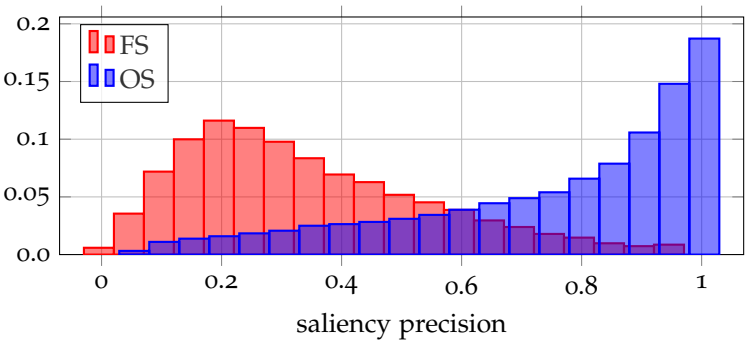


Figure 12.4: Histogram of saliency precision for FS and OS maps on Instre.

12.5 EXPERIMENTS

SETUP We evaluate our global representation on regions detected by EGM on FS and OS maps, denoted as FS.EGM and OS.EGM, respectively. We use the revisited $\mathcal{R}$Oxford and $\mathcal{R}$Paris benchmarks [350]

Regions at different scales form new vertices in the graph; the final image representation is pooled over scales.

as well as Instre [469] with the protocol of [194]. We apply detection to dataset images only, using the provided bounding boxes on queries. We use VGG [409] with GeM [352] pooling, fine-tuned by [352]. We adopt a *multi-scale* approach [137, 352]. We set $\sigma = 2.5$ (12.3), $\alpha = 0.99$ (9.6), $\gamma = 2$ and $\gamma' = 3$ (12.9). We search by cosine similarity and manifold similarity, using global diffusion [194].

High precision means that a saliency map is well aligned to the ground truth bounding boxes.

Using the bounding box annotation of Instre, we evaluate the quality of saliency maps by *precision*, defined as the sum of saliency over ground truth boxes, normalized by the sum over the entire image. We evaluate retrieval performance by mAP.

		MEDIUM		HARD	
METHOD	INSTRE	$\mathcal{R}$OXFORD	$\mathcal{R}$PARIS	$\mathcal{R}$OXFORD	$\mathcal{R}$PARIS
GeM [352]	57.0	62.0	69.3	33.7	44.3
FS.EGM	57.7	63.0	68.7	34.5	43.9
OS.EGM	**61.3**	**64.2**	**69.9**	**35.9**	**46.1**
		DIFFUSION [194]			
GeM [352]	75.0	**69.3**	83.9	41.1	**73.9**
FS.EGM	74.6	71.0	84.1	40.6	72.5
OS.EGM	**77.4**	69.0	**85.4**	**41.9**	72.3

Table 12.1: mAP of our global representation on regions detected by EGM on FS and OS maps against GeM, by cosine or manifold [194] similarity.

RESULTS Figure 12.4 shows histograms of precision of FS and OS maps on Instre. The improvement of OS is impressive.

Table 12.1 compares against GeM [352]. OS.EGM boosts performance in most cases. The gain is more pronounced on Instre, which contains small objects on severe background clutter.

12.6 DISCUSSION

Our region detection approach is dataset specific but requires no supervision and no learning other than computing the k-NN graph centrality. We precisely localize objects without ground truth bounding boxes. We avoid indexing of regional features: Our global features perform well under severe background clutter and occlusions. After its introduction [406], we have extended GOD to use multi-scale GeM [352] representation and shown that Katz centrality performs best among a number of alternatives [407].

Part III

LEARNING

Building on deep network models, we address learning visual representations by exploring data, focusing on solutions assuming only limited or no supervision. We progress from instance-level to category-level tasks and conclude with a study of the sensitivity of models to their input.

13 OUTLINE

This chapter serves as an outline or road map of Part III. *We present historical and more recent background on learning with different levels of supervision developed in 2014 or later, after the deep learning outburst. In this context, we position our own contributions developed in 2018-2019. Our work addresses learning visual representations by exploring data, focusing on solutions assuming only limited or no supervision. It progresses from instance-level to category-level tasks and concludes with a study of the sensitivity of models to their input. We outline the structure of* Part III *in terms of methods, key publications and corresponding chapters.*

13.1 CONTEXT

After or during the development of ideas leading to training deep architectures from scratch in 2012-2016 as outlined in Section 8.1, deep learning is established as the framework of choice for most computer vision tasks. The process commonly involves collecting and annotating task-specific datasets and optionally adapting architectures and defining new loss functions. The network backbone remains the same with few exceptions like feature pyramids [262] for dense tasks or entirely new designs for different inputs, *e.g.* 3d point sets [182].

With most of the research community relying on and improving the same framework, the period from 2014 until today sees unprecedented productivity and progress. All components of the framework are being scrutinized and ideas spread as a matter of days. What is interesting is that further increase of the network depth and the amount of annotated visual data is becoming less important than other trends. We focus here on three trends:

1. learning with less supervision;
2. the convergence of category-level and instance-level tasks; and
3. research directions going from niche to mainstream.

Reducing supervision.

Unsupervised learning is historically common practice for individual components or initialization of the representation. Learning of vocabularies for BoW [83, 413] is unsupervised. Neocognitron [118] and layer-wise pre-training [38, 173, 356] are unsupervised too. Backpropagation allows end-to-end learning of the entire representation, but for this representation to be of high quality, full supervision is common, requiring large amounts of labeled data. Reducing this requirement is an important subject of research.

End-to-end unsupervised learning is common in *autoencoders* [460] and *generative models* [135, 225]. By defining simple tasks where labels

can be obtained without human supervision, it is possible to train arbitrary networks as if learning was supervised. Such tasks include for instance watching moving objects in video [332, 470] or predicting geometric layout [97], rotation [126] or cluster assignment [61]. This is commonly referred to as *self-supervision*.

Learning on limited labeled data may be combined with unsupervised objectives on unlabeled data or more supervision on other datasets, domains or tasks. In such combinations, learning may be joint or in stages, essentially decoupling representation learning from the end task. Examples include *transfer learning* [100, 327, 501], *domain adaptation* [121, 448], *multi-task learning* [231, 283], *incremental learning* [259, 365, 500], *few-shot learning* [261, 362, 415], *semi-supervised learning* [426, 482], *active learning* [394, 396], *learning from noisy labels* [259, 421] and *weakly-supervised learning* [43, 519].

Convergence of category-level and instance-level tasks.

There are tasks where classes at inference are different from classes at learning, for instance *fine-grained classification* [322], *face recognition* [390], *person re-identification* [7], *local descriptor learning* [153] and *instance retrieval* [136, 351]. Those are typically tasks of fine-grained similarity and commonly addressed by *metric learning*, supervised by humans or other algorithms. However, treating them as supervised classification is attracting increasing attention [269, 512].

Few-shot learning is a category-level task but it commonly relies on variants of a nearest neighbor classifier and treated as metric learning too [415, 461]. Instance-level formulations are also common in self-supervised representation learning [490] or classification [292]. Most elements are now common in category-level and instance-level tasks, including architectures, loss functions and representation learning. The main difference remains in the data and labels. Those define precisely the task in terms of *factors of variation* to which invariances need to be learned, *e.g.* within-class appearance variation for category-level and viewpoint changes for instance-level tasks.

Research directions going mainstream.

There are emerging research directions that are part of the natural evolution of the deep learning field, for instance *Neural Architecture Search* (NAS) [268, 525] and *differentiable programming* [54, 115]. However, there are also research directions that have existed long before deep learning without attracting much attention and are now mainstream for different reasons.

We focus on two examples: *few-shot learning* [461] and *adversarial examples* [425]. The former, generalizing from few examples, is becoming increasingly important as it is one of the few challenging tasks where the human skill is not understood yet and, if solved, it would allow instant inference in domains with scarce data. The latter, studying the sensitivity of models to their input by crafting imperceptible perturbations that make them fail, is already essential in the quest for robust models for safety-critical applications.

Our contributions.

In this context, PART III presents part of our work carried out in the period 2018-2019, which addresses learning visual representations by exploring visual data, focusing on solutions assuming only limited or

no supervision and progressing from instance-level to category-level tasks. Our contributions consist of:

1. methods for category-level and instance-level tasks using limited supervision, in particular unsupervised deep *metric learning* [192] and *semi-supervised learning* [193], both based on manifold similarity as developed in PART II;
2. a *few-shot learning* method [261] that studies activation maps and learns multiple layers to convergence for the first time; and
3. an *adversarial attack* [505] that generates invisible perturbations and improves the standard success rate *vs*. distortion performance despite the additional smoothness constraint.

Importantly, using the same machinery that we use for exploring data, we are now able to improve the representation and at the same the improved representation helps explore better. Our attack is also an important step in understanding the role of adversarial examples on the data manifold, which has not been explored much.

13.2 BACKGROUND AND CONTRIBUTIONS

METRIC LEARNING *Principal Component Analysis* (PCA), introduced by Pearson in 1901 [333] uses an orthogonal transformation to find a coordinate system for a dataset such that variables are uncorrelated and given by descending order of variance. As such, PCA can be used for *dimension reduction*, yielding the minimal reconstruction error assuming a linear data model. The classical version of *Multidimensional Scaling* (MDS), introduced by Young and Householder in 1938 [503], is identical but expressed in terms of the distance matrix rather than the covariance matrix.

This subtle difference opens the door to *nonlinear dimension reduction* by replacing the Euclidean by some other metric. An example is *Isomap* [429], which represents the dataset by a k-NN graph and uses a *graph distance* instead. *Kernel* PCA [389] is obtained similarly by expressing PCA in terms of the Gram matrix and then replacing the Euclidean by some other inner product. Hence, it is defined in terms of similarities rather than distances.

The graph distance is defined in terms of shortest paths found by the Dijkstra algorithm [96].

Rather than preserving all pairwise distances or similarities in the dataset, *topology preservation* methods focus on neighbors only. For instance, *Laplacian eigenmaps* [34], formulated as eigenvalue decomposition like all previous methods, minimizes pairwise distances weighted by proximity in the input space. Interestingly, the decomposition obtained by our FSR [189] discussed in Section 10.4 is the same, except for the eigenvalues being soft-weighted by function h_α (9.12) rather than hard-thresholded for dimension reduction.

All the above methods are unsupervised. The nonlinear ones are collectively referred to as *nonlinear dimension reduction*, *manifold learning* or *unsupervised metric learning*. By contrast, *supervised metric learning*, studied by Xing *et al*. in 2003 [491] for supervised classification,

See Lee and Verleysen [253] for a survey of unsupervised methods.

minimizes (resp. maximizes) distances between *positive* (resp. *negative*) pairs of examples, which are given and typically defined as pairs having the same (resp. different) class label. Such learning is useful for k-NN classifiers, using for instance *Neighborhood Component Analysis* (NCA) [130] or a *triplet* loss [477].

See Yang and Jin [494] for a survey of supervised methods.

Embedding.

Given a training set in some input space, the goal of metric learning in general is to learn a mapping from the input space to an *embedding* space. In all methods discussed so far, this mapping is either only linear or restricted to the training set. In the latter case, *out-of-sample extension*, *i.e.*, extension to unseen examples, is not possible unless the training set is memorized [39]. Learning explicit parametric nonlinear mappings such that the training set is not explicitly memorized is where neural networks excel.

A linear mapping means that the distance metric is Mahalanobis [281].

Autoencoders.

Processing one example at a time, Ackley *et al*. derive in 1985 [2] compact representations using nonlinear autoassociative mapping and a bottlenect layer. This is an explicit, learned nonlinear version of PCA including an encoder and a decoder. In 2006, Hinton and Salakhutdinov [171] extend to a deep autoencoder architecture that is pretrained layer-wise and fine-tuned end-to-end, both unsupervised. But how about loss functions requiring at least two examples at a time to specify pairwise distances or similarities?

Siamese architecture.

Baldi and Chauvin introduce in 1993 [27] an architecture where two training examples are mapped by the same network to respective representations, then a binary classifier predicts the pair as matching or non-matching based on the distance between the two representations. This architecture is given the name *Siamese* by Bromley *et al*. in 1994 [52]. It is then used for supervised metric learning by Chopra *et al*. in 2005 [72] and for unsupervised dimension reduction by Hadsell *et al*. in 2006 [150], who introduce the *contrastive* loss.

Modern *deep metric learning*, used in tasks like *fine-grained classification* [322], *face recognition* [390], *person re-identification* [7], *local descriptor learning* [153] and *image retrieval* [136, 351], is mostly supervised by labels provided by humans or other algorithms. Most common loss functions are *contrastive* [150] and a modern version of *triplet* [468]. Because possible pairs and triplets are too many, most of the difficulty shifts from the loss function to *hard example mining* [158, 489], which depends on nearest neighbor search.

Our Mining on Manifolds *(MoM) [192].*

Our *Mining on Manifolds* (MoM) [192] is one of the first unsupervised deep metric learning methods that does not assume external algorithms [136, 351] or some structure of the data [98, 470]. It learns similarity in the embedding space guided by manifold similarity in some initial feature space, as developed in Chapter 10. Compared to supervised methods, it is competitive on fine-grained classification and superior on instance-level retrieval, with labels provided by humans and other algorithms respectively.

SEMI-SUPERVISED LEARNING In 1965, Scudder [392] introduces maybe one of the first semi-supervised classification methods, which learns a classifier on some initial set of labeled examples and then

iteratively makes predictions on unlabeled examples and updates the classifier accordingly.

In the 1970s, *generative models* are studied in a semi-supervised setting. Even before the introduction of EM [91], Hosmer studies in 1973 [176] the iterative *Maximum Likelihood* (ML) estimation of a Gaussian mixture distribution with one component per class, where only a small part of the data is known to originate in a given component. Again, the unlabeled examples participate in the estimation by being soft-assigned to components.

See Chapelle et al. *[66] for a survey of semi-supervised methods before deep learning.*

Also in the 1970s, Vapnik [452] advocates *transductive inference*, making predictions only on the unlabeled test set—whereas in *inductive inference*, a general decision rule is inferred first. Because the entire unlabeled test set is used, transductive implies semi-supervised, but not conversely. Vapnik also introduces in 1998 [451] an early transductive version of the SVM, maximizing the margin over unlabeled as well as labeled examples. In 1999, Joachims [209] introduces what is known today as *transductive* SVM, allowing class overlap and finding an approximate solution in the form of local search.

The cluster assumption*: The decision boundary should lie in low-density regions.*

In the 2000s, two transductive methods known as *Label Propagation* (LP) model the manifold structure of the data by a k-NN graph. One is by Zhu and Ghahramani in 2002 [521], where the solution retains the labels on labeled examples and is harmonic on unlabeled ones [522]. The other is by Zhou *et al*. in 2003 [516], where new labels are allowed on labeled examples. This can be useful *e.g*. when classes overlap or even when labels are noisy. The latter solution uses the same kind of graph filtering as discussed for manifold search in Chapter 10, but with one query per class.

The manifold assumption*: High-dimensional data lies on low-dimensional manifolds.*

> *Deep learning naturally follows the inductive approach, in the sense that not only the learned model is expected to work on unseen examples, but the training set is expected to be discarded.*

Weston *et al*. introduce in 2008 [482] maybe one of the first deep learning approaches. Like *metric learning* [34, 150], it adds an unsupervised loss to minimize the distance of the embeddings of two examples that are close in the input space. *Pseudo-label*, introduced by Lee in 2013 [252], is a modern incarnation of the earliest approaches [392], treating predictions as if they were true labels.

The smoothness assumption*: If two examples are close in a high-density region, then so should be the corresponding predictions.*

Many modern approaches add an unsupervised *consistency* loss on all data. For instance, *temporal ensemble* [241] and *mean teacher* [426] minimize the distance of two embeddings of the same example obtained by two similar models. This suggests a dual to the smoothness assumption: If two models are close in the parameter space, then so should be the corresponding predictions.

Our *Deep Label Propagation* (DLP) [193] is an inductive incarnation of LP in a modern deep learning setting. It uses pseudo-labels on unlabeled data too. However, our pseudo-labels are inferred by LP [516] rather than by the classifier directly [252]. Because the graph is based on the network embedding, DLP alternates between performing LP

Our Deep Label Propagation *(DLP) [193].*

and updating the embedding. Weighting unlabeled examples by certainty provides robustness to incorrect pseudo-labels. DLP is found to be complementary to consistency losses.

Motivation from human skills.

FEW-SHOT LEARNING Humans can learn new concepts from just one example. This skill is developed quite early and applies to language, vision or associations between language and vision. Berko shows in 1958 [41] that we unconsciously learn linguistic rules by studying how children, aged 4-7, can solve linguistic tasks like forming the plural, past tense or possessive of nonsense words. Carrey studies in 1978 [59] how children, aged 3, acquire a new word associated with a particular color. Landau *et al*. show in 1988 [245] that children, aged 2-3, learn new objects of particular shape more easily when named with nonsense words than when not.

Early models.

Early computational models attribute this human skill to "*prior knowledge*" and devise different ways of encoding it. In 1997, Yip and Sussman [499] induce constraints on linguistic representations of speech from a corpus of common nouns and verbs. In 2000, Miller *et al*. [299] generalize over geometric transformations in learning novel handwritten digits from one example by learning a density of transformations on a corpus of digits. In 2003, Fei-Fei *et al*. [111] learn a mixture distribution over appearance and shape from one example of a novel generic object category, *e.g*. face or motorbike, by inducing a conjugate prior from other categories. In 2005, Bart and Ullman [29] learn novel classes from one example by using informative patches of similar, familiar classes and adapting them to the example at hand. In 2011, Lake *et al*. [243] study a generative model of handwritten character composition from strokes and parse novel examples using a library of strokes learned on different alphabets. In 2012, Mensink *et al*. [290] use metric learning for the *Nearest Class Mean* (NCM) classifier to learn novel classes from few examples.

For instance, italicized digits are modeled by a shearing transformation.

Deep learning.

In the deep learning era, learning novel object categories from one or few examples becomes popular in 2016 when Vinyals *et al*. [461] form mini-batches called *episodes* from a labeled set of *base* classes to mimic classification tasks from a limited *support* set over a distinct set of *novel* classes. Inference is based on a soft nearest neighbor classifier. Snell *et al*. improve this model in 2017 [415] by averaging the embeddings of support examples into class *prototypes*. This is the same as the NCM classifier, which is used by Waltner *et al*. in 2016 [465] with a linear embedding on top of CNN features. The problem is essentially *supervised metric learning*, particularly when the loss function is defined beyond pairs or triplets [292, 322, 449].

Meta-learning.

However, given the support examples, there is opportunity to continue learning. *Meta-learning* [388, 459] refers to learning at two levels, where generic knowledge is acquired before adapting to more specific tasks. In few-shot learning, this translates to learning on base classes how to learn from few novel-class examples without overfitting. Metric learning is then seen as learning how to compare queries with support examples [461] or class prototypes [415, 465]. *Optimization*

meta-learning [116, 362] amounts to learning a model that is easy to fine-tune in few steps. *Augmentation meta-learning* [155, 472] refers to learning how to *generate* novel-class examples.

The challenge applies equally to metric learning [512] and face recognition in particular [355, 466].

The meta-learning setup is challenged by Gidaris and Komodakis [125], who rather learn on base classes a simple parametric classifier, based on *cosine similarity*. The same classifier is introduced independently by Qi *et al*. [347], who further fine-tune the network, assuming access to the base class training set.

Our Dense Classification (DC) *[261].*

Our *Dense Classification* (DC) [261] builds on this simple classifier by applying it densely over all locations in an image. This is a form of *implicit data augmentation* of dense shifts and crops with a single network evaluation. As a result, we obtain smoother activation maps that are more aligned with objects. This is the first time that activation maps are studied in the context of few-shot learning.

Our implanting *[261].*

In the same work [261], we introduce *implanting*. Neural implants are convolutional filters in a new processing stream parallel to a previously trained network, which remains fixed. Few implants learn new, task-specific features with reduced risk of overfitting. Implanting thus enables training of multiple layers to convergence in the few-shot regime, departing from prior methods that train either only the last layer [155] or only for a few iterations [362].

ADVERSARIAL EXAMPLES Shortly after the success of AlexNet in 2012 [235], a new challenge appears. Szegedy *et al*. find in 2013 [425] that neural networks are sensitive to their inputs: Adding an imperceptible perturbation makes the classifier fail.

Adversarial examples before deep learning.

Such *adversarial examples* are not new: In 2009, Hsu *et al*. [179] attack the SIFT detector either by manipulating pixels in the input image to introduce duplicate local extrema, or by replacing patches by similar patches from an image database, where no features are detected. A similar search over a SIFT descriptor database is used by Weinzaepfel *et al*. [478] to reconstruct the input image from its collection of SIFT descriptors. With CNNs being differentiable, Szegedy *et al*. [425] rather modify the input image by using back-propagation to minimize a loss function. The objective is to modify the classifier prediction while keeping the input distortion low.

Why study adversarial examples?

Adversarial examples are not particular to nonlinearities either: In 2015, Goodfellow *et al*. [133] attribute their existence in linear models to the *high dimensionality* of the input space. The reason why their study becomes so important can be rather sought in the fact that we now trust CNNs in safety-critical tasks like driving cars. To make matters worse, adversarial examples are transferable from one model to another [270] and there exist *Universal Adversarial Perturbations* (UAPs) that apply to several models [304].

Adversarial training.

Goodfellow *et al*. [133] also introduce a single-step attack that is fast enough to generate adversarial examples on the fly during training. Such *adversarial training* is a defense mechanism that improves the robustness to adversarial attacks. Kurakin *et al*. then introduce in 2016 [237] a more powerful iterative attack. Unless such a powerful it-

erative attack is used [279], adversarial training is easily broken [442]. As a result, training robust models becomes very expensive.

Adversarial robustness vs. generalization.

The situation becomes more complex when Tsipras *et al*. [445] prove on a simple data model that adversarial robustness comes at the additional cost of dropping accuracy on *benign* (not adversarial) examples. But this data model is just chosen to simplify the theoretical proof. So, is this result true in general? Stutz *et al*. [419] show empirically that, under the manifold assumption, adversarial examples leave the manifold; on-manifold adversarial examples exist and using the them from adversarial training boosts generalization.

Stutz *et al*. [419] use an autoencoder [247] to approximate the underlying manifold of each class. As a result, experiments are constrained to tiny images. To generate more realistic on-manifold adversarial examples, a general property of natural images can be used instead: *smoothness*. Previous attempts in this direction include *e.g*. spatial filtering [518], harmonic functions [169] and attacking in the Fourier domain [146]. All of these constructs are independent of the input image and none of these attacks is competitive.

Smoothness.

Our smooth adversarial examples *[505].*

Our *smooth adversarial examples* [505] are an attempt to improve the visual quality of adversarial examples by smoothing the noisy gradient *guided* by the input image, similarly to *style transfer* [277, 345]: The perturbation is locally smooth on flat areas of the input image, but it may be noisy on textured areas and sharp across edges. Smoothing follows Chapter 9 and the graph is defined over pixels.

Despite the additional smoothness constraint, which reduces the effective dimensionality of the perturbation, our attack has the same success rate at lower distortion compared to a regular attack. Even when the distortion is higher, the perturbation is totally invisible.

13.3 STRUCTURE

Chapter 14 addresses *metric learning* for ranking tasks including fine-grained classification and image search. It discusses *Mining on Manifolds* (MoM) [192], an unsupervised method based on differences between Euclidean and manifold similarity.

Chapter 15 studies *Semi-Supervised Learning* (SSL), where labels are limited. It presents *Deep Label Propagation* (DLP) [193], an inductive version of the classic *Label Propagation* (LP) [516].

Chapter 16 discusses the more challenging problem of *Few-Shot Learning* (FSL), where not only labels but raw data is limited too. It presents two solutions, *Dense Classification* (DC) and *implanting* [261], which do not rely on meta-learning.

Finally, Chapter 17 concludes with a study of *adversarial examples*, that is, imperceptible perturbations of a given input that make a model fail. We argue that popular attacks have a fundamental limitation in terms of *imperceptibility* and present *smooth adversarial examples* [505], an attempt to address this limitation.

14 METRIC LEARNING

We introduce an unsupervised framework for hard example mining [192]. The only input is a collection of images and an initial CNN representation. Positive examples are distant points on a manifold; negative are nearby points on different manifolds. Both are revealed by disagreements between Euclidean and manifold similarities, and can be used with any discriminative loss. We apply to fine-grained classification and image retrieval.

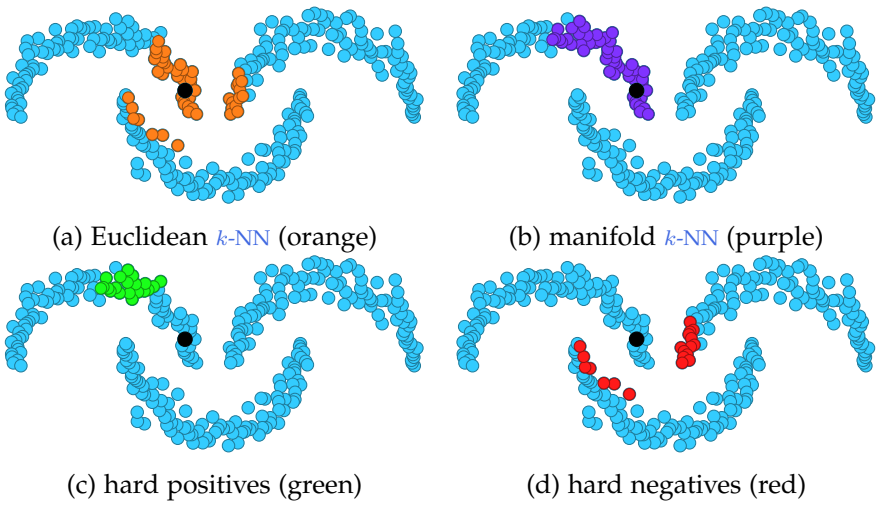
14.1 INTRODUCTION

It is common to start with a pre-trained network [165, 409, 423] and apply *metric learning* [72, 158, 468] to fine-tune the network for a task like fine-grained classification [158, 475], particular object retrieval [136, 351], or person re-identification [7]. Loss functions like contrastive [72], triplet [468] or batch-level [322] are typically used on *hard examples*, found by *sampling* or *mining* mechanisms [158, 308].

Those are training examples for which the network performs poorly.

Training labels may be found by existing algorithms [136, 351], spatial [98] or temporal [470, 471] structure of the data, but in most cases by human supervision [25, 322]. On one hand, by using *class labels*, we miss the opportunity of learning from unlabeled data, learned class representations are unimodal [370] and the problem remains supervised classification. On the other hand, conventional *manifold learning* methods are unsupervised [34, 386, 430] but have difficulties generalizing to new data.

Most do not learn an explicit mapping from the input to the embedding space.



(a) Euclidean k-NN (orange) (b) manifold k-NN (purple)

(c) hard positives (green) (d) hard negatives (red)

Figure 14.1: Given an anchor point (black) and its k nearest Euclidean (NN_k) and manifold (NN_k^m) neighbors in a dataset (cyan), we choose *positive* examples as $\mathrm{NN}_k^m \setminus \mathrm{NN}_k$, and *negative* as $\mathrm{NN}_k \setminus \mathrm{NN}_k^m$. The selection is fully unsupervised, including anchors.

Learning a new representation dispenses the need to store the dataset and compute the manifold similarity, and generalizes better to unseen datasets.

We attempt to bridge this gap by introducing *Mining on Manifolds* (MoM) [192], an unsupervised hard example mining mechanism using manifold similarity [194] on an initial representation space of unlabeled data. As illustrated in Figure 14.1, given an anchor point, neighbors on the manifold that are not Euclidean neighbors are considered *positive* examples to be attracted to the anchor. Conversely, Euclidean neighbors that are not manifold neighbors are considered *negative* and should be repelled. We apply our method to *fine-grained classification* and *particular object retrieval*.

14.2 PRELIMINARIES

An example may be an image, video, image region, local patch, etc.

PROBLEM We are given a *training set* $X = \{x_1, \ldots, x_n\} \subset \mathcal{X}$, where $\mathcal{X}$ is an *input space*. Function $f_\theta : \mathcal{X} \to \mathbb{R}^d$ maps an example $x \in \mathcal{X}$ to a vector $\mathbf{u} = f_\theta(x)$ in a d-dimensional *embedding space* $\mathcal{U}$, where θ is a set of parameters to be learned. Examples X are represented by a set of features $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \subset \mathcal{V}$, where $\mathcal{V}$ is a *feature space* and $\mathbf{v}_i = \phi(x_i)$ for $i \in [n]$. Function ϕ may be f_{θ_0}, *i.e.* the same model f with parameters θ_0, supervised or not, or a different model.

We write $[n] := \{1, \ldots, n\}$ *for* $n \in \mathbb{N}$.

Two images are matching *if they are* visually similar, *where "similarity" depends on the application.*

The goal of *metric learning* is to learn the parameters θ such that *matching* examples are mapped to nearby points in the embedding space $\mathcal{U}$, while *non-matching* examples are well separated. For instance, a matching (non-matching) *pair* consists of a *reference* or *anchor* example x^r and a *positive* (*negative*) example x^+ (x^-). Alternatively, we may use a *triplet* (x^r, x^+, x^-). Assuming only the input examples X and their features V, our goal is to mine such labels without any supervision [72, 158, 308, 468], existing algorithms [136, 351] or assumptions on the structure of the training data [98, 470, 471].

MANIFOLD SIMILARITY By $\mathrm{NN}_k(\mathbf{v})$ ($\mathrm{NN}_k^m(\mathbf{v})$) we denote the k *Euclidean* (*manifold*) nearest neighbors of vector $\mathbf{v}$ in V, *i.e.*, the k most similar examples in V by Euclidean (manifold) *similarity function* $s : \mathcal{V}^2 \to \mathbb{R}$ ($s^m : V^2 \to \mathbb{R}$). To define s^m, we form the adjacency matrix W of the Euclidean k-NN graph of V as in Section 10.2. Then, following Chapter 9, we normalize it into $\mathcal{W}$ by (9.3) and compute the $n \times n$ regularized Laplacian $\mathcal{L}_\alpha$ by (9.6), where $\alpha \in [0, 1)$. Following Section 10.3 and (10.8), the solution $\mathbf{z}_i^\star$ of the linear system

In fact, s^m is a function of the entire set V.

As in Chapter 10, we solve this system by CG [317].

$$\mathcal{L}_\alpha \mathbf{z} = \mathbf{e}_i, \tag{14.1}$$

That is, $\mathbf{e}_i$ is the i-th column of an $n \times n$ identity matrix.

where $\mathbf{e}_i$ is the standard n-dimensional basis vector, expresses the manifold similarity of $\mathbf{v}_i \in V$ to all vectors in V. We thus define the *manifold similarity* of $\mathbf{v}_i, \mathbf{v}_j \in V$ as

$$s^m(\mathbf{v}_i, \mathbf{v}_j) := \mathbf{z}_i^\star(j), \tag{14.2}$$

$\mathcal{L}_\alpha^{-1}$ is dense but we never compute it; we only compute its columns as needed.

i.e., the j-th element of $\mathbf{z}_i^\star$. Observe that s^m is symmetric because in fact $s^m(\mathbf{v}_i, \mathbf{v}_j)$ is the (i, j)-th element of $\mathcal{L}_\alpha^{-1}$, which is symmetric. Finally, the *manifold nearest neighbors* $\mathrm{NN}_k^m(\mathbf{v}_i)$ of $\mathbf{v}_i$ in V are the elements of V corresponding to the k maximum elements of $\mathbf{z}_i^\star$.

14.3 HARD EXAMPLE SELECTION

POSITIVES Given an *anchor* or *query* example x^r and the corresponding feature $\mathbf{v}^r = \phi(x^r)$, we define the *positive pool* $N^+(x^r)$ of x^r as the examples in X that correspond to the manifold neighbors of $\mathbf{v}^r$ that are not Euclidean neighbors in the feature space:

k controls the diversity of positives, with larger values corresponding to harder *examples.*

$$N^+(x^r) := \{x \in X : \phi(x) \in \mathrm{NN}_k^m(\mathbf{v}^r) \setminus \mathrm{NN}_k(\mathbf{v}^r)\}. \tag{14.3}$$

As illustrated in Figure 14.1(c), these examples are assumed to be *matching* with x^r, yet not retrieved well in the feature space. In the embedding space, positives should be *attracted* to the anchor so that Euclidean and manifold neighbors agree.

Mining hard positive examples, in contrast to negatives, is not common.

NEGATIVES Similarly, we define the *negative pool* $N^-(x^r)$ of anchor x^r as the examples in X that correspond to the Euclidean neighbors of $\mathbf{v}^r$ that are not manifold neighbors in the feature space:

$$N^-(x^r) := \{x \in X : \phi(x) \in \mathrm{NN}_k(\mathbf{v}^r) \setminus \mathrm{NN}_k^m(\mathbf{v}^r)\}. \tag{14.4}$$

As illustrated in Figure 14.1(d), these examples are assumed to be *non-matching* with x^r, yet too close in the feature space. In the embedding space, negatives should be *repelled* from the anchor.

It is common practice, and known to be beneficial [410], to select hard negative examples.

ANCHORS We need a diverse collection of anchors that have many neighboring examples in the feature space V. This is achieved by the *modes* of the k-NN graph G. Following [71], we find those as the local maxima of the stationary distribution on G. Details are given in [192]. We denote by Q the resulting *anchor* or *query set*.

14.4 TRAINING

At a given training epoch, for each anchor $x^r \in Q$, a positive example x^+ is drawn at random from its positive pool $N^+(x^r)$, and a negative x^- from a subset of its negative pool $N^-(x^r)$. This subset consists of the examples corresponding to the Euclidean nearest neighbors of $f_\theta(x^r)$ in the embedding space, where θ is the current set of parameters. The training set for the epoch is the set of tuples (x^r, x^+, x^-) for $x^r \in Q$. Given such a tuple with corresponding embeddings $\mathbf{u}^r := f_\theta(x^r)$, $\mathbf{u}^+ := f_\theta(x^r)$ and $\mathbf{u}^- := f_\theta(x^-)$, we use the *contrastive loss* [150],

While Q, N^+, N^- are computed once, hard negative sampling uses the current network representation.

$$\ell_c(\mathbf{u}^r, \mathbf{u}^+, \mathbf{u}^-) := \|\mathbf{u}^r - \mathbf{u}^+\|^2 + [m - \|\mathbf{u}^r - \mathbf{u}^-\|]_+^2, \tag{14.5}$$

combining a positive and a negative pair, or the *triplet loss* [468]

$$\ell_t(\mathbf{u}^r, \mathbf{u}^+, \mathbf{u}^-) := [m + \|\mathbf{u}^r - \mathbf{u}^+\|^2 - \|\mathbf{u}^r - \mathbf{u}^-\|]_+^2, \tag{14.6}$$

We write $[x]_+ := \max(x, 0)$ for the positive part *of $x \in \mathbb{R}$.*

where m is a *margin* parameter.

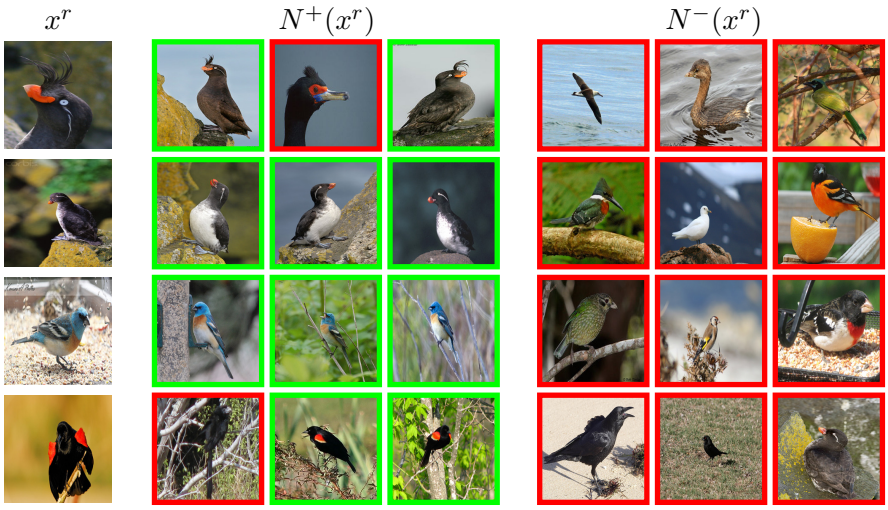


Figure 14.2: CUB200-2011 anchor images x^r, positives $N^+(x^r)$ and negatives $N^-(x^r)$ used for fine-grained classification. True positives (negatives) in green (red).

14.5 APPLICATIONS

The goal is to learn embeddings that discriminate instances of the same class from instances of different classes.

FINE-GRAINED CLASSIFICATION We use the *Caltech-UCSD Birds* (CUB)200-2011 dataset [464], comprising 200 bird species as classes. An embedding is learned on 100 classes, while the remaining 100 are used for testing [322]: Given a test query, the remaining test images of the same class should be top-ranked according to embedding similarity to the query [158, 475]. This is evaluated by *Recall@k*, while *Normalized Mutual Information* (NMI) [286] measures clustering quality. *We only use labels for testing*.

The feature space $\mathcal{V}$ may have been learned using supervision on some other training set/domain.

Figure 14.2 shows examples of anchors with positives and negatives. True positives (negatives) are examples with same (different) label as the anchor. Despite the absence of labels at training, we achieve a very clean negatives and a reasonably clean positives.

PARTICULAR OBJECT RETRIEVAL We use a 1M subset of the same Flickr7M collection used by [351] to learn the embedding. Test sets comprise small objects (Instre [469]), natural scenery (Holidays [202]) and buildings/landmarks (Oxford5k [341] and Paris6k [342]). For large-scale experiments, we add 100k distractor images [341] to Oxford5k and Paris6k, referred to as Oxford105k and Paris106k respectively. Given a test query, the remaining test images depicting the same *instance* should be top-ranked according to embedding similarity [136, 351], as evaluated by mAP. Modern methods use existing algorithms to mine pairs [351] or triplets [136] on the training set. *We only use the provided features*.

Similarity is even more fine-grained than in bird species. Objects are less deformable, but there is extreme diversity in viewpoint, lighting, occlusion and clutter.

Figure 14.3 shows examples of anchors with positives and negatives. Positives are challenging, while negatives are interesting, depicting different objects, which still look similar.

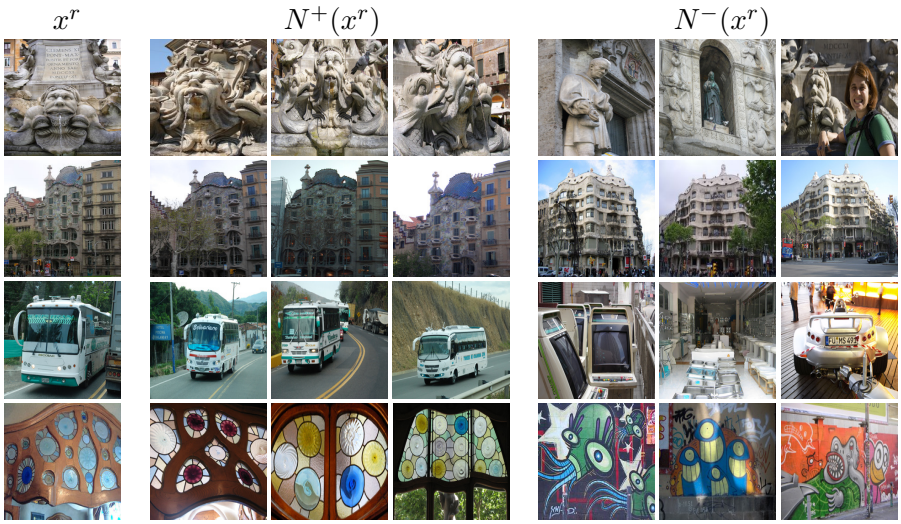


Figure 14.3: Flickr7M anchor images x^r, positives $N^+(x^r)$ and negatives $N^-(\mathbf{x}^r)$ used for particular object retrieval.

14.6 EXPERIMENTS

SETUP In both applications, the initial model f_θ is a network pre-trained on ImageNet and the feature set V consists of R-MAC [439] on the last convolutional layer of the same model. We use Euclidean similarity $s(\mathbf{v}, \mathbf{v}') := [\mathbf{v}^\top \mathbf{v}']_+^3$ and set $\alpha = 0.99$ as in Section 10.5 for manifold similarity. We set k to 30 in the k-NN graph, 50 for positives (14.3) and 100 (10,000) for negatives (14.4) for classification (retrieval). We use the *triplet loss* (14.6) with $m = 0.5$ for classification and *contrastive loss* (14.5) with $m = 0.7$ for retrieval. Training hyperparameters are detailed in [192].

All vector representations are ℓ_2-normalized.

Negative pools are constrained to the 50 hardest examples.

METHOD	LABELS	R@1	R@2	R@4	R@8	NMI
Baseline		35.0	46.8	59.3	72.0	48.1
Cyclic match [256]		40.8	52.8	65.1	76.0	52.6
MoM (ours)		**45.3**	**57.8**	**68.6**	**78.4**	**55.0**
Triplet+semi-hard [390]	✓	42.3	55.0	66.4	77.2	55.4
Lifted-structure [322]	✓	43.6	56.6	68.6	79.6	56.5
Triplet+ [158]	✓	45.9	57.7	69.6	79.8	58.1
Clustering [416]	✓	48.2	61.4	71.8	81.9	59.2
Triplet+++ [158]	✓	49.8	62.3	74.1	83.3	59.9

Table 14.1: Recall@k and NMI on CUB-200-2011 using GoogLeNet. Comparing our MoM against [256] and baseline as pre-trained on ImageNet. "Labels" indicates using ground-truth labels at training.

FINE-GRAINED CLASSIFICATION We use GoogLeNet [423] baseline as pre-trained on ImageNet, we reduce to an embedding dimensionality of $d = 64$ by adding a fully-connected layer and we fine-tune with *triplet loss*. We use all training images as anchors.

This is affordable since the training set is small (6k images).

As shown in Table 14.1, our approach outperforms the *unsupervised* approach [256] and competes or even outperforms *supervised* methods that are using ground-truth labels on the training set.

METHOD	HOL	INSTRE	OXF5K	OXF105K	PAR6K	PAR106K
TESTING ON MAC [439]						
Baseline	79.4	48.5	58.5	50.3	73.0	59.0
SfM [351]	81.4	48.5	**79.7**	73.9	82.4	74.6
MoM (ours)	**82.6**	**55.5**	78.7	**74.3**	**83.1**	**75.6**
TESTING ON R-MAC [439]						
Baseline	87.0	55.6	68.0	61.0	76.6	72.1
SfM [351]	84.4	47.7	77.8	70.1	84.1	76.8
MoM (ours)	**87.5**	**57.7**	**78.2**	**72.6**	**85.1**	**78.0**

Table 14.2: mAP on particular object retrieval using VGG. Comparing our MoM against SfM-based fine-tuning [351] and baseline as pre-trained on ImageNet. Fine-tuning performed for MAC [439].

We use unsupervised whitening *[201] for the baseline and* supervised whitening *[297] for fine-tuned networks, as in [351].*

PARTICULAR OBJECT RETRIEVAL We use VGG [409] baseline as pre-trained on ImageNet and we fine-tune MAC representation with *contrastive loss*. At testing, we use both MAC and R-MAC [439]. We compare against [351], which mines labels via a *Structure-from-Motion* (SfM) pipeline based on conventional features. Anchor selection is essential here, due to the size of the training set.

As shown in Table 14.2, we improve over the baseline as well as [351] in most cases. We even improve on Holidays and Instre, where [351] shows little improvement or is even inferior to the baseline.

14.7 DISCUSSION

In this work, we depart from using ground-truth labels or conventional algorithms in metric learning for fine grained similarity. We experiment with standard contrastive and triplet loss, but there are many other losses that are functions of more than two or three examples [30, 236, 248, 322, 449, 489]. It is possible to iterate our approach, updating the graph and the pools based on the current embedding space, then updating the embeddings, and so on.

15 SEMI-SUPERVISED LEARNING

Label Propagation *(LP) [516] is a* transductive *method for semi-supervised classification. In this work, we extend it to an* inductive *setting, introducing* Deep Label Propagation *(DLP) [193]. We generate pseudo-labels for the unlabeled training examples by LP and train a deep neural network to classify unseen examples. LP uses a nearest neighbor graph of the training set that we create based on the embeddings of the same network. Therefore we alternate between performing LP and updating the network.*

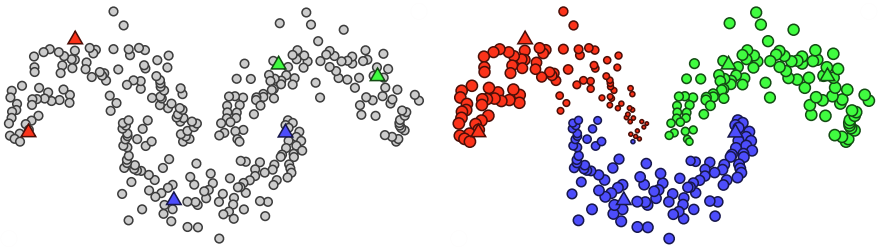
15.1 INTRODUCTION

Deep neural networks require large amounts of training examples. Visual data is available in large quantities, but supervision is provided either by humans, which is expensive, or automatically on proxy tasks [61, 97, 126, 137, 192, 332, 352, 471, 490], which is inferior or not appropriate for high-level tasks like classification.

Semi-Supervised Learning (SSL) is becoming increasingly important because it can combine data carefully labeled by humans with abundant unlabeled data to train powerful networks. In *transductive inference* [516, 522], prediction is restricted to the unlabeled examples of the training set. In modern *inductive learning*, the goal is generalization to unseen data, while the training data is discarded. This is achieved *e.g.* by combining a supervised loss on labeled data with unsupervised objectives on all data [426, 482]; or, an existing classifier can be used to assign *pseudo-labels* [252, 381].

Using a powerful classifier trained on carefully labeled data can provide high-quality pseudo-labels, opening the door to learning from large scale unlabeled data [354].

Classic transductive methods have not been fully exploited in the inductive setting. The same holds for the *manifold assumption*—that similar examples should get the same prediction. In this work, we use *Label Propagation* (LP) [516], a transductive classification method based



(a) Training examples and labels (b) LP predictions and certainty

Figure 15.1: Toy example of *Label Propagation* (LP). Triangles (circles) indicate labeled (un-labeled) examples. (a) Gray (color) indicates unlabeled (labeled by color) examples. (b) LP predictions by (15.5) (certainty by (15.7)) indicated by color (size).

on the manifold assumption, to infer pseudo-labels for unlabeled data and use them to train a classifier. We call this inductive extension *Deep Label Propagation* (DLP) [193].

DLP alternates between two steps: (a) Train the network on labeled and pseudo-labeled data. (b) Use the k-NN graph of the embeddings of the network to perform LP, inferring pseudo-labels for unlabeled examples, as well as a certainty-based weight per example. Training is performed on all data, weighted by certainty.

15.2 PRELIMINARIES

PROBLEM We are given a *training set* $X := \{x_1, \dots, x_n\} \subset \mathcal{X}$, where $\mathcal{X}$ is an *input space*. A subset $X_L := \{x_i : i \in L\}$ with $L \subset [n]$ is labeled by $\mathbf{y}_L := (y_i)_{i \in L}$ with $y_i \in [c]$, where c is a number of *class labels*. The remaining examples $X_U := X \setminus X_L$ with $U := [n] \setminus L$ are unlabeled. The goal is to use all examples X and labels $\mathbf{y}_L$ to train a classifier that maps new examples from $\mathcal{X}$ to class labels.

We write $[n] = \{1, \dots, n\}$ for $n \in \mathbb{N}$.

CLASSIFIER The *classifier network* $f_\theta : \mathcal{X} \to \mathbb{R}^c$, where θ are the network parameters, maps an input example $x_i \in \mathcal{X}$ to a vector $f_\theta(x_i) \in \mathbb{R}^c$ of class probabilities. The corresponding *prediction* $\hat{y}_i \in [c]$ is the class of maximum probability

$f_\theta(x_i)_j$ denotes the j-th element of vector $f_\theta(x_i)$.

$$\hat{y}_i := \arg\max_{j \in [c]} f_\theta(x_i)_j. \tag{15.1}$$

For example, f_θ may consist of ϕ_θ followed by an FC layer and softmax.

As a by-product of learning, we assume access to an *embedding network* $\phi_\theta : \mathcal{X} \to \mathbb{R}^d$, mapping x_i to a *feature vector* $\mathbf{v}_i := \phi_\theta(x_i)$.

SUPERVISED LOSS In *supervised classification*, network parameters θ are learned by minimizing a *supervised* cost function of the form

This function applies only to labeled examples X_L.

$$J_s(X_L, \mathbf{y}_L; \theta) := \sum_{i \in L} \ell\left(f_\theta(x_i), y_i\right). \tag{15.2}$$

A standard choice for the *loss function* ℓ is *cross-entropy*, given by $\ell(\mathbf{p}, j) := -\log p_j$ for $\mathbf{p} \in \mathbb{R}^c_+$, $j \in [c]$.

15.3 BACKGROUND

PSEUDO-LABELING By assigning a pseudo-label $\hat{y}_i$ to each example x_i for $i \in U$ and writing $\hat{\mathbf{y}}_U := (\hat{y}_i)_{i \in U}$, the following *pseudo-label* cost function applies to unlabeled examples X_U:

$$J_p(X_U, \hat{\mathbf{y}}_U; \theta) := \sum_{i \in U} \ell\left(f_\theta(x_i), \hat{y}_i\right). \tag{15.3}$$

An example is [252], where θ is learned by (15.2) on X_L and then pseudo-labels are assigned by network predictions (15.1) on X_U.

LABEL PROPAGATION Following the definitions of Chapter 9, given an $n \times n$ adjacency matrix W of the training set, we normalize it into $\mathcal{W}$ by (9.3) and compute the $n \times n$ regularized Laplacian $\mathcal{L}_\alpha$ by (9.6), where $\alpha \in [0, 1)$. We also define an $n \times c$ *label matrix* Y by

The i-th row of Y is an one-hot encoding of label y_i if $i \in L$ and zero otherwise.

$$y_{ij} := \begin{cases} 1, & \text{if } i \in L \wedge y_i = j \\ 0, & \text{otherwise} \end{cases} \tag{15.4}$$

for $i, j \in [n]$. Defining the $n \times c$ matrix $Z^\star$ as the solution of linear system (9.9), LP [516] makes a prediction for $x_i \in X$ according to

We write the (i, j)-th element of matrix A as a_{ij}.

$$\hat{y}_i := \arg\max_{j \in [c]} z^\star_{ij}. \tag{15.5}$$

As illustrated in the toy example of Figure 15.1, LP is a *transductive* method: Its predictions are constrained to X.

To extend to unseen examples in $\mathcal{X}$, one needs to store the training set X.

Similarly to (9.17), $Z^\star$ minimizes a quadratic cost where a *smoothness term* encourages nearby examples get the same predictions, while a *fitness term* encourages predictions to respect labels Y [516].

15.4 DEEP LABEL PROPAGATION

In an inductive framework, the *smoothness term* becomes an unsupervised loss encouraging consistency between nearby example predictions. Indeed, such solution is adopted by [482]. This is not very efficient because the gradient propagates from each example to its neighbors only at each iteration. Our idea is thus the following:

Solving (15.6) for $Z^\star$ is efficient because it does not need feeding data to the network. Then, the learning process drives all examples directly to $Z^\star$.

> *Instead of just encouraging nearby examples to get the same predictions, we encourage all examples to get predictions directly as obtained by label propagation, as if they were all labeled.*

PSEUDO-LABELS Given network parameters θ, let $V := \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be a feature set, where $\mathbf{v}_i := \phi_\theta(x_i)$ for $i \in [n]$. We form the adjacency matrix W of the k-NN graph of V as discussed in Section 10.2. Defining matrices $\mathcal{L}_\alpha$ (9.6) and Y (15.4) as discussed in Section 15.3, we perform LP by solving the linear system (9.9)

The same solution has been used for interactive image segmentation [224], semantic image segmentation [64] and image retrieval [194].

$$\mathcal{L}_\alpha Z = Y \tag{15.6}$$

using the CG method, which is faster than the iterative process of [516]. Given the solution $Z^\star$, we infer pseudo-labels $\hat{\mathbf{y}}_U := (\hat{y}_i)_{i \in U}$, where $\hat{y}_i$ is given by (15.5).

CERTAINTY AND BALANCING Pseudo-labels are not equally certain. Apart from prediction $\hat{y}_i$ (15.5), LP can give a vector of class probabilities $\eta_1(\mathbf{z}^\star_i)$, where $\mathbf{z}^\star_i$ is the i-th row of $Z^\star$ and η_1 denotes ℓ_1-normalization. We can then measure the certainty of $\hat{y}_i$ and assign example x_i a weight $\omega_i \in [0, 1]$ for $i \in [n]$, defined by

We write $\eta_1(\mathbf{z}) := \mathbf{z} / \|\mathbf{z}\|_1$ for $\mathbf{z} \in \mathbb{R}^c$.

$$\omega_i := 1 - \frac{H(\eta_1(\mathbf{z}^\star_i))}{\log(c)}, \tag{15.7}$$

where $H(\mathbf{p}) := -\sum_{j\in[c]} p_j \log p_j$ is the *entropy* of distribution $\mathbf{p} \in \mathbb{R}^c_+$. As illustrated in Figure 15.1(b), the manifold containing "red" labels is still predicted as "red", although several examples are far away from the labels, thus having little certainty.

In addition, to encourage balance of pseudo-labels over classes, we assign class j a weight $\beta_j := (|L_j| + |U_j|)^{-1}$, where L_j (U_j) is the number of examples labeled (pseudo-labeled) as class j for $j \in [c]$.

In contrast to (15.3), pseudo-labels originate in label propagation rather than network predictions.

WEIGHTED COST Given the above definitions of example and class weights, the following *weighted* cost function applies to both labeled and pseudo-labeled examples

$$J_w(X, \mathbf{y}_L, \hat{\mathbf{y}}_U; \theta) := \sum_{i\in L} \beta_{y_i} \ell\left(f_\theta(x_i), y_i\right) + \sum_{i\in U} \omega_i \beta_{\hat{y}_i} \ell\left(f_\theta(x_i), \hat{y}_i\right). \quad (15.8)$$

This is the sum of weighted versions of J_s (15.2) and J_p (15.3), where $\hat{y}_i$ is now defined by (15.5).

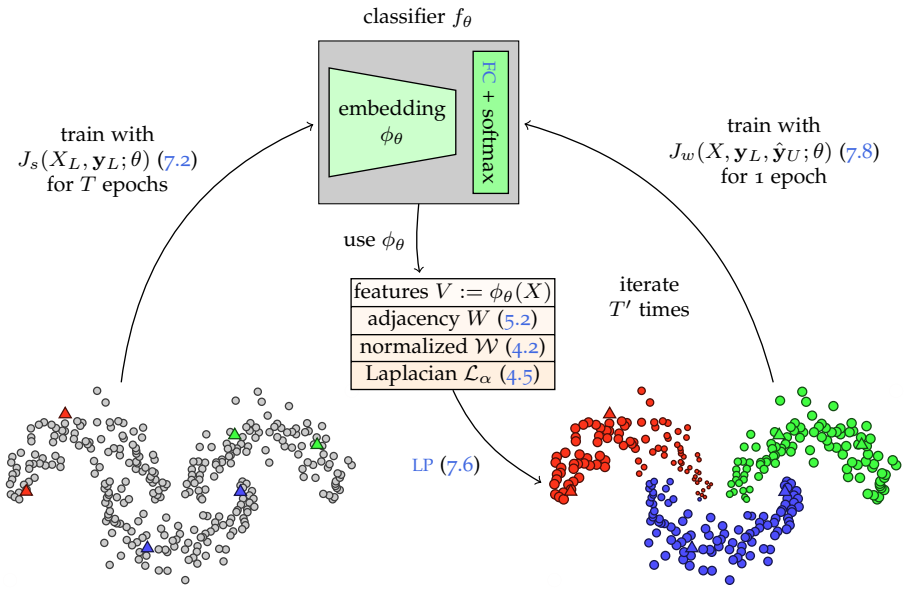


Figure 15.2: *Deep Label Propagation* (DLP). We first learn a classifier f_θ on labeled examples X_L (15.2). We then iterate (a) computing a k-NN graph of the current features $V := \phi_\theta(X)$ and its regularized Laplacian $\mathcal{L}_\alpha$ (9.6), (b) performing LP (15.6), and (c) training f_θ with true labels $\mathbf{y}_L$ on labeled examples X_L and pseudo-labels $\hat{\mathbf{y}}_U$ (15.5) + weights (15.7) on unlabeled examples X_U (15.8).

ITERATIVE TRAINING Starting with randomly initialized parameters θ, we train the network f_θ for T epochs on the labeled examples X_L and labels $\mathbf{y}_L$ using the supervised cost (15.2). Using the learned embedding ϕ_θ, we then iterate the following process T' times:

1. Extract features $V := \phi_\theta(X)$ on the entire training set X and form the Laplacian $\mathcal{L}_\alpha$ (9.6) of the k-NN graph of V.

2. Perform LP by solving linear system (15.6) and infer pseudo-labels $\hat{\mathbf{y}}_U$ by (15.5) and weights by (15.7).
3. Train f_θ for one epoch on the training set X, labels $\mathbf{y}_L$ and pseudo-labels $\hat{\mathbf{y}}_U$ using the weighted cost J_w (15.8).

This iterative learning process is illustrated in Figure 15.2.

15.5 EXPERIMENTS

SETUP We evaluate our DLP against or combined with *Mean Teacher* (MT) [426], which applies (15.2) to the labeled examples and an unsupervised *consistency loss* between two models to the entire training set. The combination uses the sum of the two cost functions. We use *Canadian Institute for Advanced Research* (CIFAR)-10 [234], CIFAR-100 [234] and *mini*ImageNet [461], containing 10, 100 and 100 classes respectively; we draw 10, 3 and 3 splits respectively of the training set into labeled and unlabeled subsets. All training (test) sets contain 50k (10k) images. The labeled subsets are uniform over classes and vary between 1-20% of the training set.

As detailed in [193], we introduce a SSL setup for mini*ImageNet, which has been previously used for FSL [125, 362].*

We use the same 13-layer convolutional network that is used in prior work [241, 426]—referred to as C13—on CIFAR-10 and CIFAR-100, and Resnet-18 [165] on *mini*ImageNet. Pre-processing, data augmentation and training hyperparameters are detailed in [193]. We set $k = 50$ for the k-NN graph and $\alpha = 0.99$ in (9.6). We train for 180 epochs in total. We evaluate on the test set by *mean classification error* and *standard deviation* over the splits.

In both networks, f_θ consists of ϕ_θ followed by ℓ_2-normalization, an FC layer and softmax.

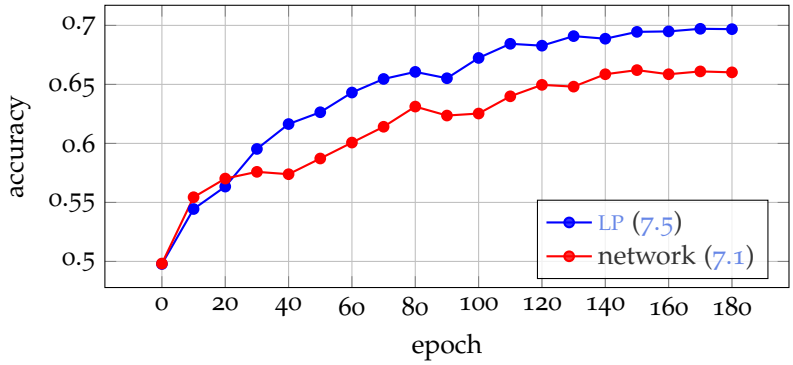
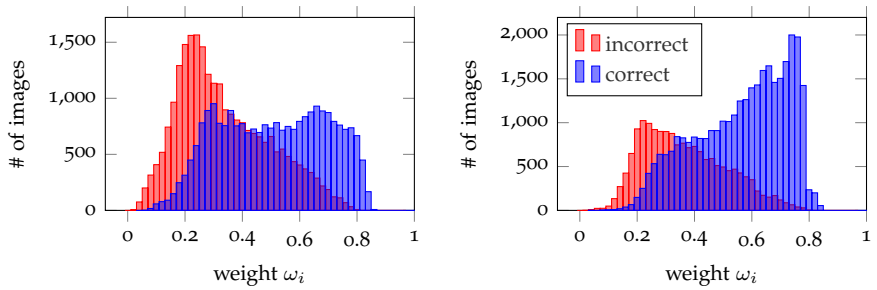


Figure 15.3: Accuracy of pseudo-labels predicted by LP (15.5) against network (15.1) using C13 on CIFAR-10 with 500 labels.

RESULTS Figure 15.3 shows the progress of the pseudo-label accuracy on unlabeled images X_U throughout the training process. LP predictions are consistently better than network predictions.

Figure 15.4 demonstrates how weights ω_i (15.7) accurately estimate the certainty of the predictions: While most examples are misclassified in the beginning, predictions become more accurate as training evolves. The proposed weighting mechanism is robust to incorrect pseudo-labels and prevents model collapse.



(a) epoch 0 (b) epoch 90

Figure 15.4: Distribution of weights ω_i (15.7) for unlabeled images at different epochs during training of C13 on CIFAR-10 with 500 labels.

DATASET	CIFAR-10		CIFAR-100		*mini*IMAGENET	
# LABELS	500	1,000	4,000	10,000	4,000	10,000
Supervised	49.08±0.83	40.03±1.11	55.43±0.11	40.67±0.49	53.07±0.68	38.28±0.38
DLP	32.40±1.80	22.02±0.88	46.20±0.76	38.43±1.88	**47.58±0.94**	36.14±2.19
MT [426]	27.45±2.64	19.04±0.51	45.36±0.49	36.08±0.51	49.35±0.22	32.51±1.31
MT [426] + DLP	**24.02±2.44**	**16.93±0.70**	**43.73±0.20**	**35.92±0.47**	50.52±0.39	**31.99±0.55**

Table 15.1: Error rate (mean±std over splits) of our DLP against or combined with MT [426] for different numbers of labels $|L|$ using C13 on CIFAR-10 and CIFAR-100 and ResNet-18 on *mini*ImageNet. Supervised baseline uses (15.2) only on the labeled examples X_L. The remaining methods are semi-supervised on the entire set X.

As shown in Table 15.1, either our DLP or the combination with MT [426] performs the best out on all datasets, demonstrating that the two approaches are complimentary. The gain over MT alone is more pronounced when the number of labels is small. DLP alone performs best on *mini*ImageNet with 4k labels.

15.6 DISCUSSION

Most recent approaches for deep SSL rely on training with unsupervised objectives on both labeled and unlabeled examples. Our approach relies on graph-based label propagation to infer pseudo-labels for the unlabeled examples. These pseudo-labels are shown to be more accurate than the ones inferred by the network itself. This idea is in principle complementary to unsupervised objectives, which is experimentally confirmed.

16 FEW-SHOT LEARNING

In Few-Shot Learning *(FSL), we target knowledge transfer from a training set with abundant data to sets with few available examples. In this work [261], we introduce two simple solutions: (i)* Dense Classification *(DC) over feature maps, which for the first time studies local activations in the domain of FSL, and (ii)* implanting*, that is, attaching new neurons to a previously trained network to learn new, task-specific features. Implanting enables training of multiple layers in the few-shot regime, departing from most related methods that train only the final layer.*

16.1 INTRODUCTION

Even if it is possible to learn with limited or no supervision, there are tasks and classes with even limited raw data, *i.e.* from the long-tail [473]. Deep neural networks pose several challenges in the low-data regime, in particular in terms of overfitting and generalization. The subject of FSL is to learn to *recognize previously unseen classes only with very few annotated examples*.

This is not a new problem [112], yet there is a recent interest in *meta-learning* [42, 116, 230, 385, 461]: Even when there is single large training set with a fixed set of classes, it is treated as a collection of sets of different classes, where each class has a few annotated examples. Here we argue instead that a simple pipeline using all available classes and data with a *parametric classifier* is equally effective.

This is inspired by early work on learning to learn *[175, 431].*

Most few-shot learning approaches do not deal explicitly with spatial information since feature maps are usually flattened or pooled before the classification layer. Instead, we show that performing *Dense Classification* (DC) over feature maps during representation learning consistently improves performance on novel tasks.

While *incremental learning* touches similar aspects with FSL by learning to adapt a network to new tasks [283, 284] or extending a network with new layers for each new task [378], few of these ideas have been adopted in FSL. We introduce neural *implants*, which are layers attached to an already trained network, enabling it to quickly adapt to new tasks with few examples.

16.2 PRELIMINARIES

PROBLEM We are given a *training set* $X := \{x_1, \ldots, x_n\} \subset \mathcal{X}$ where $\mathcal{X}$ is an *input space*, and corresponding labels $\mathbf{y} := (y_1, \ldots, y_n)$ with $y_i \in [c]$, where c is a set of *base classes*. On this training set we are allowed to learn a representation of the domain $\mathcal{X}$ such that we can solve new tasks. This representation learning we call *stage 1*.

We write $[n] = \{1, \ldots, n\}$ *for* $n \in \mathbb{N}$.

In a new *Few-Shot Learning* (FSL) task, we are given a set of few *support* examples $X' := \{x'_1, \ldots, x'_{n'}\} \subset \mathcal{X}$ and corresponding labels $\mathbf{y}' := (y'_1, \ldots, y'_{n'})$ with $y'_i \in [c']$, where c' is a number of *novel classes* and $n' \ll n$; with this new data, the goal is to learn a classifier that maps a new *query* example from $\mathcal{X}$ to a class label in $[c']$. The latter classifier learning we call *stage 2*.

Base and novel classes are disjoint.

Classification is called *c'-way*; if there is a fixed number k of support examples per novel class, it is called *k-shot*. FSL is typically evaluated on a large number of new tasks, with queries and support examples randomly sampled from $(X', \mathbf{y}')$.

Novel classifier learning does not exclude continuing the representation learning.

CLASSIFIER The *classifier network* $f_\theta : \mathcal{X} \to \mathbb{R}^c$ (resp. $\mathbb{R}^{c'}$), where θ are the network parameters, maps an input example $x \in \mathcal{X}$ to a vector $f_\theta(x) \in \mathbb{R}^c$ (resp. $\mathbb{R}^{c'}$) of base (resp. novel) class probabilities. The *prediction* for input $x \in \mathcal{X}$ is the class of maximum probability $\arg\max_{j \in [c]} f_\theta(x)_j$ (resp. $[c']$). As a by-product, we assume an *embedding network* $\phi_\theta : \mathcal{X} \to \mathbb{R}^{r \times d}$. Since we study the spatial properties of the input, the embedding is a tensor, where r is the spatial dimensions and d the feature dimension. It can still be a vector if $r = 1$.

$f_\theta(x)_j$ denotes the j-th element of vector $f_\theta(x)$.

For 2d images, the embedding is a 3d tensor in $\mathbb{R}^{w \times h \times d}$, where $r = w \times h$ is the spatial resolution.

16.3 BACKGROUND

PROTOTYPES In *Prototypical Networks* (PN) [415], one finds a single prototype per novel class and classifies a query to the nearest prototype. Given $X', \mathbf{y}'$ and an index set $S \subset [n']$, let $S_j := \{i \in S : y'_i = j\}$ index the support examples in S labeled in class $j \in [c']$. The *prototype* $\mathbf{c}_j \in \mathbb{R}^{r \times d}$ of class j is given by the average of those examples

$$\mathbf{c}_j = \frac{1}{|S_j|} \sum_{i \in S_j} \phi_\theta(x'_i) \tag{16.1}$$

for $j \in [c']$. If $C := (\mathbf{c}_1, \ldots, \mathbf{c}_{c'})$, the classifier is defined as

$$f_{\theta,C}(x) := \boldsymbol{\sigma}\left([s(\phi_\theta(x), \mathbf{c}_j)]_{j=1}^{c'}\right) \tag{16.2}$$

for $x \in \mathcal{X}$, where s is *cosine similarity* and $\boldsymbol{\sigma} : \mathbb{R}^m \to \mathbb{R}^m$ is the *softmax function* $\boldsymbol{\sigma}(\mathbf{a}) := (e^{a_1}, \ldots, e^{a_m}) \Big/ \sum_{j=1}^m e^{a_j}$ for $\mathbf{a} \in \mathbb{R}^m$, $m \in \mathbb{N}$.

For $n \in \mathbb{N}$, we write $[e(i)]_{i=1}^n := (e(1), \ldots, e(n))$ for expression $e(i)$ of variable $i \in [n]$.

$s(\mathbf{v}, \mathbf{v}') := \langle \mathbf{v}, \mathbf{v}' \rangle / (\|\mathbf{v}\| \, \|\mathbf{v}'\|)$ for $\mathbf{v}, \mathbf{v}' \in \mathbb{R}^{r \times d}$; $\langle \cdot, \cdot \rangle$ and $\|\cdot\|$ are Frobenius inner product and norm respectively.

In *stage 2*, the full index set $S := [n']$ is used and computing class prototypes (16.1) is the only learning to be done.

In *stage 1*, a number of fictitious tasks called *episodes* are generated by randomly sampling a number classes from $[c]$ and then a number of examples per class from X with their labels from $\mathbf{y}$. These collections are denoted as $X', \mathbf{y}'$ respectively, of length n'. The set $[n']$ is partitioned into a *support set* $S \subset [n']$ and a *query set* $Q := [n'] \setminus S$. The classifier is then trained by minimizing over θ the cost function

$$J_p(X', \mathbf{y}'; \theta) := \sum_{i \in Q} \ell(f_{\theta,C}(x'_i), y'_i) \tag{16.3}$$

These collections are supposed to be support examples and queries of novel classes; queries are now labeled and the goal is to classify them correctly.

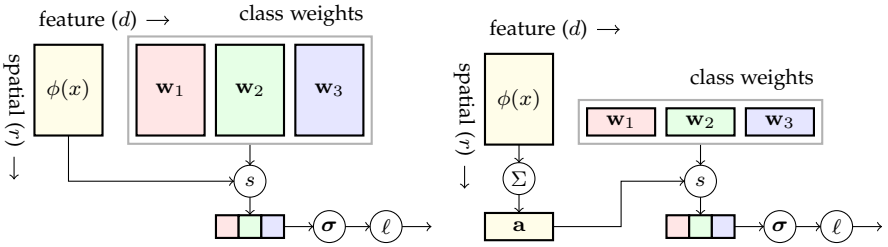
on the query set Q, where ℓ is the *cross-entropy* loss function given by $\ell(\mathbf{p}, j) := -\log p_j$ for $\mathbf{p} \in \mathbb{R}^m_+$, $j \in [m]$ and $m \in \mathbb{N}$.

COSINE CLASSIFIER This is a parametric classifier trained in a standard classification setting in *stage 1* [125, 347]. If $\mathbf{w}_j \in \mathbb{R}^{r\times d}$ is the weight parameter of class $j \in [c]$, the classifier is defined by

$$f_{\theta,W}(x) := \boldsymbol{\sigma}\left(\tau[s(\phi_\theta(x), \mathbf{w}_j)]_{j=1}^{c}\right) \tag{16.4}$$

for $x \in \mathcal{X}$, where $W := (\mathbf{w}_1, \ldots, \mathbf{w}_c)$ and $\tau > 0$ is a trainable *scale parameter*. Training involves minimizing over θ, W the cost function

$$J_c(X, Y; \theta, W) := \sum_{i=1}^{n} \ell(f_{\theta,W}(x_i), y_i). \tag{16.5}$$



(a) flattening (b) global pooling

Figure 16.1: An embedding $\phi(x)$ is compared to class weights $\mathbf{w}_j$ by similarity (s); softmax ($\boldsymbol{\sigma}$) and cross-entropy (ℓ) follow. (a) *Flattening* is equivalent to class weights having the same shape $r \times d$ as $\phi(x)$. (b) By *global pooling*, $\phi(x)$ is reduced (Σ) to vector $\mathbf{a} \in \mathbb{R}^d$ before being compared to class weights, which are in $\mathbb{R}^d$ too.

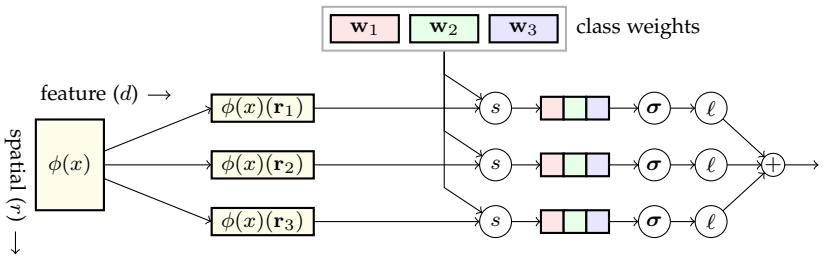


Figure 16.2: *Dense Classification* (DC). The embedding $A := \phi(x) \in \mathbb{R}^{r\times d}$ is seen as a collection of vectors $A(\mathbf{r}_1), \ldots, A(\mathbf{r}_3) \in \mathbb{R}^d$, each representing a region of x. Each vector is compared independently to the same class weights and the losses are added.

16.4 DENSE CLASSIFICATION

There are two common ways of handling an embedding tensor $\phi_\theta(x) \in \mathbb{R}^{r\times d}$. The first is an FC layer [125, 415, 461], which can be seen as *flattening* the tensor into a vector of length $r \times d$, as in Figure 16.1(a). This is discriminative, but not invariant. The second is *Global Average Pooling* (GAP) [125, 301, 328], reducing the tensor to a vector of length d, as in Figure 16.1(b). This is invariant, but less discriminative.

Alternatively, the weight parameter is a tensor of the same shape as the embedding.

For 2d images, $\Omega = [w] \times [h]$.

Given $A \in \mathbb{R}^{r \times d}$, $A(\mathbf{r}) \in \mathbb{R}^d$ collects elements of all channels of A at position $\mathbf{r} \in \Omega$.

This operation is 1×1 convolution followed by depth-wise softmax.

Given $A \in \mathbb{R}^{r \times c}$, $A^{(j)} \in \mathbb{R}^r$ collects elements of A at all positions in channel $j \in [c]$.

ARCHITECTURE *Dense Classification* (DC), a new approach, is shown in Figure 16.2. We view the embedding $\phi_\theta(x)$ as a collection of vectors $[\phi_\theta(x)(\mathbf{r})]_{\mathbf{r}\in\Omega}$, where Ω is the spatial domain and $\phi_\theta(x)(\mathbf{r}) \in \mathbb{R}^d$ represents a single spatial position $\mathbf{r} \in \Omega$.

In *stage 1*, we adopt a *cosine classifier* [125, 347], where weight parameters are vectors in $\mathbb{R}^d$, *shared* over all spatial positions, and the classifier $f_{\theta,W} : \mathcal{X} \to \mathbb{R}^{r\times c}$ maps an input to a tensor in $\mathbb{R}^{r\times c}$. If $\mathbf{w}_j \in \mathbb{R}^d$ is the weight parameter of class $j \in [c]$, (16.4) becomes

$$f_{\theta,W}(x) := \left[\boldsymbol{\sigma}\left(\tau[s(\phi_\theta(x)(\mathbf{r}), \mathbf{w}_j)]_{j=1}^c\right)\right]_{\mathbf{r}\in\Omega} \tag{16.6}$$

for $x \in \mathcal{X}$. Then, $f_{\theta,W}(x)(\mathbf{r}) \in \mathbb{R}^c$ is a vector of class probabilities at position $\mathbf{r} \in \Omega$, while $f_{\theta,W}(x)^{(j)} \in \mathbb{R}^r$ is the probability of class $j \in [c]$ as a function of position. This differs from *Class Activation Mapping* (CAM) [515] in that softmax suppresses all but the strongest responses at each position.

Figure 16.3: Examples overlaid with CAM [515] using ResNet-12 (*cf*. §16.7) trained with GAP or DC (16.6). Blue (red) is low (high) activation for ground truth. Top: Base classes (*walker hound*, *tile roof*). Bottom: Novel classes (*king crab*, *ant*).

TRAINING The cost function (16.5) becomes

$$J_d(X, \mathbf{y}; \theta, W) := \sum_{i=1}^{n} \sum_{\mathbf{r}\in\Omega} \ell(f_{\theta,W}(x_i)(\mathbf{r}), y_i). \tag{16.7}$$

DC differs from semantic segmentation *[273, 319] in that we use an image-level label.*

As shown in Figure 16.3, by encouraging the classifier to make correct predictions everywhere, DC results in smoother activation maps that are more aligned with objects. DC behaves like *implicit data augmentation* of dense shifts and crops with a single network evaluation.

16.5 IMPLANTING

From stage 1, we only keep the embedding network ϕ_θ. By *implanting*, we find new discriminative features for a new task in *stage 2*.

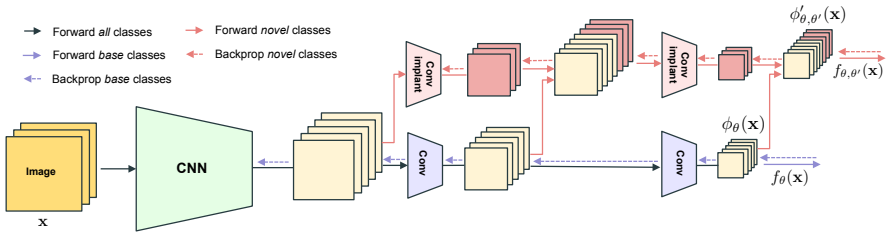


Figure 16.4: *Neural implants* are convolutional filters in a new processing stream parallel to the base network. The input of an implant is the depth-wise concatenation of activations from both streams. Parameters learned in stage 1 are *frozen* in stage 2.

ARCHITECTURE Beginning with the *base* embedding network ϕ_θ, we *widen* it by adding new convolution kernels, called *implants*, in a number of its top convolutional layers. As illustrated in Figure 16.4, we are creating a new stream of data in parallel to the base stream, learning additional connections for the new tasks.

With implanting, we reduce the risk or overfitting by adding a limited amount of new parameters, while keeping the previously trained ones frozen.

Let A_l (A'_l) be the output activation of base (implant) layer l. Then the input of an implant at layer $l+1$ is the depth-wise concatenation $[A_l, A'_l]$ if A'_l exists, and A_l otherwise. The set of all new parameters is $\theta' := (\theta'_{l_0}, \dots, \theta'_L)$, where θ'_l are the parameters of the l-th implant, l_0 is the first layer with an implant and L the network depth. The *widened* embedding network is $\phi_{\theta,\theta'}$.

TRAINING We use fictitious *subtasks* as in PN [415], however we are now working on novel classes in *stage* 2. In each subtask, we use each true support example alone as query and the rest as support: For each $i \in [n']$, we define a *query set* $Q_i := \{i\}$ and a *support set* $S_i := [n'] \setminus Q_i$. We compute prototypes C_i on index set S_i according to (16.1), replacing θ by (θ, θ'). The *widened* classifier f_{θ,θ',C_i} is similarly given by (16.2). Similarly to (16.3), we *freeze* the base parameters θ and learn the implants by minimizing over θ' the cost function

This process is deterministic. Because only one support example is missing, the true task is approximated well.

Prototypes are recomputed at each iteration based on the current version of implants.

$$J(X', \mathbf{y}'; \theta, \theta') := \sum_{i=1}^{n'} \ell(f_{\theta,\theta',C_i}(x'_i), y'_i). \tag{16.8}$$

16.6 INFERENCE

We perform GAP to the embeddings of the support examples and compute class prototypes by (16.1). Given a query $x \in \mathcal{X}$, we can perform GAP to its embedding $\phi_{\theta,\theta'}(x)$ too and classify it to the nearest prototype. Alternatively, we can *densely classify* the embedding $\phi_{\theta,\theta'}(x)$, by soft-assigning the embedding $\phi_{\theta,\theta'}(x)(\mathbf{r})$ of each spatial position $\mathbf{r}$ independently, averaging over all positions $\mathbf{r} \in \Omega$ according to

Inference is the same whether the embedding network has been implanted or not.

$$f_{\theta,\theta',C}(x) := \frac{1}{r} \sum_{\mathbf{r}\in\Omega} \boldsymbol{\sigma}\left(\tau[s(\phi_{\theta,\theta'}(x)(\mathbf{r}), \mathbf{c}_j)]_{j=1}^{c'}\right) \tag{16.9}$$

and classifying to $\arg\max_{j\in[c']} f_{\theta,\theta',C}(x)_j$.

16.7 EXPERIMENTS

miniImageNet is a subset of ImageNet ILSVRC-12 [377], containing 60k images of resolution 84×84 in 100 classes.

SETUP We use ResNet-12 [328] as an embedding network. We evaluate on the split [362] of *mini*ImageNet [461], consisting of 64, 16, 20 classes for training, validation and testing respectively. To generate the support set X' of a novel task, we randomly sample c' classes from the validation or test set and from each class we sample k images. We report the *average accuracy* and the corresponding 95% *confidence interval* over 5k (10k) such tasks for implanting (remaining) experiments, where each task contains 30 queries per class.

METHOD	1-SHOT	5-SHOT	10-SHOT
GAP	58.61±0.18	76.40±0.13	80.76±0.11
DC (ours)	**62.53±0.19**	78.95±0.13	82.66±0.11
DC + WIDE	61.73±0.19	78.25±0.14	82.03±0.12
DC + IMP (ours)	–	**79.77±0.19**	**83.83±0.16**
Gidaris *et al.* [125]	55.45±0.70	73.00±0.60	–
PN [328]	56.50±0.40	74.20±0.20	78.60±0.40
TADAM [328]	58.50±0.30	76.70±0.30	80.80±0.30

Table 16.1: *Average 5-way novel-class accuracy on* mini*ImageNet* using ResNet-12 except for [125]. GAP, DC and WIDE (last residual block widened by 16 channels): *Stage 1*. IMP (implanting): *Stage 2*, using GAP on both support and queries. At testing, we use GAP on support and GAP or DC on queries, depending on the choice of stage 1.

The most gain appears in 1-shot classification.

Our best results are at least 3% better in all settings.

RESULTS Table 16.1 shows that using DC rather than GAP in stage 1 improves novel tasks significantly. In stage 2, we add implants of 16 channels to all convolutional layers of the last residual block of the network, which helps further. This gain does not come from increased feature dimensionality, because just widening in stage 1 actually hurts performance. As shown in the bottom part, we outperform other methods by a large margin. Our baseline GAP is already competitive with TADAM [328], the previous state-of-the-art using the same network, while using a simpler cosine classifier in stage 1.

16.8 DISCUSSION

In this work we build upon a simplified process for learning on the base classes using a standard parametric classifier. We investigate activation maps for the first time in FSL and devise a new way of handling spatial information, improving the spatial distribution of the activation. We further adapt the network for new tasks by implanting neurons with limited new parameters. For the first time in FSL, we train several convolutional layers *to convergence*.

17 ADVERSARIAL EXAMPLES

In this work, we investigate the visual quality of the adversarial examples. We introduce smooth adversarial examples *[505], where the perturbation is locally smooth on flat areas of the input image, while it may still be noisy on textured areas and sharp across edges. This operation relies on Laplacian smoothing, which we integrate in the attack pipeline. Despite the additional constraint of smoothness, our attack has the same probability of success at lower distortion under a white-box scenario.*

17.1 INTRODUCTION

Adversarial examples result from applying an *imperceptible* perturbation to an image that can change a neural network's prediction [425]. Despite the progress in understanding the sensitivity of neural networks to their input, assessing imperceptibility remains elusive: User studies show that p-norms are largely unsuitable [399]. But human assessment of whether an image is adversarial is hard when the p-norm of the perturbation is small. We thus ask the following question:

> *Given a single image, can the effect of a perturbation be magnified to the extent that it becomes visible and a human may decide whether this example is benign or adversarial?*

This work has spawned research on adversarial attacks and defenses including team competitions [238].

Figure 17.1 shows that the answer is affirmative for a range of popular attacks. Assuming that natural images are piecewise smooth, we devise a simple *adversarial magnification* mechanism [505]. Without knowledge of the original image, it can reveal not only the presence of an adversarial perturbation but also *identify* the attack.

One can recognize the pattern of Fig. 4 of [304] in our Figure 17.1(g), revealing a Universal Adversarial Perturbation *(UAP).*

Motivated by this example, we argue that popular adversarial attacks have a fundamental limitation in terms of imperceptibility that we address by *smooth adversarial examples* [505]. More than just being "natural" [513] or smooth [146, 169], they are consistent with the smoothness pattern of the input image. They are photorealistic, low-distortion, and invisible even under magnification.

17.2 PRELIMINARIES

CLASSIFIER Let $\mathcal{X} := [0,1]^{n \times d}$ denote *images* of n pixels and d color channels. A *classifier* $f : \mathcal{X} \to \mathbb{R}^k$ maps an image $\mathbf{x} \in \mathcal{X}$ to a vector $f(\mathbf{x}) \in \mathbb{R}^c$ of class probabilities over c given classes. The *prediction* $\pi : \mathcal{X} \to [c]$ maps $\mathbf{x} \in \mathcal{X}$ to the class of maximum probability:

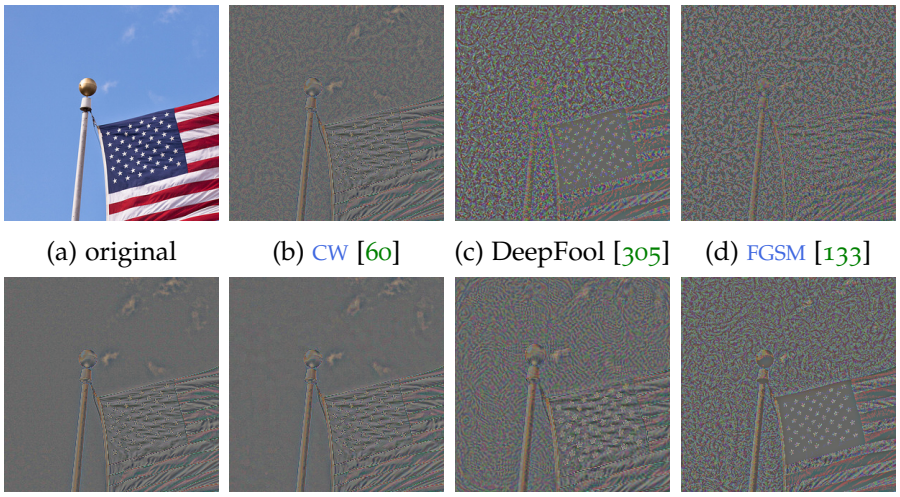
$$\pi(\mathbf{x}) := \arg\max_{k \in [c]} f(\mathbf{x})_k. \tag{17.1}$$

The parameters of the classifier are not shown: They remain fixed.

$[n] := \{1, \dots, n\}$ for $n \in \mathbb{N}$.

If a *true label* $t \in [c]$ is known, the prediction is *correct* if $\pi(\mathbf{x}) = t$.

$f(\mathbf{x})_k$ is the k-th element of vector $f(\mathbf{x})$.



(a) original (b) CW [60] (c) DeepFool [305] (d) FGSM [133]

(e) magnified (f) sCW (ours) (g) UAP [304] (h) I-FGSM [237]

Figure 17.1: (*Zoom in for better view.*) Magnified versions of adversarial examples generated by different attacks (b-d), (f-g) on input image (a), revealing the adversarial perturbation. Our *smooth adversarial example*, even when magnified (f), is indistinguishable from the magnified version of the original (e).

PROBLEM Let $\mathbf{x} \in \mathcal{X}$ be an image with true label t. An *adversarial example* $\mathbf{a} \in \mathcal{X}$ is an image such that the *distortion* $\|\mathbf{x} - \mathbf{a}\|$ and the probability $f(\mathbf{a})_t$ are small. This problem takes two forms:

We use the Frobenious norm to measure distortion (2-norm for $d = 1$).

1. *Target distortion, minimal probability*:

$$\min_{\mathbf{a} \in \mathcal{X}} f(\mathbf{a})_t \tag{17.2}$$

$$\text{subject to} \quad \|\mathbf{a} - \mathbf{x}\| \leq \epsilon, \tag{17.3}$$

where ϵ is a *distortion target*. The performance is measured by the *probability of success* $\mathbb{P}(\pi(\mathbf{a}) \neq t)$ as a function of ϵ.

These attacks are untargeted*; a* targeted *attack maximizes $f(\mathbf{a})_{t'}$ or requires $\pi(\mathbf{a}) = t'$ for a target label $t' \neq t$.*

2. *Target success, minimal distortion*:

$$\min_{\mathbf{a} \in \mathcal{X}} \|\mathbf{a} - \mathbf{x}\| \tag{17.4}$$

$$\text{subject to} \quad \pi(\mathbf{a}) \neq t. \tag{17.5}$$

The performance is measured by the *expected distortion* $\mathbb{E}(\|\mathbf{a} - \mathbf{x}\|)$. We focus on this form.

17.3 BACKGROUND: ATTACKS

We assume $d = 1$ in this section: Color channels are treated independently.

TARGET DISTORTION Given a distortion target ϵ, the *Fast Gradient Sign Method* (FGSM) [133] performs a single step in the direction of the (element-wise) sign of the loss gradient with ∞-norm ϵ,

$$\mathbf{a} := \text{proj}_{\mathcal{X}} \left(\mathbf{x} + \epsilon \operatorname{sign} \nabla_{\mathbf{x}} \ell(f(\mathbf{x}), t) \right), \tag{17.6}$$

where ℓ is the *cross-entropy* loss $\ell(\mathbf{p}, k) := -\log p_k$ for $\mathbf{p} \in \mathbb{R}^c$, $k \in [c]$. *Iterative FGSM* (I-FGSM) [237] initializes $\mathbf{a}_0 := \mathbf{x}$ and then iterates

$\text{proj}_X(\mathbf{a}) := \arg\min_{\mathbf{a}' \in X} \|\mathbf{a} - \mathbf{a}'\|$, *where X is closed and convex.*

$$\mathbf{a}^{(t+1)} := \operatorname{proj}_{\mathcal{X}\cap B_\infty[\mathbf{x};\epsilon]}\left(\mathbf{a}^{(t)} + \alpha \operatorname{sign} \nabla_{\mathbf{x}} \ell(f(\mathbf{a}^{(t)}), t)\right). \tag{17.7}$$

In both attacks, projection is element-wise by clipping.

$B_p[\mathbf{x};\epsilon]$ is the closed p-norm ball of radius ϵ, center $\mathbf{x}$.

TARGET SUCCESS Szegedy *et al*. [425] propose a Lagrangian formulation of (17.4)-(17.5), minimizing the cost function

$$J(\mathbf{a}, c) := \|\mathbf{x} - \mathbf{a}\|^2 + \lambda \ell(f(\mathbf{a}), t), \tag{17.8}$$

where λ is a Lagrange multiplier for (17.5). The attack of *Carlini & Wagner* (CW) [60] pertains to this approach. It uses the loss function

$$\ell_m(\mathbf{p}, k) := [\log p_k - \max_{j\neq k} \log p_j + m]_+ \tag{17.9}$$

$[x]_+ := \max(x, 0)$ is the positive part *of $x \in \mathbb{R}$.*

for $\mathbf{p} \in \mathbb{R}^c$, $k \in [c]$, encouraging logit $\log p_k$ to be less than any other $\log p_j$ for $j \neq k$ by at least margin $m \geq 0$. A change of variable eliminates the box constraint, replacing $\mathbf{a} \in \mathcal{X}$ by $\sigma(\mathbf{w})$, where $\mathbf{w} \in \mathbb{R}^{n\times d}$ and σ is the element-wise *sigmoid function*. The CW attack then uses the Adam optimizer [226] to minimize the cost function

ℓ_m is a hard version of the negative cross-entropy $-\ell$ for $m = 0$.

$\sigma(x) := 1/(1 + e^{-x})$ for $x \in \mathbb{R}$.

$$J(\mathbf{w}, \lambda) := \|\sigma(\mathbf{w}) - \mathbf{x}\|^2 + \lambda \ell_m(f(\sigma(\mathbf{w})), t). \tag{17.10}$$

over $\mathbf{w} \in \mathbb{R}^{n\times d}$. This is repeated for different λ by line search.

When the margin is reached, ℓ_m vanishes and the distortion term pulls $\sigma(\mathbf{w})$ back to $\mathbf{x}$.

17.4 GUIDED SMOOTHING

To make our perturbation invisible even under magnification, we smooth it *guided by* the input image. Guiding is similar to *guided filtering* [161, 340], but we use *graph filtering* [224, 517] for the filtering operation itself, as discussed in Chapter 9.

GRAPH We build a weighted undirected graph having the n pixels of the input image $\mathbf{x}$ as vertices. The i-th vertex is associated with *feature* $\mathbf{x}_i \in [0, 1]^d$, the i-th row of $\mathbf{x}$, and *position* $\mathbf{r}_i \in \Omega$, where $\Omega := [w] \times [h]$ is the spatial domain. Edge (i, j) has weight

w, h are the width and height of $\mathbf{x}$.

$$w_{ij} := \begin{cases} \kappa_f(\mathbf{x}_i, \mathbf{x}_j)\kappa_s(\mathbf{r}_i, \mathbf{r}_j), & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases} \tag{17.11}$$

for $i, j \in [n]$, where κ_f is a *feature kernel* and κ_s is a *spatial kernel*. The kernels are truncated such that the adjacency matrix W is sparse.

FILTERING Following Chapter 9, we normalize W into $\mathcal{W}$ by (9.3) and compute the regularized Laplacian $\mathcal{L}_\alpha$ by (9.6), where $\alpha \in [0, 1)$. Then, given a signal $\mathbf{y} \in \mathbb{R}^{n\times d}$, the smoothed version is $\mathcal{L}_\alpha^{-1}\mathbf{y}$ (9.9). To preserve the dynamic range of $\mathbf{y}$, we row-wise normalize by

Here, $\mathbf{1} \in \mathbb{R}^n$ is an all-ones vector.

$$s_\alpha(\mathbf{y}) := \operatorname{diag}(\mathcal{L}_\alpha^{-1}\mathbf{1}))^{-1}\mathcal{L}_\alpha^{-1}(\mathbf{y}). \tag{17.12}$$

The *smoothing function* s_α of course depends on $\mathbf{x}$. We say s_α is *smoothing guided* by $\mathbf{x}$ and the output is *smooth like* $\mathbf{x}$.

α controls the bandwidth: s_α is all-pass for $\alpha = 0$ and low-pass *as $\alpha \to 1$ (cf. Figure 9.1).*

In *interactive segmentation* [139, 224, 457] (resp. *semi-supervised classification* [517, 522]), $\mathbf{y}$ represents labels at few pixels (resp. examples) and is zero elsewhere. Here it is an arbitrary real-valued signal.

$\|\cdot\|_F$ is the Frobenius norm.

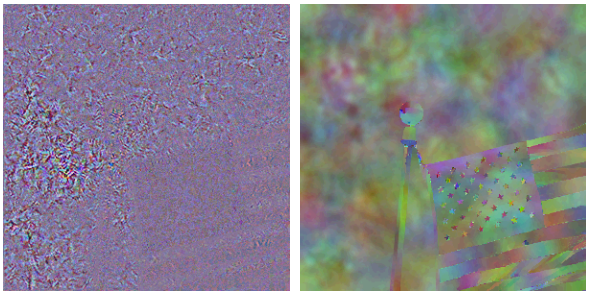
ENERGY MINIMIZATION Similarly to (9.17), $s_\alpha(\mathbf{y})$ is proportional to the minimizer, over $\mathbf{z} \in \mathbb{R}^{n\times d}$, of the quadratic function

$$Q_\alpha(\mathbf{z}, \mathbf{y}) := \frac{\alpha}{2} \sum_{i,j} w_{ij} \|\hat{\mathbf{z}}_i - \hat{\mathbf{z}}_j\|^2 + (1-\alpha) \|\mathbf{z} - \mathbf{y}\|_F^2 , \tag{17.13}$$

$\hat{\mathbf{z}}_i$ is encouraged to stay close to $\hat{\mathbf{z}}_j$ when w_{ij} is large.

where $\hat{\mathbf{z}} := D^{-1/2}\mathbf{z}$ and D is given by (9.2) [517]. The first pairwise *smoothness term* encourages $\mathbf{z}$ to be smooth wherever $\mathbf{x}$ is. The second unary *fitness term* encourages $\mathbf{z}$ to stay close to $\mathbf{y}$.

(a) original $\mathbf{x}$ (b) adversarial $\mathbf{a}$: CW (c) adversarial $\mathbf{a}$: sCW



(d) $\|\mathbf{z}\| = 3.64$ (e) $\|\mathbf{z}\| = 4.59$

Figure 17.2: Attacks against Inception-v3 on ImageNet. (d,e) Perturbation $\mathbf{z}$, scaled to $[0, 1]$ independently per channel. Despite its higher distortion, $\mathbf{z}$ is 'smooth like' $\mathbf{x}$ and totally invisible for sCW.

17.5 SMOOTH ADVERSARIAL EXAMPLES

Our key idea is that the perturbation $\mathbf{z} := \mathbf{a} - \mathbf{x}$ is *smooth like* the input image $\mathbf{x}$. This is achieved by a smoothing operation *guided by* $\mathbf{x}$, which we integrate into CW (17.10). Instead of representing $\mathbf{z}$ implicitly as a function $\sigma(\mathbf{w}) - \mathbf{x}$ of another parameter $\mathbf{w}$, we minimize over $\mathbf{z} \in \mathbb{R}^{n\times d}$ directly, while using the element-wise *clipping function* $\mathrm{proj}_{\mathcal{X}}(\mathbf{a}) = \min([\mathbf{a}]_+, 1)$ to satisfy the box constraint $\mathbf{a} \in \mathcal{X}$ (17.4):

This is because we need to process the perturbation $\mathbf{z}$ independently of $\mathbf{x}$.

$$\min_{\mathbf{z}} \ \lambda \|\mathbf{z}\|^2 + \ell_m(f(\mathrm{proj}_{\mathcal{X}}(\mathbf{x} + \mathbf{z})), t). \tag{17.14}$$

Optimizing (17.14) results in 'independent' updates at each pixel.

SMOOTHNESS PENALTY A straightforward choice would be to introduce a pairwise loss term $\mu \sum_{i,j} w_{ij} \|\hat{\mathbf{z}}_i - \hat{\mathbf{z}}_j\|^2$ into (17.14). A problem is that the spatial kernel is typically narrow, so it would take a lot of iterations to achieve smoothness globally, each requiring a forward and backward pass through the classifier network.

SMOOTHNESS CONSTRAINT Instead, we introduce the loss term $\mu Q_\alpha(\mathbf{z}, \mathbf{y})$ (17.13) where $\mathbf{y} \in \mathbb{R}^{n \times d}$ is unconstrained, while $\mathbf{z}$ should be close to $\mathbf{y}$ and smooth like $\mathbf{x}$ (17.13), as well as small (17.14). Then, by letting $\mu \to \infty$, this term becomes a hard constraint $\mathbf{z} = s_\alpha(\mathbf{y})$ (17.12), imposing a *globally* smooth solution at each iteration:

$$\min_{\mathbf{y}} \ \lambda \|s_\alpha(\mathbf{y})\|^2 + \ell_m(f(\text{proj}_{\mathcal{X}}(\mathbf{x} + s_\alpha(\mathbf{y}))), t). \tag{17.15}$$

During optimization, every iterate of perturbation $\mathbf{z}$ is indeed smooth like $\mathbf{x}$. We call this the *smooth Carlini & Wagner* (sCW) attack.

OPTIMIZATION Problem (17.15) has the same form as (17.14), where $\mathbf{z}$ has been replaced by $s_\alpha(\mathbf{y})$. This implies that we can use the same optimization method as the CW attack, where we initialize by $\mathbf{y} = \mathbf{0}_{n \times d}$ and we apply function s_α at each iteration.

We use CG [317] to solve the linear system $\mathcal{L}_\alpha \mathbf{z} = \mathbf{y}$ (9.7) for $\mathbf{z}$. Matrix $\mathcal{L}_\alpha$ is fixed during optimization, depending only on the input image $\mathbf{x}$. Gradients are easy to compute because smoothing is linear: In the backward pass, the gradient of objective (17.15) w.r.t. $\mathbf{y}$ is obtained from the gradient w.r.t. $\mathbf{z}$ (or $\mathbf{a}$) by smoothing, much like how $\mathbf{z}$ is obtained from $\mathbf{y}$ in the forward pass, $\mathbf{z} = s_\alpha(\mathbf{y})$ [505].

In the backward pass, we auto-differentiate through the forward CG iterations.

As shown in Figure 17.2, sCW produces smooth perturbations that are totally invisible in natural images. The reason is the 'phantom' of the original, revealed when the perturbation is isolated.

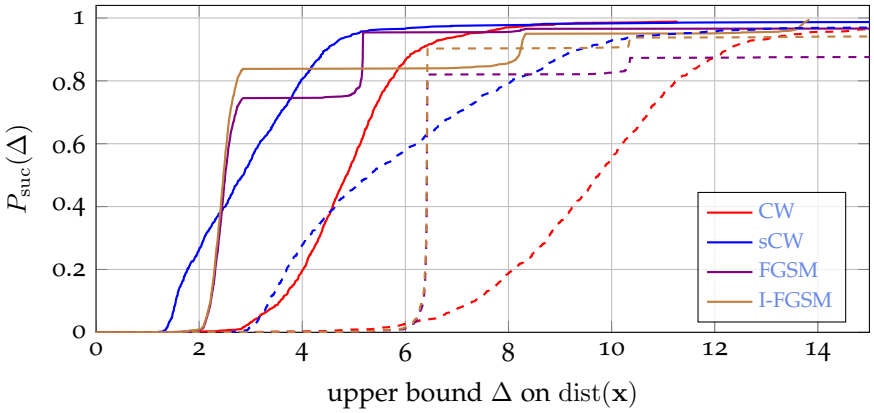


Figure 17.3: Operating characteristics against Inception-v3 (solid) and ResNet-v2-50 (dashed) on ImageNet [239].

17.6 EXPERIMENTS

SETUP We focus on a *white-box* setting, where the network is fully known to the attacker. We compare *target distrortion* ∞-norm attacks FGSM [133] and I-FGSM [237] as well as *target success* 2-norm attacks CW [60] and our smooth version sCW. We use Inception-v3 [424] and ResNet-v2-50 [166] networks on dataset [239], comprising 1,000 ImageNet [92] images. We use a Laplacian feature kernel and a 3×3

Attacks are untargetted.

neighborhood spatial kernel (17.11); we set $\alpha = 0.997$ and use 50 iterations of CG (17.12). Parameters of the attacks are detailed in [505].

EVALUATION PROTOCOL We introduce a new protocol that is more elaborate and more fair in comparing different types of attacks as discussed in Section 17.2. Given a test image set, we only consider its subset X of N images that are classified correctly without any attack. Let X_{suc} be the subset of X where the attack succeeds and $\text{dist}(\mathbf{x}) := \|\mathbf{a} - \mathbf{x}\|$ the distortion for image $\mathbf{x} \in X_{\text{suc}}$. The *operating characteristic* function $P_{\text{suc}} : \mathbb{R}_+ \to [0, 1]$ measures the *probability of success* as a function of a given upper bound Δ on distortion:

This function increases from 0 to $|X_{\text{suc}}|/N$ as Δ ranges from 0 to ∞.

$$P_{\text{suc}}(\Delta) := \frac{1}{N}|\{\mathbf{x} \in X_{\text{suc}} : \text{dist}(\mathbf{x}) \leq \Delta\}| \tag{17.16}$$

for $\Delta \in \mathbb{R}_+$. For *distortion targeting* attacks, we run an attack over the test set with different target distortions ϵ. The attack succeeds on image $\mathbf{x} \in X$ if it succeeds in at least one run. For $\mathbf{x} \in X_{\text{suc}}$, the distortion $\text{dist}(\mathbf{x})$ is the minimum distortion over all runs.

Resnet-v2 is more robust than Inception-v3: The operating characteristics are shifted to the right.

RESULTS As shown in Figure 17.2, our sCW improves a lot the original CW in terms of distortion, while keeping the probability of success roughly the same. This is surprising: We would expect a price to be paid for a better invisibility as the smoothing is adding an extra constraint on the perturbation.

As a result, the operating characteristic is piecewise constant in Figure 17.3.

CW internally optimizes its parameter λ independently per image, while for FGSM and I-FGSM we externally set a small set of target distortions ϵ on the dataset. Our new evaluation protocol is fair, given that CW is more expensive. As an interesting finding, *distortion-targeting attacks are more competitive than previously thought*.

17.7 DISCUSSION

Smoothing helps mask the adversarial perturbation, when it is 'like' the input image. It allows the attacker to delude the classifier thanks to larger distortions, while still being invisible. It is impressive how sCW improves upon CW in terms of distortion and imperceptibility at the same time, given that it is more constrained. The question raised in the introduction is still open: Figure 17.1(f) shows that a human cannot make the difference between the input image and its adversarial example even under magnification. This does not mean that an algorithm will not detect some statistical evidence.

Part IV

BEYOND

We summarize more of our past and present contributions, reflect on our contributions in the present context and consolidate the ideas exposed in this manuscript. We then put forth a vision for future research and attempt to draw a road map of ideas that are likely to come.

18 REFLECTION

In this chapter we first discuss briefly more of our relevant contributions spanning a period of 20 years as well as more recent and ongoing work that did not make it into the main part of this manuscript. We then revisit our contributions, discuss them in the present context, interpret the properties of different representations, make connections between different ideas, highlight limitations and hint to future work. Finally, we consolidate the ideas exposed in this manuscript by summarizing our contributions and structuring them into a road map, opening the way to new directions.

18.1 WHAT ELSE?

Only a few articles are selected to be exposed in the three technical parts of this manuscript. There are several others that fit perfectly the main storyline. We include here a synopsis of a wider selection of articles organized by subject and time, ranging over a period of 20 years. In the interest of space, there is very limited to no context or background in this case, even though there are problems we have not discussed so far.

VIDEO ABSTRACTION *1998-2000.* Our early research focuses on pooling regional descriptors into a *global image representation*. We combine image partitions obtained by criteria like color and motion statistics, giving rise to a *multidimensional histogram* [14]. Adding a *depth map* partition in the case of stereo sequences, we obtain a highly accurate and temporally consistent object support [104].

We first apply this global representation to automated *video abstraction* by analyzing the temporal evolution of video sequences in the representation space and detecting *extremal points* [14]. By introducing a *correlation measure* on sets of video frames, we then attack the same problem by means of *combinatorial optimization* [13].

We then apply the same representation to *video retrieval*, introducing a *relevance feedback* mechanism in 1999 [103]. In retrospect, this is an early *supervised metric learning* [491] approach, where positive and negative pairs are obtained by user feedback. The mapping is linear, giving rise to a closed-form solution. It is a *transductive* method, since the metric is learned on the test set. Relevance feedback is a standard motivating example of transductive inference [209].

SPATIOTEMPORAL SALIENCY *2005-2013.* By extending the model of Itti *et al*. [195] to video, we develop *spatiotemporal saliency* models[1] that employ competition across different feature channels, spatiotemporal lo-

1 http://image.ntua.gr/iva/research/visual_saliency

cations and scales. We apply these models to a wide range of problems, including *visual attention modeling* [360], *salient event detection* [359], *movie summarization* [108] and *video classification* [361].

By selecting points at local extrema of the saliency map, we extend to *spatiotemporal feature detection*[2], balancing well between sparsity and discriminative power. We apply to *human action recognition* [357], with spatiotemporal descriptors, a BoW model and a nearest neighbor classifier. We outperform by a large margin previous sparser *Space-Time Interest Point* (STIP) detectors [99, 391, 488].

2008. OBJECT PROPOSALS Alexe *et al*. [5] are known to introduce the first *class-agnostic* method to find candidate regions for object detection in 2010. Such a region is called *object proposal*, *region proposal* or *Region of Interest* (RoI). Initially, region proposals are found by *bottom-up* cues like hierarchical segmentation [450] or edges [524]. Their use becomes widespread also in CNN-based detectors [128] and they are subsequently *learned*, *e.g*. by a RPN [367].

What is less known is that we investigate this idea in 2008 [218] using hierarchical grouping of interest points, in particular Harris corners [156]. Interest points are considered as a baseline by Alexe *et al*. [5], but without grouping.

2010-2016. LOCAL FEATURE DETECTION Image gradient and edges are considered too unstable for local feature detection. Yet, using *computational geometry* constructions, repeatable local features arise. Starting with unstable, *single-scale edges*, we develop two such solutions using *Delaunay triangulation* on local maxima of the Euclidean distance transform[3] [358] and *weighted* α*-shapes*[4] [453, 454].

Alternatively, starting with single-scale image gradient, we compute the exact *weighted distance transform* and *weighted medial axis* and partition the image similarly to *watershed* segmentation. The resulting *Medial Feature Detector* (MFD)[5] [16] provides subpixel-accurate features capturing the Gestalt principle of *closure*.

Our detectors yield regions of arbitrary shape and scale that provide good *coverage* of the image at a fraction of the total number of features compared to other detectors. Most importantly, while most related work is limited to matching experiments, we additionally provide experimental evaluation on the end task of *instance-level search*, outperforming existing detectors with only modest requirements in terms of inverted index size and query time.

2010-2014. GEOMETRY INDEXING AND FEATURE SELECTION In instance-level search, *geometry* is traditionally only considered in a sequential process of *spatial verification* that is applied to a short list of top-ranking images [341, 433] according to similarities by global repre-

2 http://image.ntua.gr/iva/research/spatiotemporal_feature_detection
3 http://image.ntua.gr/iva/research/edge_based_feature_detection
4 http://image.ntua.gr/iva/research/wash/
5 http://image.ntua.gr/iva/research/medial_features/

sentations like BoW. Representations *incorporating geometry* are either not invariant [249] or limited to weak constraints [84, 203].

By exploiting the shape (affine) parameters of local features, our *Feature Map Hashing* (FMH)[6] [17] is the first to incorporate global feature geometry in the index, while enjoying invariance to affine transforms. While queries are very fast, the obtained representation is large. Thus, FMH originally scales only up to 50k images. One way to obtain a more compact representation is *feature selection* by matching multiple views of the same object or scene and focusing on the common parts. Consequently, FMH scales up to 1M images [435].

But, what happens to *isolated views*, having nothing to match with? Our *SymCity*[7] [434] selects features from isolated views by detecting *symmetries* and *repeating patterns*. The underlying assumption is that in urban scenes, man-made structures exhibit a significant amount of such structure, while 'transient' objects like vehicles and persons standing in front of buildings do not. Building on our HPM [433], detection is blazing fast.

CLUSTERING AND NEAREST NEIGHBOR SEARCH Inspired by the close relation between clustering and nearest neighbor search as well as the success of one helping to solve the other, we introduce *Dimensionality-Recursive Vector Quantization* (DRVQ) [12], a paradigm where both problems are solved simultaneously. Traditionally, in the assignment step of the k-means algorithm, one needs to search for the nearest centroid for each data point. In DRVQ, we rather start from centroids and construct a distance map over the entire space. Thus, search reduces to a lookup operation. Consequently, DRVQ is orders of magnitude faster than any other solution. *2013-2015.*

In previous work [12, 18], we cluster local descriptors to construct visual vocabularies. By contrast, powerful global CNN representations allow scaling up to *image clustering*. By quantizing them, we introduce *Inverted-Quantized k-Means* (IQM) [15], an extremely efficient *web-scale image clustering* method, subsuming the properties of EGM [18] (dynamic estimation of the number of centroids) and DRVQ [12] (inverted search from centroids to points). We achieve clustering of 100M images in less than an hour on a single machine.

LOCATION RECOGNITION A particular application of image retrieval is *visual location recognition*. We introduce a *panorama-to-panorama* matching process for location recognition from CNN representations of street-view images [188], reaching near-perfect performance on a challenging standard benchmark. *2017.*

DISCOVERY OF MID-LEVEL PARTS Learning *mid-level* discriminative parts for image classification is very common, even before deep learning. Our *automatic discovery of discriminative parts* [403] casts part *2017.*

6 http://image.ntua.gr/iva/research/feature_map_hashing
7 http://image.ntua.gr/iva/research/symcity/

learning as a *quadratic assignment* problem, allowing the use of a number of known relaxations and optimization algorithms. It is based on pre-trained networks for feature extraction and makes predictions by a linear classifier on a part-based encoding.

An *unsupervised* version of this work is applied both to *scene classification* and *instance-level search* [402]. By *unsupervised*, we mean that class labels are not used in part learning, while they are indeed used in learning the classifier on top of part-based encodings.

18.2 CURRENT WORK

There is also current unpublished work, carried out in 2019. This is even more relevant since it paves the way towards future ideas discussed in Chapter 19. We summarize this work here, accompanied with limited context and motivation.

ACTIVE LEARNING. We contribute an immediate extension of DLP [193] to deep active leaning [119, 394]. In *active learning* [396], one begins learning on limited labeled data, *selects* which examples to label next from a large unlabeled pool and iterates in multiple *cycles*. Using this pool for selection only is the opposite of what would normally work well when learning a deep model from scratch.

Using all available data at model training.

We depart from this setting by using both labeled and unlabeled data during model training [404]. We do so by using *unsupervised* feature learning at the beginning and *semi-supervised* learning at every cycle. We find that this brings a spectacular accuracy improvement compared to the differences between selection strategies. We also find that in the new setting, active learning is not effective when the quality of the representation is low, *i.e.* exactly when labeled data is limited. These findings suggest that we should revisit the standard settings and the evaluation protocol of deep active learning.

OBJECT DETECTION. Another extension of the semi-supervised paradigm is to *object detection*. Here, the most common setting advocating less supervision is *Weakly-Supervised Object Detection* (WSOD) [43], assuming no bounding box annotation but image-level labels on *all* data. There are *mixed settings* where *e.g.* few images come with bounding boxes and labels, and a large amount with image-level labels only [493]. These are sometimes called *semi-supervised*.

Few images with image-level labels and plenty completely unlabeled *images.*

We believe that the true analogue to semi-supervised learning in object detection is to have few clean images with image-level labels and a large amount of *completely unlabeled* images. We attack this more challenging problem for the first time by our *Nano-Supervised Object Detection* (NSOD) [495]. In particular, we first learn a *teacher classifier* by semi-supervised classification and then a *student detector*, training a WSOD model on pseudo-labels obtained by the teacher. By using more unlabeled images, we achieve performance competitive or superior to many state of the art WSOD solutions.

FEW-SHOT LEARNING. *Few-shot learning* is often motivated by the ability of humans to learn new tasks from few examples. However, standard benchmarks assume that the representation is learned on a limited amount of base class data, ignoring how much prior knowledge a human may have accumulated before learning new tasks. At the same time, even if a powerful representation is available, it may happen in some domain that base class data is limited.

Pre-trained representation on different domain, limited base-class data.

This motivates us to study a new setting: The representation is obtained from a classifier pre-trained on a large-scale dataset of a different domain, while the base class data is limited to few examples per class. Their role is to adapt the representation to the domain at hand rather than learn from scratch. We call this new setting *few-shot few-shot learning* [260].

Few clean labeled support examples and plenty novel-class data with noisy labels.

At the other extreme, it may happen in some domain where, apart from the few clean support examples, there exists a large amount of novel-class data, annotated with *noisy labels*. We attack this problem by forming a nearest neighbor graph over both clean and noisy data and training a *Graph Convolutional Network* (GCN) to predict class relevance of noisy examples. We then learn a classifier for the end task, weighting each noisy example by its relevance [187]. This is particularly effective in one-shot learning.

ADVERSARIAL EXAMPLES. When white-box attacks are successful, it is typically only the distortion that matters in their evaluation. We argue that *speed* is important as well, especially when considering that fast attacks are required by *adversarial training*. Given more time, iterative methods can always find better solutions.

Speed vs. *distortion.*

We investigate this speed-distortion trade-off and introduce a new attack called *Boundary Projection* (BP) [506] that improves upon existing methods by a large margin. Our key idea is that the classification boundary is a *manifold* in the image space: We thus quickly reach the boundary and then optimize distortion on this manifold.

Quantization.

We also study the effect of *quantization*, which is ignored in most related work by treating perturbations as real. This is important because the attacker's goal is to publish adversarial images (*e.g.* on the Internet), and publishing implies encoding in bytes.

18.3 SOME THOUGHTS

In the following, we revisit our contributions including some of those summarized in Sections 18.1 and 18.2 above, discuss them in the present context, interpret the properties of different representations, make connections between different ideas, highlight certain limitations or unclear situations and hint to potential future work. Some of the discussion is continued in Section 19.3.

REPRESENTATION QUALITY = EFFICIENCY The introduction of our work on *scene maps* in 2010 [19] begins by

> *"Images in community photo collections have scaled to billions [. . .] State of the art visual image retrieval has not yet scaled to permit searching into such huge collections."*

But, *can't we* search quickly in a collection of one billion items? Of course we can, if each item has a compact description. PQ-based solutions easily work at this scale since 2011 [206], including our LOPQ [213]. However, these results are on a dataset of SIFT descriptors and an image has *thousands* of such descriptors.

LOPQ has a precision of 47% with 128-bit codes and a query time of 53ms at this scale [213].

One can of course use aggregated representations like BoW [341], VLAD [204] or *Fisher vectors* [109, 339]. However, none works as well as having access to local descriptors [199, 202] or aggregating over large vocabularies, as in our ASMK [436, 437].

CNN-based representations [137, 352] work better even than ASMK with a *single* vector representation of length *e.g.* 512. This is a game changer. Indeed, searching over one billion images becomes routine [24] and the entire Flickr collection is indexed by LOPQ on deep descriptors in 2017[8]. Importantly, CNN-based representations are not constrained to instance-level similarity; they can be easily adapted to *any* similarity depending on the training task.

Of course, using powerful CNN representations *and* local features or descriptors improves performance further as shown in our benchmark [350] and DSM [405], but by increasing the cost. We need to always examine performance relative to resources taken, including the size of the representation and the query time.

ON POOLING So, *why* and *how* does it work? For one thing, *shallow* representations are just two layers deep, where the second layer involves learning a vocabulary on a training set; *deep* ones are based on learning highly nonlinear maps to optimize an objective like classification or ranking on a training set plus supervision.

More than that, the first layer of shallow representations is based on image patches, sparsely or densely sampled, and the second layer pools (aggregates) everything globally in *one step*, without taking into account the appearance or geometry of the neighboring patches. One exception is *Spatial Pyramid Matching* (SPM) [249], where pooling takes place at several levels of different spatial resolution, hence considering neighborhoods of different size. In both cases however, what is pooled is the quantized encodings over the vocabulary; the discriminative power of the original descriptors is lost. In SPM, pooling across all neighborhoods uses the *same* vocabulary.

One-step global pooling.

Gradual pooling over different neighborhoods.

By contrast, CNNs pool the representation *gradually* over several layers. At each layer, neighborhoods of different size are used. What is pooled is continuous activations over filter kernels rather than quantized encodings. Different kernels are used at each layer, hence at each neighborhood. As a result, a deep descriptor at a particular location

8 https://yahooresearch.tumblr.com/post/158115871236/introducing-similarity-search-at-flick

encodes *appearance* and *geometry* over a large *receptive field* and is very discriminative.

The same argument applies even in the absence of pooling, *e.g.* in the *all convolutional* network [417]. In this case, hand-crafted spatial pooling is replaced by strided convolution, allowing the network to learn its own spatial downsampling. In any case, there is always a final global spatial pooling in order to extract a global representation, and there is apparently less interference among different locations compared to shallow representations.

FOLLOW THE DATA By measuring similarity based on Euclidean distance, we make assumptions on the data distribution. In a high-dimensional space, such assumptions may not be realistic. The input space of a 1 megapixel color image has 3 million dimensions. Are all inputs equally likely? Certainly not: Natural images do not look like noise. This space is extremely sparsely populated and wildly inappropriate for any task other than capturing or displaying.

This number is 75-150 million in the case of the human retina.

Then, all about representation learning is *manifold learning* [37]: learning a nonlinear mapping to a lower-dimensional, more densely populated space, where dimensions act as *natural coordinates* in the input space. For instance, when the whole input shifts, few components should change in the representation space.

So, in the representation space, shallow or deep, is the distribution more well-behaved, like uniform, Gaussian, unimodal or clustered? Not really: 7

1. SIFT descriptors are a shallow representation of small image patches and our experiments with LOPQ [213] demonstrate that the statistics are different in each location in the descriptor space. In terms of *compression* efficiency, we need to adapt the encoder locally. Implicitly, LOPQ learns a nonlinear, non-smooth mapping before quantizing.

128 dimensions.

2. CNN descriptors are a deep representation of regions or entire images and our experiments with *diffusion* [194] demonstrate that in terms of similarity measurements, we need to adapt the metric locally. Implicitly, FSR [189] learns a nonlinear, smooth mapping.

E.g., *512 or 2,048 dimensions.*

So, can we improve the representation space to make it more well-behaved, following the data distribution? This is what we do in MoM [192] and DLP [193]: We represent the manifold structure of the data by a nearest neighbor graph, propagate similarities on the graph and use them to learn a new representation space, with or without supervision. This is done *iteratively* in DLP; although we have not tested, the same may apply to MoM[9].

Manifold learning.

Of course, doing this already assumes a compact representation as we discuss above. Using shallow representations, the closest work

Diffusion on shallow representations.

9 Recent results on unsupervised metric learning show that MoM fails when learning from scratch [58]. This might be fixed by iteratively updating the graph.

to our diffusion is *VisualRank* [208]. To measure similarity reliably, geometric verification is used, which makes similarity computation expensive. For this reason, truncation of the top 1000 items only takes place using text queries and then individual adjacency matrices are pre-computed offline per query, using hashing. By contrast, a single matrix over 1 million images is easily computed in our case [190] using CNN-based descriptors without any approximation.

It is clear that a representation of a *single* vector per image is even more important in quadratic-complexity tasks like graph-based methods [189, 190, 192–194] or clustering, *e.g.* by IQM [15], than in linear-complexity tasks like search. Of course, scaling up diffusion to billions is an open problem discussed in Chapter 19.

1000 WORDS *A picture is worth a thousand words*. What does that mean for us? Literally, with enough resolution, we can arrange 32×32 arbitrary small images of objects in a large one. The large image becomes a *container* and its descriptor is pooled over the descriptors of the small ones, causing interference. This is an extreme example, but looking for a small object among clutter is very common. We are back to estimating *partial* similarity. In particular, we can:

Partial similarity.

1. split into arbitrary regions and match independently, as in our *regional* diffusion [194];
2. *discover* the objects offline and match independently, as discussed below;
3. use spatial matching on local features, as in DSM [405]; or
4. increase the dimensionality to reduce interference [191, 504].

All solutions have increased cost, either offline or online, space or time. The point is, we cannot expect to fit any number of arbitrary objects into a small vector. The argument on gradual pooling into descriptors that encode appearance and geometry over the receptive field makes sense up to the level of objects, which have some common appearance and geometry. It does not apply to *clutter* that may appear in any possible layout.

Interestingly, in our work on *location recognition* [188], we average descriptors of images that are cropped out of a panoramic image, or pool over the activation tensor of the panoramic image directly. This works well but of course is a different situation.

DISCOVER, DISCOVER, DISCOVER Of the solutions presented above on estimating partial similarity, discovering the objects is appealing because it may be done offline and does not require representing a collection of objects or searching online. A lot of our work is based on this idea. Candidate object regions may be found in *isolated images*, but not very precisely:

Hypothesizing in isolated images.

1. Local features may be selected *e.g.* by detecting *symmetries* or *repeating patterns*, performing spatial matching of an image to itself, as we do in *SymCity* [434].
2. *Object proposals* may be found by bottom-up grouping of primitives, like local features or segments [218].
3. A *saliecy map* may be obtained by bottom-up operations [359] or directly from a CNN activation map as in CroW [214]; regions may be detected on a saliency map by our EGM [18], as we do in GOD [406, 407].

Discovering in image collections.

Objects may be discovered more precisely in *image collections*:

1. In video sequences, stereo or depth input, we can delineate objects very precisely, guided by *motion* or *depth* [104].
2. In collections containing multiple views of the same object or scene, we can perform spatial matching pairwise to find inliers and focus on the common parts, as we do in *feature selection* [435].
3. Alternatively, we can find common parts of several views and represent all of them jointly, as we do in *scene maps* [19].
4. Alternatively, we can match regions pairwise using CNN a representation. Optionally, we can find frequent objects according to *graph centrality* and weight objects accordingly, as we do in GOD [406, 407].

Exploiting data structure.

Clearly, in all cases where we use exhaustive pairwise matching, this is because we assume *unstructured* image collections. *Video sequences* simplify the problem, because only consecutive frames need to be matched and displacements are small. *Stereo* or *depth* input is even easier; accurate objects nearly pop-out in this case.

In *detect-to-retrieve* [427], a class-agnostic object detector is trained on a dataset annotated with bounding boxes. This is an efficient solution but, given a new dataset or domain, we cannot assume human annotation. GOD is unsupervised and can generate pseudo-annotation to train a detector instead.

It is also interesting to extend detection to *instance segmentation*. Given a collection comprising images or video depicting objects of unknown categories, optionally also a few images annotated with bounding boxes, a challenging goal would be to train an instance segmentation network to segment out all objects in the collection.

All-to-one.

ALL-TO-ONE OR ALL-TO-ALL? In *classification*, it is standard to have an FC layer with one prototype vector per class and a loss function that is softmax followed by cross-entropy. Each example is treated independently and the objective is that its representation vector is closer to the prototype of the correct class than any other. Thus, *all* examples are compared to *one* prototype per class.

All-to-all. By contrast, the objective in *metric learning* is that each example, treated as an anchor, is closer to positives than to negatives, optionally by some margin. The amount of positives and negatives varies according to hard example mining [158, 489]. However, it is becoming clear with global loss functions [56, 292, 322, 368, 449, 474] that, if we consider mining as part of the loss function, the actual objective is that an anchor is attracted to or repelled from each other example by a different weight depending on the label and the distance [474]. Thus, *all* examples are compared to *all* other examples.

All-to-many. There are intermediate situations where there are number of prototypes (centroids) per class, that is, each class is modeled by a *multimodal* distribution [233, 291, 308, 370, 487]. Thus, *all* examples are compared to *many* prototypes. The situation is the same in unsupervised methods that rely on pseudo-labels. In this sense, unsupervised representation learning can be seen as unsupervised metric learning in an *all-to-all* [490, 497] or *all-to-many* [58, 61] setting.

Class separation and margin. The argument for *all-to-all* metric learning is that by considering all pairs of examples, the class distributions become more separated. However, a similar effect can be achieved by applying a *margin* to the classification loss. In this sense, there are several *all-to-one* methods normalizing the class prototypes and computing *cosine similarity* with a *temperature* [125, 347, 355, 512], as well as methods adding different *margins* [93, 269, 467].

If *all-to-one* methods work well, then why do we need the more expensive *all-to-all* methods? Unfortunately, the situation is not clear because different methods are common in different tasks, *e.g.* fine-grained classification, face recognition, image retrieval or even few-shot learning. In any case, there is no guarantee of class separation, since classes at learning and inference are different.

18.4 CONSOLIDATION

It is time to consolidate the ideas and the discussion presented thus far. We provide a summary and open the way to new ideas discussed in Chapter 19.

SUMMARY We summarize our main contributions, connecting them and suggesting possible extensions. The contributions are ordered by coherence rather than following the separation of shallow *vs.* deep representations as in the technical parts of the manuscript.

1. Although developed as a *clustering* method for *vocabulary learning*, EGM [18] applies generally to fitting a mixture distribution and finding the number of components. We use it equally to fit regions to saliency maps in GOD. It could be used for *object detection*, *non-maximum suppression* or *mode seeking* problems, even in HPM. It is easy to differentiate and could be used anywhere in combination with EM, *e.g.* in *capsules* [170].

2. Although developed on shallow representations, ASMK [436, 437] is a generic method *adapting descriptors to vocabularies*. It still applies to deep representations [350]; in fact recent work extends ASMK in this context [427].

3. Assuming single-correspondence hypotheses, HPM [20, 433] is a faster and more flexible *spatial matching* method than RANSAC, allowing for multiple objects. In DSM we experiment with FSM [341] for spatial matching, but HPM could be used equally. Because it does not count inliers, HPM is also easy to differentiate, easier than RANSAC [50, 372].

4. DSM [405] is primarily a *local feature detector* and may be used with any matching method. We treat feature channels as visual words, but descriptors could be taken as well, from the same locations in the activation map. In retrospect, since local features are so well separated in *individual channels*, it is surprising how all other detectors map everything to a single channel first, where local features are not well separated any more[10].

5. LOPQ [213] compresses data for *nearest neighbor search*; it works equally well on shallow local and deep global descriptors [24]. It adapts locally the quantizers to the data distribution and since distributions in the representation space are unlikely to become uniform[11], LOPQ will probably remain relevant.

6. With only one or few vectors per image, *diffusion* [189, 194, 350] is becoming standard in manifold search [144, 350] and there are several extensions [65, 267]. It applies equally to *instance-level* and *category-level* tasks; we use the same model in MoM for unsupervised metric learning and DLP for semi-supervised learning. Extending to billions is a challenge.

7. MoM [192] is one of the first methods principally designed as *unsupervised deep metric learning* and there are now follow-up methods that are adapting supervised methods with pseudo-labels [497] or combining with unsupervised feature learning [58]. MoM is essentially training a *student* model having manifold similarity as a *teacher*. This idea could be easily combined with any other loss function. Extending to semi-supervised learning would be very interesting.

8. DLP [193] lies at the heart of the main theme of our work, with exploration of the manifold structure of the data improving the representation and vice versa. It is meant for *semi-supervised classification*, a *category-level* task, despite being based on diffusion, originally developed for *instance-level* search.

10 Concurrently, two more works [78, 106] detect local features independently per channel, but only as point features at global maxima, one per channel.

11 In fact, there are attempts to *make* the distribution uniform [380, 511].

9. Our work on *few-shot learning* [261] is the first to consider *activation maps*[12] for implicit data augmentation and train several layers to *convergence* on the few novel-class examples. We consider alternative settings using *more data* at pre-training [260] or at meta-training [187].

10. In *scene maps* [19], we explore an unstructured image collection, find common parts and collect these parts coming from several images into a *single representation*. In GOD [406, 407], we explore and find common parts similarly, then find repeating patterns and *weight* them according to centrality in each image. Naturally, we could combine the two ideas: representing images jointly *and* weighting by centrality. It would be interesting to extend to an *inductive* version to handle unseen images without having access to the collection.

11. Our *smooth adversarial examples* [505] are closer to the manifold of natural images than unconstrained adversarial examples. It would be interesting to use them in *adversarial training* to investigate if this improves generalization. For this to happen, it makes sense to combine with our fast attack BP [506] .

ROAD MAP We complete this chapter by building a road map of the ideas presented in this manuscript, summarizing everything in one phrase and opening the way to new ideas.

From representing to exploring.

On one hand, there is a path from representing individual items to exploring collections:

1. Improvements in visual representation and matching processes yield improved *search* by visual similarity, as measured quantitatively against human annotation. For instance, EGM [18] and ASMK [436, 437] improve the representation, while HPM [20, 433] and DSM [405] improve matching.

2. Improved search yields improved *discovery of global structure*. For instance, it yields improved nearest neighbor graphs, used by *diffusion* [189, 194, 350] and by GOD [406, 407] to discover objects. It also yields improved clustering, used by *scene maps* [19] to discover views of the same scene.

From exploring to representing.

On the other hand, there is an opposite path from exploring collections to representing individual items:

1. Improvements in the quality of global structure yield improved *search* or improved *representation* of individual images or groups of images. For instance, *diffusion* [189, 194, 350] explores data by *manifold similarity*; GOD [406, 407] represents images by focusing on objects and suppressing clutter; and a *scene map* [19] jointly represents aligned views of the same scene. However, this requires access to the target image collection.

12 Concurrently, two works consider attention mechanisms to focus on the foreground [481, 507].

2. Based on the global structure, deep parametric models can learn new representations and *generalize* to unseen images or entire collections. For instance, MoM [192] learns to respect the manifold structure of the data by *unsupervised metric learning* and DLP [193] similarly by *semi-supervised learning*. Ideas on *few-shot learning* [187, 260, 261] and *adversarial examples* [505, 506] are not as mature yet, but in the same direction.

We can summarize these observations in the following:

> *Exploring data and learning the representation are mutually beneficial.*

So, where does that lead next? Most of our work considers completely *unstructured* data collections, this is why we need to search for structure. On one hand, the situation is easier in *sequential* data like video, where each frame is automatically associated to the next, although we still need to search for recurring patterns. On the other hand, the situation is more difficult if we do not have access to all data at all times. We would need to *organize* the continuous input into stored representations.

Chapter 19 attempts to pave the way towards lifting the boundary between training and test sets, as well as the boundary between models and data.

19 OUTLOOK

This last chapter is an attempt to draw a road map of ideas that are likely to come and research directions that may be worth exploring. We motivate discussion by discussing the role of computing power and data in the development of deep learning and the relative shortage of storage capacity in current hardware and memory mechanisms in modern networks. We put forth a vision for future research towards mechanisms translating storage capacity to better performance. Finally, we present a number of research directions that can be summarized as a differentiable, incremental version of most ideas we have discussed in this manuscript.

19.1 MOTIVATION

Progressing hand in hand, *machine learning* and *computer vision* have made huge leaps towards interpreting the world around us, automating tasks and solving problems beyond human reach. There are two catalysts in this development: *computing power* and *data*.

Computing power.

Once a mechanism is in place to translate more computing power into better performance, the community instantly becomes enthusiastic in shifting from the pursuit of efficiency [244, 462] to exhaustive *parallel search* [68, 263], *deeper architectures* [165, 196] and large-scale *distributed training* [138, 502].

With the wide adoption of GPUs, the amount of *computing power* available to the average PhD student arguably grows by at least a factor of 100 between 2012 and 2017. The use of supercomputers for academic research becomes more and more common and the performance of the top system worldwide rises from 1.76 to 148 peta-*FLoating-point Operations Per Second* (FLOPS) between 2009 and 2019[1]. Yet, this is incomparable to the power of biological visual systems and is only expected to grow by further progress in hardware. A possible future is *analog electronics* [151, 180, 366, 397].

Visual data.

We often motivate a new method or task by arguing on human skills that we do not yet understand [145, 242, 499]. However, since the outburst of deep learning in 2012, the growth of *visual data* used in learning representations is less impressive than the growth of computing power, until 2017 [421]. Certainly, the average amount and quality of such data is also beyond comparison to the amount and quality of visual data that a grown-up human has seen.

Supervision.

One obstacle is the need for *supervision* by humans, which is very expensive especially when it includes localization as in object detection [240, 331] or dense annotation as in instance segmentation [147,

1 Compare *Jaguar* (https://www.top500.org/lists/2009/11/) with *Summit* (https://www.top500.org/lists/2019/11/).

240]; even more so in video [55, 81]. Fortunately, via *transfer learning* [100, 327, 501] and progress in *self-supervision* [61, 97, 126, 332, 470], *noisy supervision* [259, 421], *weak supervision* [43, 519] and *mixed supervision* settings [181, 193, 523], it is becoming increasingly clear that representation learning and potential human supervision on a given task can be essentially decoupled. This allows representation learning on larger scale datasets [62, 280, 492]. *Synthetic data* [348, 369] are helping in this direction too.

Memory mechanisms.

Still, models and data are treated as distinct entities. It is standard practice to discard the training set once a representation has been learned. This is what makes *incremental learning* [259, 365, 500] challenging. Neural networks can memorize the training set under certain conditions, but not when learning to generalize from large training sets [353, 379]. *Memory networks* are an explicit memory mechanism for sequences like text or video [312, 420]. But such mechanisms are far from becoming a standard component of the otherwise stateless networks used in vision tasks. By contrast, the human *visual long-term memory* has a massive capacity for details [51].

Explicit storage and retrieval.

In *image retrieval* [350], embeddings of a massive collection are indeed explicitly stored. However, this collection is again distinct from the training set, limiting the chances of optimizing the representation to the particular collection—*e.g.* GOD [407] learns attention on a fixed representation. The reason is that any update in the representation would invalidate the stored embeddings. The stored collection is seen as a test set, hence any optimization would be seen as *overfitting* [267]. While representation learning is end-to-end, storage still involves external post-processing like PCA and whitening. This is what makes retrieval on a dynamic collection challenging.

19.2 A VISION

This discussion leads us back to the questions raised in Chapter 1. What if there were no boundaries between training and testing? What if there was a continuous process of observing high-quality visual data, some times accompanied by supervision? What if we could not just learn tasks incrementally but memorize incrementally? What if we could recall instantly? What if we could learn the representation while memorizing?

A vision for future research that we put forth in this last chapter is to progress towards *making data a first-class citizen* in visual recognition tasks. This refers to data representations becoming explicit part of a model rather than just its training process.

> *Just as there are mechanisms to automatically translate more computing power to better performance, the same should happen with storage capacity.*

To make it more concrete: While the computing power may grow by a factor of 100 by using GPUs, the storage capacity drops by a

factor of 10 at the same time; what if it could grow by a factor of 1000 instead? Are there mechanisms to use it in improving performance in visual recognition tasks, just like "stacking more layers"?

Such mechanisms could lead for instance to an artificial *long-term visual memory* that learns what to store and what to forget for a given capacity. All operations would be online, continuously associating real-time input to stored data and keeping information organized at all times. Learning and inference would be one.

In light of this vision, Section 19.3 discusses a number of suggested research directions starting from the current state of the art. Just like the main goal of this manuscript, all directions are about *learning visual representations from data with limited supervision* and applying them to visual recognition tasks.

19.3 DIRECTIONS

This journey is certainly not over yet and there are too many ideas to be explored. Only a few are discussed here, in full conscience that ideas tend to be quickly invalidated or replaced by new in the ever changing landscape of machine learning and computer vision research. Some are natural extensions of the work discussed in this manuscript. Some rather lead to the formulation of new tasks.

RETHINKING METRIC LEARNING There are many tasks where *supervised metric learning* (*e.g.* with pairs or triplets) appears to have a similar objective with *supervised classification* (*e.g.* with variants of cross-entropy), but classes at inference are different from classes at learning. These include *e.g.* fine-grained classification, face recognition, person re-identification, local descriptor learning and instance retrieval, as discussed in Section 13.2. Few-shot learning also includes two training stages with different classes and is treated as either metric learning or classification.

The connection between supervised metric learning and supervised classification is often unclear. Their relative performance is also unclear as comparisons often do not involve more than one tasks. Increasingly complex loss functions involving tuples of examples [322] raise the problem of *sampling* from a seemingly endless choice of tuples [489]. *Ranking* loss functions on even larger tuples [56, 368] appear to suffer less from this problem, but still both sampling and the loss function rely on supervision.

Our MoM [192] is *unsupervised* and limits both positive and negative pairs by nearest neighbor search. The unsupervised setting is gaining momentum [490, 497]. Ideally, metric learning should be explored in all settings where classification has been explored: *e.g.* semi-supervised, few-shot and incremental learning, seen or unseen categories, as well as *distillation* [174, 354]. In particular, ranking loss functions [56, 368] would be most interesting to explore in all such settings. In the unsupervised setting, it would allow *e.g.* self-learning to rank.

Originally, contrastive loss [150] is applied in an unsupervised setting too.

Distillation would amount to training a student model to rank like a teacher model.

In classification, each example is processed independently, while in metric learning, the loss is a function of more than one examples, resulting in greater cost. It is necessary to study, under all settings, the relative performance of the two approaches as a function of the amount and distribution of available data (*e.g.* multimodal or not). A better understanding of the properties of the two approaches will allow a smoother progress towards more challenging problems like *long-tail* [217, 473] and *open-set* recognition [36, 271].

Localization tasks.

It is natural to extend the above studies to localization tasks including *spatial attention*, *object detection* and *instance segmentation*. Different supervision settings have not been explored as much as in classification. One interesting example is our NSOD [495], which is in a sense a true analogue of semi-supervised learning in object detection. Also, a *local version* of metric learning has not been explored much. For instance, pairs (or ranking) of image *regions* in this problem would play the same role as pairs (or ranking) of images in metric learning.

SPARSE ACTIVATIONS AND ALIGNMENT Among other tasks, convolutional networks are able to perform *correspondence* [73, 272] as discussed in Section 8.2. The applications range from *dense optical flow* in video [102] to *sparse correspondence* of object parts, even across semantic categories [1]. This has revived the interest in end-to-end trainable networks inspired from the conventional pipeline of feature detection, descriptor extraction and spatial matching, *e.g.* using RANSAC [50, 372].

However, candidate correspondences are often dense and exhaustive over four dimensions [372], which is very expensive, or we are back to hundreds of high-dimensional descriptors per image, losing the compactness of the representation [318]. Sparse local feature detection takes place on a single channel and requires a significant amount of engineering and supervision [94, 498].

By introducing DSM [405], we rather show that (i) *sparse local features* arise in individual channels of convolutional activations without explicit training, and (ii) feature channels behave like *visual words*, dispensing the need for descriptors or vocabularies. Essentially, a sparse approximation of the activation tensor is a compact representation that gives rise to efficient pairwise matching under *geometric alignment*. Such matching is beyond the standard matching of vectors obtained by global spatial pooling.

E.g. *by Euclidean distance, dot product or cosine similarity.*

In metric learning, images are considered in pairs, so alignment may become a natural part of the matching process. In classification (or detection), images (or regions) are matched against class prototypes—for instance, by an FC layer containing one prototype vector per class. Extending this matching to incorporate alignment is more challenging because prototypes involve averaging over classes. To allow deformation, matching should be flexible. To allow appearance variation, a *multimodal* class representation [291, 370] may be

necessary, having multiple prototype tensors per class, potentially distributed over different layers.

Why alignment?

Such extensions may open the door to end-to-end learning using geometrically aligned tensors in *category-level tasks*. Tensors are more discriminative than vectors obtained by spatial pooling, but they are not invariant. Explicit semantic alignment can answer the invariance *vs*. discriminative power dilemma. In semi-supervised settings, pairwise alignment can be used to transfer predictions from one image to another more reliably than global image similarity. This can improve graph-based methods like our DLP [193], even when training a standard classifier.

Alignment may also act as an *attention mechanism* to foreground objects. This is already useful in video, where objects may be identified through motion [332, 470] and in few-shot learning [154, 178], where there are not enough examples to learn the foreground implicitly. It can of course help in unsupervised object discovery, *e.g.* by initializing regions via pairwise image matching [70] rather than individually per image as in our GOD [406]. More interestingly, it may help in weakly supervised learning, where pixel-wise matching to a prototype vector notoriously focuses on the most discriminative object parts, failing to discover entire objects [177, 221].

HIGH-DIMENSIONAL CONVOLUTION The development of CNNs assumes that the input lies in a 2d Euclidean space. Handling higher-dimensional or non-Euclidean data is often treated by *Graph Convolutional Networks* (GCNs) [90, 227], where only pair-wise scalar affinity is used between input elements. This works well when only scalar information is available, but is not appropriate when the input originates in Euclidean space, *e.g.* 2d images, 3d video (where one dimension is time), or 3d shapes (point sets or surfaces).

E.g. *in the case of documents and citations.*

Early approaches to 3d data operate on multiple 2d views [10]. This can take advantage of networks pre-trained on visual data, but incurs information loss due to projection. PointNet [346] operates directly on 3d point clouds but does not allow for local interaction between points. The latter can be achieved by centering objects on the origin and expressing data and the convolutional operation in *spherical coordinates* [80, 107]. This also yields 3d rotation invariance but still involves projection on a sphere.

More principled approaches generalize 2d convolution to 3d, while maintaining the sparse representation [11, 344, 476]. Unlike GCN [90, 227], such generalizations *preserve coordinates* and use them to define local interactions via a kernel, just like standard convolution. However, they are still restricted to the same point set as the input. A 2d analogue of this idea is to represent 2d sketches by a binary image and restrict the output of convolution to the sketch [140]. This is against the principle of hierarchical representation of CNNs, *i.e.*, activations at certain locations in a layer give rise to activations at different locations in the next [118]. However, dense 3d activations [520] are too large to handle over a deep architecture.

Parametric convolution.

An interesting direction is to investigate a generalization of convolution whereby activations are sparse but localized onto *arbitrary locations*. This could be accomplished by using a mixture model for both activations and convolution kernels [11], but then fitting a new output mixture instead of restricting to the input point set. There is a connection of this approach to *capsules* [170] in that EM is used to fit a model of capsules in a layer to votes from the previous layer. By allowing an arbitrary number of points, the challenge is to dynamically control the number of output components. Our EGM [18] could be investigated in this direction.

Such sparse *parametric convolution* would be useful not just for high-dimensional input, but for 2d images too. For instance, it would allow a sparse representation of a *scale-space*, which is inherently 3d. More generally, it would allow a *transformation* or *pose* space where the size of the representation would depend on the input data only and not on the dimensionality of the space.

New supervision settings.

Clearly, work on high-dimensional data is not as mature as on 2d visual data. For instance, networks for 3d point sets are typically used for tasks like classification or semantic part segmentation under full supervision. A straightforward direction is then to explore new tasks and supervision settings like metric learning for similarity retrieval [167], semi-supervised, weakly supervised and few-shot learning. Of course, self-supervision will allow the use of large-scale unlabeled 3d data collections for pre-training. This is very important since labeled 3d data is not as abundant as 2d images.

MANIFOLDS, INDEXING AND GEOMETRY As discussed in Chapter 10, CNNs provide a powerful representation with only one or few vectors per image, allowing efficient *manifold similarity search*. We develop a spatial (or 'temporal') approach [194] based on a linear system solution, a spectral approach [189], where similarity is based on dot product in an embedding space, and a hybrid approach [190], controlling the space-time trade-off between these two extremes.

As discussed in Chapter 14, we also use manifold similarity for *unsupervised metric learning* [192], where the objective is that Euclidean similarity in the learned embedding space behaves like manifold similarity in the original feature space. Finally, as discussed in Chapter 15, we use manifold similarity for *semi-supervised classification* [193] and *active learning* [404], assigning pseudo-labels to unlabeled data according to manifold similarity to labeled data per class.

Billion-scale manifold similarity search.

However, all solutions are based on a nearest neighbor graph. How can manifold similarity search scale further up *e.g.* to *billions* of vectors? Currently, the only such mechanism is to *truncate* the adjacency matrix, essentially only re-ranking the elements of the corresponding subgraph with no hope of retrieving beyond that. Unfortunately, truncation still relies on Euclidean distance in the feature space. As discussed in Chapter 6, indexing schemes based on quantizers like our LOPQ [213] can scale up to billions of vectors in the Euclidean

space. It is interesting to investigate truncation based on quantizers in the *spectral embedding space*.

Geometry-driven manifold similarity.

Further improvement is possible by defining the adjacency matrix according to *spatial matching*, which is more reliable than matching global vector representations [65]. Still, using just a scalar gives little clue whether similarity is transitive, hence diffusion may suffer from *drift*. For instance, by walking along a street and looking sideways, diffusion will tell us that everything we see is similar.

A solution may be to generalize the operations. In the adjacency matrix, we replace scalar similarities by transformations found via spatial matching, like our DSM [405]. In matrix-vector multiplication, we replace scalar multiplication by composition of transformations and addition by mode seeking in the transformation space, like our HPM [20]. By doing so, we keep track of the transformation between the query and each image in the dataset, such that we can stop propagating when the object of interest goes out of sight.

This process bears similarities to iconoid shift *[483].*

According to Chapter 10, the diffusion process that we follow for manifold similarity search is linear and shallow. A nonlinear, deep version is a GCN [227]: Every propagation step becomes a layer and is accompanied by a linear mapping over a feature space and a nonlinearity. Of course, learning a GCN for manifold similarity search [267] is in fact overfitting the test set and can handle unseen queries but not unseen datasets. Nevertheless, this approach still makes sense in an *incremental* learning scenario, where the training set is memorized and the graph is dynamically updated.

Graph convolution *[227] and geometry.*

Another difficulty is that learning a GCN typically assumes that the entire dataset resides in memory such that propagation is performed over the entire graph at each layer. For back-propagation, all activations reside in memory as well. At large scale, the standard solution is *stochastic optimization* using mini-batches, which means storing all intermediate activations of the dataset. A challenging scenario is to compute the graph *dynamically* per layer [476], according to *spatial matching*. Then, activations become tensors and a compact representation as in our DSM [405] becomes indispensable.

LEARNING WHILE MEMORIZING The dominant paradigm in *category-level* tasks like classification and detection is that the training set is disposed of at inference, leaving all accumulated knowledge in the network parameters. The dominant paradigm in *instance-level* tasks like retrieval is that the test set is indexed against a fixed representation, because any network update would necessitate re-computing embeddings and re-indexing the entire test set.

Both paradigms may be challenged by a *memory* that is growing as we learn. In *category-level* tasks, a 'summary' of the training set *e.g.* a subset or a multimodal distribution per class can be made accessible while learning a new task. In *instance-level* tasks, training and test sets become part of a continuously growing knowledge, while the representation is updated as more data is stored.

Summarizing the training set of the *current task* by means of a subset or a multimodal parametric distribution in the input or feature space is common *e.g.* in metric learning [233, 308, 370], few-shot learning [508] and adversarial defenses [57, 411]. The same is common for the training set of *previous tasks e.g.* in incremental learning [35, 63, 365] and few-shot learning [3, 347]. In the latter case, the data is represented in the input space in order to update the network. Of course, this approach does not scale well.

In *metric learning* [56, 322] and *retrieval* [136, 351] tasks, it is standard to memorize the entire test set explicitly in the feature space. Following a common paradigm of few-shot learning [347], the embeddings of the test set may be thought of as an extremely wide FC layer. To our knowledge, storing the training set and incremental learning have not been explored in this context.

Storing data.

How should the data be represented? If it is meant to use the data in updating the network, then we should choose some layer before any layers to be updated (*e.g.*, *input images* if the entire network is to be updated); spatial information needs to be preserved and averaging is difficult. If it is meant to search by similarity to new examples, then a representation should be chosen among the *deepest layers*; spatial pooling into global vectors and averaging over the data is possible, but any network updates invalidate the data.

It would be interesting to investigate a compromise between these two extremes, *i.e.* use the representation of some *intermediate layer*, allowing both search by similarity and updates of all subsequent layers. Sparse activation tensors can keep the representation compact and averaging over similar examples can be an option, assuming alignment. Alternatively, *invertible* architectures [33, 131] would allow reconstruction from deep features.

Recalling data.

How should stored data be used while learning? In incremental classification, it is common to apply a *distillation loss* [63, 365] to preserve predictions on previous tasks. Alternatively, it is possible to constrain the network, *e.g.* with *synaptic plasticity* mechanisms [6, 228, 293] to prevent updates of connections that are important for previous tasks, or *network expansion* mechanisms [114, 378, 500] that guarantee predictions on previous tasks. The more explicit the mechanism, the larger the required architecture.

These ideas date back to Hebb [168].

An interesting direction to investigate is to apply such mechanisms to *metric learning* and *instance-level* tasks. The challenge is the sheer volume of stored embeddings that need to be preserved. A possible approach is to focus, for each new example, to similar data previously stored. By doing so, we maintain a summary of stored representations. All previous ideas apply in this case, including search by manifold similarity, sparse activations and geometry.

BIBLIOGRAPHY

[1] Kfir Aberman, Jing Liao, Mingyi Shi, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. "Neural best-buddies: Sparse cross-domain correspondence." In: *ACM Transactions on Graphics* 37.4 (2018), p. 69 (cit. p. 146).

[2] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. "A Learning Algorithm for Boltzmann Machines." In: *Cognitive Science* 9.1 (1985), pp. 147–169 (cit. p. 98).

[3] Arman Afrasiyabi, Jean-François Lalonde, and Christian Gagné. "Associative Alignment for Few-shot Image Classification." In: *arXiv preprint arXiv:1912.05094* (2019) (cit. p. 150).

[4] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. "Building Rome in a Day." In: *ICCV*. 2009 (cit. p. 19).

[5] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. "What Is an Object?" In: *CVPR*. 2010 (cit. pp. 62, 130).

[6] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. "Memory aware synapses: Learning what (not) to forget." In: *ECCV*. 2018 (cit. p. 150).

[7] Jon Almazan, Bojana Gajic, Naila Murray, and Diane Larlus. "Re-ID done right: towards good practices for person re-identification." In: *arXiv preprint arXiv:1801.05339* (2018) (cit. pp. 82, 96, 98, 103).

[8] R. Arandjelovic and A. Zisserman. "Three things everyone should know to improve object retrieval." In: *CVPR*. June 2012 (cit. p. 37).

[9] R. Arandjelović and A. Zisserman. "All about VLAD." In: *CVPR*. 2013 (cit. p. 36).

[10] Yoshifumi Nishida Asako Kanezaki Yasuyuki Matsushita. "RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints." In: *CVPR*. 2018 (cit. p. 147).

[11] Matan Atzmon, Haggai Maron, and Yaron Lipman. "Point convolutional neural networks by extension operators." In: *ACM Trans. Graph.* 37.4 (2018), 71:1–71:12 (cit. pp. 147, 148).

[12] Y. Avrithis. "Quantize and Conquer: A dimensionality-recursive solution to clustering, vector quantization, and image retrieval." In: *ICCV*. 2013 (cit. p. 131).

[13] Y. Avrithis, A. Doulamis, N. Doulamis, and S. Kollias. "A Stochastic Framework for Optimal Key Frame Extraction from MPEG Video Databases." In: *CVIU* 75.1–2 (July 1999), pp. 3–24 (cit. p. 129).

[14] Y. Avrithis, N. Doulamis, A. Doulamis, and S. Kollias. "Efficient Content Representation in MPEG Video Databases." In: *CVPR Workshops*. 1998 (cit. p. 129).

[15] Y. Avrithis, Y. Kalantidis, E. Anagnostopoulos, and I. Z. Emiris. "Web-scale image clustering revisited." In: *ICCV*. 2015 (cit. pp. 131, 136).

[16] Y. Avrithis and K. Rapantzikos. "The Medial Feature Detector: Stable Regions from Image Boundaries." In: *ICCV*. 2011 (cit. p. 130).

[17] Y. Avrithis, G. Tolias, and Y. Kalantidis. "Feature Map Hashing: Sub-linear Indexing of Appearance and Global Geometry." In: *Proceedings of ACM Multimedia*. 2010 (cit. p. 131).

[18] Yannis Avrithis and Yannis Kalantidis. "Approximate Gaussian mixtures for large scale vocabularies." In: *ECCV*. 2012 (cit. pp. vii, 4, 13, 16, 20–25, 63, 88, 131, 137, 138, 140, 148).

[19] Yannis Avrithis, Yannis Kalantidis, Giorgos Tolias, and Evaggelos Spyrou. "Retrieving Landmark and Non-Landmark Images from Community Photo Collections." In: *ACM Multimedia*. 2010 (cit. pp. vii, 4, 14, 19, 20, 45, 50, 51, 133, 137, 140).

[20] Yannis Avrithis and Giorgos Tolias. "Hough Pyramid Matching: Speeded-Up Geometry Re-Ranking for Large Scale Image Retrieval." In: *IJCV* 107.1 (2014), pp. 1–19 (cit. pp. vii, 4, 13, 17, 20, 27, 32, 139, 140, 149).

[21] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. "From generic to specific deep representations for visual recognition." In: *arXiv preprint arXiv:-1406.5774* (2014) (cit. pp. 58, 59, 90).

[22] Artem Babenko and Victor Lempitsky. "The Inverted Multi-Index." In: *CVPR*. 2012 (cit. pp. 39, 41, 43, 44).

[23] Artem Babenko and Victor Lempitsky. "Aggregating Deep Convolutional Features for Image Retrieval." In: *ICCV*. 2015 (cit. pp. 75, 81, 87).

[24] Artem Babenko and Victor Lempitsky. "Efficient Indexing of Billion-Scale Datasets of Deep Descriptors." In: *CVPR*. June 2016 (cit. pp. 134, 139).

[25] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. "Neural Codes for Image Retrieval." In: *ECCV*. 2014 (cit. pp. 55, 58, 59, 87, 103).

[26] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern information retrieval*. ACM press New York, 1999 (cit. p. 16).

[27] Pierre Baldi and Yves Chauvin. "Neural Networks for Fingerprint Recognition." In: *Neural Computation* 5.3 (1993), pp. 402–418 (cit. p. 98).

[28] D.H Ballard. "Generalizing the Hough Transform to Detect Arbitrary shapes." In: *Pattern Recognition* (1981) (cit. pp. 13, 17).

[29] Evgeniy Bart and Shimon Ullman. "Cross-Generalization: Learning Novel Classes From a Single Example By Feature Replacement." In: *CVPR*. 2005 (cit. p. 100).

[30] Miguel A Bautista, Artsiom Sanakoyeu, and Björn Ommer. "Deep Unsupervised Similarity Learning using Partially Ordered Sets." In: *arXiv*. 2017 (cit. p. 108).

[31] H. Bay, T. Tuytelaars, and L. Van Gool. "SURF: Speeded Up Robust Features." In: *ECCV*. 2006 (cit. pp. 14, 25, 31, 50).

[32] Atılım Günes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. "Automatic differentiation in machine learning: a survey." In: *JMLR* 18.1 (2017) (cit. p. 56).

[33] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Joern-Henrik Jacobsen. "Invertible Residual Networks." In: *ICML*. 2019 (cit. p. 150).

[34] Mikhail Belkin and Partha Niyogi. "Laplacian eigenmaps for dimensionality reduction and data representation." In: *Neural computation* 15.6 (2003) (cit. pp. 97, 99, 103).

[35] Eden Belouadah and Adrian Popescu. "IL2M: Class Incremental Learning With Dual Memory." In: *ICCV*. 2019 (cit. p. 150).

[36] Abhijit Bendale and Terrance E Boult. "Towards open set deep networks." In: *CVPR*. 2016 (cit. p. 146).

[37] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation Learning: A Review and New Perspectives." In: *PAMI* 35.8 (2013), pp. 1798–1828 (cit. p. 135).

[38] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. "Greedy Layer-Wise Training of Deep Networks." In: *NIPS* (2007) (cit. pp. 57, 95).

[39] Yoshua Bengio, Jean-Françcois Paiement, Pascal Vincent, Olivier Delalleau, Nicolas L Roux, and Marie Ouimet. "Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering." In: *NIPS*. 2004 (cit. p. 98).

[40] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." In: *IEEE Trans. Neural Networks* 5.2 (1994), pp. 157–166 (cit. p. 57).

[41] Jean Berko. "The Child's Learning of English Morphology." In: *Word* 14.2-3 (1958), pp. 150–177 (cit. p. 100).

[42] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. "Learning feed-forward one-shot learners." In: *NIPS*. 2016 (cit. p. 115).

[43] Hakan Bilen and Andrea Vedaldi. "Weakly supervised deep detection networks." In: *CVPR*. 2016 (cit. pp. 96, 132, 144).

[44] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006 (cit. pp. 22, 57).

[45] Liefeng Bo and Cristian Sminchisescu. "Efficient Match Kernel between Sets of Features for Visual Recognition." In: *NIPS*. 2009 (cit. p. 34).

[46] Oren Boiman, Eli Shechtman, and Michal Irani. "In Defense of Nearest-Neighbor Based Image Classification." In: *CVPR*. 2008 (cit. p. 17).

[47] Paolo Boldi, Violetta Lonati, Massimo Santini, and Sebastiano Vigna. "Graph Fibrations, Graph Isomorphism, and PageRank." In: *RAIRO-Theoretical Informatics and Applications* 40.2 (2006), pp. 227–253 (cit. p. 68).

[48] Y.L. Boureau, F. Bach, Y. Lecun, and J. Ponce. "Learning Mid-Level Features for Recognition." In: *CVPR*. 2010 (cit. p. 37).

[49] Anthony Bourrier, Florent Perronnin, Rémi Gribonval, Patrick Pérez, and Hervé Jégou. "Explicit Embeddings for Nearest Neighbor Search with Mercer Kernels." In: *Journal of Mathematical Imaging and Vision* 52.3 (2015), pp. 459–468 (cit. p. 60).

[50] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. "DSAC-differentiable RANSAC for camera localization." In: *CVPR*. Vol. 3. 2017 (cit. pp. 139, 146).

[51] Timothy F. Brady, Talia Konkle, George A. Alvarez, and Aude Oliva. "Visual long-term memory has a massive storage capacity for object details." In: *Proceedings of the National Academy of Sciences* 105.38 (2008), pp. 14325–14329 (cit. p. 144).

[52] Jane Bromley, Isabelle Guyon, Yann Lecun, Eduard Säckinger, and Roopak Shah. "Signature Verification Using a "Siamese" Time Delay Neural Network." In: *NIPS*. 1994 (cit. p. 98).

[53] Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. "Automatic query expansion using SMART: TREC 3." In: *NIST special publication* (1995), pp. 69–80 (cit. p. 60).

[54] Rudy R Bunel, Alban Desmaison, Pawan K Mudigonda, Pushmeet Kohli, and Philip Torr. "Adaptive neural compilation." In: *NIPS*. 2016 (cit. p. 96).

[55] Sergi Caelles, Alberto Montes, Kevis-Kokitsi Maninis, Yuhua Chen, Luc Van Gool, Federico Perazzi, and Jordi Pont-Tuset. "The 2018 DAVIS challenge on video object segmentation." In: *arXiv preprint arXiv:1803.00557* (2018) (cit. p. 144).

[56] Fatih Cakir, Kun He, Xide Xia, Brian Kulis, and Stan Sclaroff. "Deep Metric Learning to Rank." In: *CVPR*. 2019 (cit. pp. 138, 145, 150).

[57] R Caldelli, R Becarelli, F Carrara, F Falchi, and G Amato. "Exploiting CNN Layer Activations to Improve Adversarial Image Classification." In: *ICIP*. 2019 (cit. p. 150).

[58] Xuefei Cao, Bor-Chun Chen, and Ser-Nam Lim. "Unsupervised Deep Metric Learning via Auxiliary Rotation Loss." In: *arXiv preprint arXiv:1911.07072* (2019) (cit. pp. 135, 138, 139).

[59] S Carey. "Acquiring a single new word." In: *Papers and Reports on Child Language Development* 15 (1978), pp. 17–29 (cit. p. 100).

[60] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks." In: *IEEE Symp. on Security and Privacy*. 2017 (cit. pp. 122, 123, 125).

[61] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. "Deep clustering for unsupervised learning of visual features." In: *ECCV* (2018) (cit. pp. 96, 109, 138, 144).

[62] Mathilde Caron, Piotr Bojanowski, Julien Mairal, and Armand Joulin. "Unsupervised Pre-Training of Image Features on Non-Curated Data." In: *ICCV*. 2019 (cit. p. 144).

[63] Francisco M. Castro, Manuel J. Marin-Jimenez, Nicolas Guil, Cordelia Schmid, and Karteek Alahari. "End-to-End Incremental Learning." In: *ECCV*. 2018 (cit. p. 150).

[64] Siddhartha Chandra and Iasonas Kokkinos. "Fast, Exact and Multi-Scale Inference for Semantic Image Segmentation with Deep Gaussian CRFs." In: *ECCV*. 2016, pp. 402–418 (cit. p. 111).

[65] Cheng Chang, Guangwei Yu, Chundi Liu, and Maksims Volkovs. "Explore-Exploit Graph Traversal for Image Retrieval." In: *CVPR*. 2019 (cit. pp. 139, 149).

[66] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, 2006 (cit. p. 99).

[67] Kumar Chellapilla, Sidd Puri, and Patrice Simard. "High performance convolutional neural networks for document processing." In: *International Workshop on Frontiers in Handwriting Recognition*. 2006 (cit. pp. 55, 57).

[68] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. "TensorMask: A foundation for dense object segmentation." In: *arXiv preprint arXiv:1903.12174* (2019) (cit. p. 143).

[69] Y. Cheng. "Mean Shift, Mode Seeking, and Clustering." In: *PAMI* 17.8 (1995), pp. 790–799 (cit. p. 46).

[70] Minsu Cho, Suha Kwak, Cordelia Schmid, and Jean Ponce. "Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals." In: *CVPR*. 2015 (cit. pp. 63, 147).

[71] Minsu Cho and Kyoung Mu Lee. "Mode-Seeking on Graphs via Random Walks." In: *CVPR*. 2012 (cit. p. 105).

[72] S. Chopra, R. Hadsell, and Y. LeCun. "Learning a Similarity Metric Discriminatively, with Application to Face Verification." In: *CVPR*. June 2005, pp. 539–546 (cit. pp. 59, 98, 103, 104).

[73] Christopher B Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. "Universal correspondence network." In: *NIPS*. 2016, pp. 2414–2422 (cit. pp. 81, 83, 146).

[74] O. Chum, J. Matas, and J. Kittler. "Locally Optimized RANSAC." In: *DAGM*. Springer. 2003, p. 236 (cit. pp. 48, 84).

[75] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. "Total Recall: Automatic Query Expansion With a Generative Feature Model for Object Retrieval." In: *ICCV*. 2007 (cit. pp. 49, 50, 60, 73, 79).

[76] Ondrej Chum and Jiri Matas. "Large-Scale Discovery of Spatially Related Images." In: *PAMI* 32.2 (Feb. 2010), pp. 371–377 (cit. pp. 19, 47).

[77] Fan RK Chung. *Spectral graph theory*. Vol. 92. American Mathematical Soc., 1997 (cit. p. 66).

[78] Titus Cieslewski, Michael Bloesch, and Davide Scaramuzza. "Matching Features without Descriptors: Implicitly Matched Interest Points." In: *BMVC*. 2019 (cit. p. 139).

[79] Dan C Cireşan, Ueli Meier, Jonathan Masci, Luca M Gambardella, and Jürgen Schmidhuber. "High-performance neural networks for visual object classification." In: *arXiv preprint arXiv:1102.0183* (2011) (cit. p. 57).

[80] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. "Spherical CNNs." In: *ICLR*. 2018 (cit. p. 147).

[81] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding." In: *CVPR*. 2016 (cit. p. 144).

[82] David Crandall, Lars Backstrom, Daniel Huttenlocher, and Jon Kleinberg. "Mapping the World's Photos." In: *WWW*. 2009 (cit. pp. 19, 45, 46).

[83] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. "Visual Categorization With Bags of Keypoints." In: *ECCV Workshops*. 2004 (cit. pp. 13, 16, 34, 95).

[84] Gabriela Csurka, Christopher R Dance, Florent Perronnin, and Jutta Willamowski. "Generic Visual Categorization Using Weak Geometry." In: *Toward Category-Level Object Recognition*. 2006 (cit. p. 131).

[85] Gabriela Csurka and Florent Perronnin. "Fisher Vectors: Beyond Bag-of-Visual-Words Image Representations." In: *International Conference on Computer Vision, Imaging and Computer Graphics*. 2010 (cit. p. 18).

[86] N. Dalal and B. Triggs. "Histograms of Oriented Gradients for Human Detection." In: *CVPR*. 2005 (cit. p. 15).

[87] John G Daugman. "Uncertainty Relation for Resolution in Space, Spatial Frequency, and Orientation Optimized By Two-Dimensional Visual Cortical Filters." In: *Journal of Optical Society of America* 2.7 (1985), pp. 1160–1169 (cit. p. 15).

[88] John G Daugman. "Complete Discrete 2-D Gabor Transforms By Neural Networks for Image Analysis and Compression." In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 36.7 (1988), pp. 1169–1179 (cit. pp. 13, 15, 16).

[89] Teofilo De Campos, Gabriela Csurka, and Florent Perronnin. "Images As Sets of Locally Weighted Features." In: *CVIU* 116.1 (2012), pp. 68–85 (cit. p. 62).

[90] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." In: *NIPS*. 2016, pp. 3837–3845 (cit. p. 147).

[91] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm." In: *Journal of the Royal Statistical Society. Series B (methodological)* 39.1 (1977), pp. 1–22 (cit. pp. 16, 21, 99).

[92] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In: *CVPR*. Ieee. 2009, pp. 248–255 (cit. p. 125).

[93] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. "ArcFace: Additive angular margin loss for deep face recognition." In: *CVPR*. 2019 (cit. p. 138).

[94] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. "SuperPoint: Self-supervised interest point detection and description." In: *CVPR Workshops*. 2018 (cit. pp. 61, 146).

[95] PGT Dias, AA Kassim, and V Srinivasan. "A neural network based corner detection method." In: *ICNN*. 1995 (cit. p. 61).

[96] Edsger W Dijkstra et al. "A note on two problems in connexion with graphs." In: *Numerische mathematik* 1.1 (1959), pp. 269–271 (cit. p. 97).

[97] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. "Unsupervised Visual Representation Learning By Context Prediction." In: *ICCV*. 2015 (cit. pp. 96, 109, 144).

[98] Carl Doersch and Andrew Zisserman. "Multi-Task Self-Supervised Visual Learning." In: *ICCV*. 2017 (cit. pp. 98, 103, 104).

[99] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. "Behavior Recognition via Sparse Spatio-Temporal Features." In: *VS-PETS* (cit. p. 130).

[100] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. "Decaf: A deep convolutional activation feature for generic visual recognition." In: *ICML*. 2014 (cit. pp. 55, 57, 96, 144).

[101] Michael Donoser and Horst Bischof. "Diffusion processes for retrieval revisited." In: *CVPR*. 2013 (cit. pp. 60, 73, 74, 76).

[102] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. "FlowNet: Learning optical flow with convolutional networks." In: *CVPR*. 2015, pp. 2758–2766 (cit. pp. 83, 146).

[103] A. Doulamis, Y. Avrithis, N. Doulamis, and S. Kollias. "Interactive Content-Based Retrieval in Video Databases Using Fuzzy Classification and Relevance Feedback." In: *Proceedings of IEEE International Conference on Multimedia Computing and Systems*. 1999 (cit. p. 129).

[104] N. Doulamis, A. Doulamis, Y. Avrithis, K. Ntalianis, and S. Kollias. "Efficient Summarization of Stereoscopic Video Sequences." In: *CSVT* 10.4 (June 2000), pp. 501–517 (cit. pp. 129, 137).

[105] Petros Drineas and Michael W Mahoney. "On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning." In: *JMLR* 6.Dec (2005), pp. 2153–2175 (cit. p. 77).

[106] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. "D2-Net: A Trainable CNN for Joint Description and Detection of Local Features." In: *CVPR*. 2019 (cit. p. 139).

[107] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. "Learning SO(3) equivariant representations with spherical cnns." In: *ECCV*. 2018 (cit. p. 147).

[108] G. Evangelopoulos, A. Zlatintsi, A. Potamianos, P. Maragos, K. Rapantzikos, G. Skoumas, and Y. Avrithis. "Multimodal Saliency and Fusion for Movie Summarization based on Aural, Visual, and Textual Attention." In: *IEEE Trans. Multimedia* 15.7 (Nov. 2013), pp. 1553–1568 (cit. p. 130).

[109] F. Perronnin, J. Sánchez, and T. Mensink. "Improving the Fisher kernel for Large-scale image classification." In: *ECCV*. Sept. 2010 (cit. pp. 18, 33, 34, 39, 134).

[110] Jason Farquhar, Sandor Szedmak, Hongying Meng, and John Shawe-Taylor. *Improving bag-of-keypoints image categorisation: Generative models and pdf-kernels*. Tech. rep. University of Southampton, 2005 (cit. p. 16).

[111] Li Fei-Fei, Rob Fergus, and Pietro Perona. "A Bayesian approach to unsupervised one-shot learning of object categories." In: *ICCV*. 2003 (cit. p. 100).

[112] Li Fei-Fei, Rob Fergus, and Pietro Perona. "One-shot learning of object categories." In: *IEEE Trans. PAMI* (2006) (cit. p. 115).

[113] Li Fei-Fei and Pietro Perona. "A Bayesian hierarchical model for learning natural scene categories." In: *CVPR*. 2005 (cit. pp. 15, 16).

[114] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. "PathNet: Evolution channels gradient descent in super neural networks." In: *arXiv preprint arXiv:1701.08734* (2017) (cit. p. 150).

[115] John K. Feser, Marc Brockschmidt, Alexander L. Gaunt, and Daniel Tarlow. "Neural Functional Programming." In: *ICLR Workshops*. 2017 (cit. p. 96).

[116] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks." In: *ICML*. 2017 (cit. pp. 101, 115).

[117] M.A. Fischler and R.C Bolles. "Random Sample Consensus: A Paradigm for Model Fitting With Applications to Image Analysis and Automated Cartography." In: *Communications of the ACM* 24.6 (1981), pp. 381–395 (cit. pp. 17, 27, 84).

[118] Kunihiko Fukushima. "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected By Shift in Position." In: *Biological Cybernetics* 36.4 (1980), pp. 193–202 (cit. pp. 55–57, 95, 147).

[119] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. "Deep Bayesian active learning with image data." In: *ICML*. 2017 (cit. p. 132).

[120] Stephan Gammeter, Lukas Bossard, Till Quack, and Luc V. Gool. "I Know What You Did Last Summer: Object-Level Auto-Annotation of Holiday Snaps." In: *ICCV*. 2009 (cit. pp. 19, 46, 47, 49).

[121] Yaroslav Ganin and Victor S. Lempitsky. "Unsupervised Domain Adaptation by Backpropagation." In: *ICML*. 2015 (cit. p. 96).

[122] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. "Optimized Product Quantization for Approximate Nearest Neighbor Search." In: *CVPR*. 2013 (cit. pp. 18, 39–44).

[123] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. "Optimized Product Quantization." In: *PAMI* (2014) (cit. pp. 41, 43, 44).

[124] Jan C. Van Gemert, Jan-Mark Geusebroek, Cor J. Veenman, and Arnold W.M. Smeulders. "Kernel Codebooks for Scene Categorization." In: *ECCV*. 2008 (cit. p. 16).

[125] Spyros Gidaris and Nikos Komodakis. "Dynamic Few-Shot Visual Learning without Forgetting." In: *CVPR*. 2018 (cit. pp. 101, 113, 117, 118, 120, 138).

[126] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. "Unsupervised Representation Learning by Predicting Image Rotations." In: *ICLR*. 2018 (cit. pp. 96, 109, 144).

[127] Ross Girshick. "Fast R-CNN." In: *ICCV*. 2015 (cit. pp. 58, 62).

[128] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In: *CVPR*. 2014 (cit. pp. 58, 62, 130).

[129] Xavier Glorot and Yoshua Bengio. "Understanding the Difficulty of Training Deep Feedforward Neural Networks." In: *AISTATS*. 2010 (cit. p. 57).

[130] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. "Neighbourhood components analysis." In: *NIPS*. 2005, pp. 513–520 (cit. p. 98).

[131] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. "The reversible residual network: Backpropagation without storing activations." In: *NIPS*. 2017 (cit. p. 150).

[132] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. "Multi-scale orderless pooling of deep convolutional activation features." In: *ECCV*. 2014 (cit. p. 75).

[133] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." In: *arXiv preprint arXiv:1412.6572* (2014) (cit. pp. 101, 122, 125).

[134] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. "Maxout networks." In: *arXiv preprint arXiv:1302.4389* (2013) (cit. p. 61).

[135] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In: *NIPS*. 2014 (cit. p. 95).

[136] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. "Deep Image Retrieval: Learning global representations for image search." In: *ECCV* (2016) (cit. pp. 55, 59, 62, 73, 79, 88, 96, 98, 103, 104, 106, 150).

[137] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. "End-to-End Learning of Deep Visual Representations for Image Retrieval." In: *IJCV* 124.2 (Sept. 2017), pp. 237–254 (cit. pp. 59, 78, 80, 81, 83, 85, 86, 92, 109, 134).

[138] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. "Accurate, large minibatch SGD: Training imagenet in 1 hour." In: *arXiv preprint arXiv:1706.02677* (2017) (cit. p. 143).

[139] Leo Grady. "Random Walks for Image Segmentation." In: *IEEE Trans. PAMI* 28.11 (2006), pp. 1768–1783 (cit. pp. 69, 76, 123).

[140] Benjamin Graham and Laurens van der Maaten. "Submanifold sparse convolutional networks." In: *arXiv preprint arXiv:1706.01307* (2017) (cit. p. 147).

[141] K. Grauman and T. Darrell. "The Pyramid Match Kernel: Discriminative Classification With Sets of Image Features." In: *ICCV*. 2005 (cit. pp. 17, 27).

[142] K. Grauman and T. Darrell. "The Pyramid Match Kernel: Efficient Learning with Sets of Features." In: *JMLR* 8 (2007), pp. 725–760 (cit. pp. 17, 29).

[143] Robert Gray. "Vector quantization." In: *ASSP Magazine, IEEE* 1.2 (1984), pp. 4–29 (cit. pp. 16, 40).

[144] Yinzheng Gu, Chuanpeng Li, and Yu-Gang Jiang. "Towards Optimal CNN Descriptors for Large-Scale Image Retrieval." In: *ACM Multimedia*. 2019 (cit. p. 139).

[145] Çaglar Gülçehre and Yoshua Bengio. "Knowledge matters: Importance of prior information for optimization." In: *JMLR* 17.1 (2016), pp. 226–257 (cit. p. 143).

[146] Chuan Guo, Jared S. Frank, and Kilian Q. Weinberger. "Low Frequency Adversarial Perturbation." In: *arXiv:1809.08758* (2018) (cit. pp. 102, 121).

[147] Agrim Gupta, Piotr Dollar, and Ross Girshick. "LVIS: A Dataset for Large Vocabulary Instance Segmentation." In: *CVPR*. 2019 (cit. p. 143).

[148] H. Jégou, M. Douze, and C. Schmid. "On the burstiness of visual elements." In: *CVPR*. June 2009 (cit. pp. 33, 34, 36).

[149] Wolfgang Hackbusch. *Iterative solution of large sparse systems of equations*. Springer Verlag, 1994 (cit. pp. 60, 76).

[150] Raia Hadsell, Sumit Chopra, and Yann Lecun. "Dimensionality Reduction By Learning an Invariant Mapping." In: *CVPR*. 2006 (cit. pp. 98, 99, 105, 145).

[151] Wilfried Haensch, Tayfun Gokmen, and Ruchir Puri. "The Next Generation of Deep Learning Hardware: Analog Computing." In: *Proceedings of the IEEE* 107.1 (2018), pp. 108–122 (cit. p. 143).

[152] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions." In: *SIAM Review* 53.2 (2011), pp. 217–288 (cit. pp. 70, 77).

[153] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. "MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching." In: *CVPR*. 2015 (cit. pp. 96, 98).

[154] Fusheng Hao, Fengxiang He, Jun Cheng, Lei Wang, Jianzhong Cao, and Dacheng Tao. "Collect and Select: Semantic Alignment Metric Learning for Few-Shot Learning." In: *ICCV*. 2019 (cit. p. 147).

[155] Bharath Hariharan and Ross B. Girshick. "Low-shot Visual Recognition by Shrinking and Hallucinating Features." In: *ICCV*. 2017 (cit. p. 101).

[156] C. Harris and M. Stephens. "A Combined Corner and Edge Detector." In: *Alvey Vision Conference*. 1988 (cit. pp. 14, 130).

[157] Zellig S Harris. "Distributional structure." In: *Word* 10.2-3 (1954), pp. 146–162 (cit. p. 16).

[158] Ben Harwood, Vijay Kumar B G, Gustavo Carneiro, Ian Reid, and Tom Drummond. "Smart Mining for Deep Metric Learning." In: *ICCV*. 2017 (cit. pp. 98, 103, 104, 106, 107, 138).

[159] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009 (cit. p. 24).

[160] James Hays and Alexei A. Efros. "IM2GPS: Estimating Geographic Information From a Single Image." In: *CVPR*. 2008 (cit. p. 45).

[161] Kaiming He, Jian Sun, and Xiaoou Tang. "Guided Image Filtering." In: *ECCV*. 2010 (cit. p. 123).

[162] Kaiming He, Fang Wen, and Jian Sun. "K-Means Hashing: an Affinity-Preserving Quantization Method for Learning Binary Compact Codes." In: *CVPR*. 2013 (cit. p. 39).

[163] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition." In: *ECCV*. 2014 (cit. p. 58).

[164] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification." In: *ICCV*. 2015 (cit. pp. 55, 57).

[165] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In: *CVPR*. 2016 (cit. pp. 55, 57, 78, 85, 103, 113, 143).

[166] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Identity mappings in deep residual networks." In: *ECCV*. 2016 (cit. p. 125).

[167] Xinwei He, Yang Zhou, Zhichao Zhou, Song Bai, and Xiang Bai. "Triplet-Center Loss for Multi-View 3D Object Retrieval." In: *CVPR*. 2018 (cit. p. 148).

[168] Donald Olding Hebb. *The organization of behavior: a neuropsychological theory*. Science Editions, 1962 (cit. p. 150).

[169] Wen Heng, Shuchang Zhou, and Tingting Jiang. "Harmonic Adversarial Attack Method." In: *arXiv:1807.10590* (2018) (cit. pp. 102, 121).

[170] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. "Matrix capsules with EM routing." In: *ICLR*. 2018 (cit. pp. 138, 148).

[171] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks." In: *Science* 313.5786 (2006), pp. 504–507 (cit. p. 98).

[172] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. "Improving neural networks by preventing co-adaptation of feature detectors." In: *arXiv preprint arXiv:1207.0580* (2012) (cit. p. 57).

[173] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. "A Fast Learning Algorithm for Deep Belief Nets." In: *Neural Computation* 18.7 (2006), pp. 1527–1554 (cit. pp. 57, 95).

[174] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." In: *arXiv preprint arXiv:1503.02531* (2015) (cit. p. 145).

[175] Sepp Hochreiter, A Steven Younger, and Peter R Conwell. "Learning to learn using gradient descent." In: *ICANN*. 2001 (cit. p. 115).

[176] David W Hosmer Jr. "A comparison of iterative maximum likelihood estimates of the parameters of a mixture of two normal distributions under three different types of sample." In: *Biometrics* (1973), pp. 761–770 (cit. p. 99).

[177] Qibin Hou, PengTao Jiang, Yunchao Wei, and Ming-Ming Cheng. "Self-erasing network for integral object attention." In: *NeurIPS*. 2018 (cit. p. 147).

[178] Ruibing Hou, Hong Chang, MA Bingpeng, Shiguang Shan, and Xilin Chen. "Cross Attention Network for Few-shot Classification." In: *NeurIPS*. 2019 (cit. p. 147).

[179] Chao-Yung Hsu, Chun-Shien Lu, and Soo-Chang Pei. "Secure and robust SIFT." In: *ACM Multimedia*. 2009 (cit. p. 101).

[180] Miao Hu, Catherine E Graves, Can Li, Yunning Li, Ning Ge, Eric Montgomery, Noraica Davila, Hao Jiang, R Stanley Williams, J Joshua Yang, et al. "Memristor-based analog computation and neural network classification with a dot product engine." In: *Advanced Materials* 30.9 (2018), p. 1705914 (cit. p. 143).

[181] Ronghang Hu, Piotr Dollár, Kaiming He, Trevor Darrell, and Ross Girshick. "Learning to segment every thing." In: *CVPR*. 2018 (cit. p. 144).

[182] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. "Pointwise convolutional neural networks." In: *CVPR*. 2018 (cit. p. 95).

[183] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks." In: *CVPR*. 2017 (cit. p. 57).

[184] Charles H Hubbell. "An input-output approach to clique identification." In: *Sociometry* (1965) (cit. p. 60).

[185] David H Hubel and Torsten N Wiesel. "Receptive Fields of Single Neurones in the Cat's Striate Cortex." In: *The Journal of Physiology* 148.3 (1959), p. 574 (cit. pp. 15, 56).

[186] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In: *ICML*. 2015 (cit. pp. 55, 57).

[187] A. Iscen, G. Tolias, Y. Avrithis, O. Chum, and C. Schmid. "Graph convolutional networks for learning with few clean and many noisy labels." In: *arXiv preprint arXiv:1910.00324* (2019) (cit. pp. 133, 140, 141).

[188] A. Iscen, G. Tolias, Y. Avrithis, T. Furon, and O. Chum. "Panorama to Panorama Matching for Location Recognition." In: *ICMR*. 2017 (cit. pp. 131, 136).

[189] Ahmet Iscen, Yannis Avrithis, Giorgos Tolias, Teddy Furon, and Ondrej Chum. "Fast Spectral Ranking for Similarity Search." In: *CVPR* (2018) (cit. pp. vii, 5, 56, 60, 61, 63, 70, 73, 87, 97, 135, 136, 139, 140, 148).

[190] Ahmet Iscen, Yannis Avrithis, Giorgos Tolias, Teddy Furon, and Ondrej Chum. "Hybrid Diffusion: Spectral-Temporal Graph Filtering for Manifold Ranking." In: *ACCV*. 2018 (cit. pp. 5, 56, 61, 63, 80, 136, 148).

[191] Ahmet Iscen, Teddy Furon, Vincent Gripon, Michael Rabbat, and Hervé Jégou. "Memory vectors for similarity search in high-dimensional spaces." In: *IEEE Trans. Big Data* 4.1 (2018) (cit. pp. 75, 78, 136).

[192] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. "Mining on Manifolds: Metric Learning without Labels." In: *CVPR* (2018) (cit. pp. vii, 6, 70, 97, 98, 102–105, 107, 109, 135, 136, 139, 141, 145, 148).

[193] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. "Label propagation for Deep Semi-supervised Learning." In: *CVPR*. 2019 (cit. pp. vii, 7, 71, 97, 99, 102, 109, 110, 113, 132, 135, 136, 139, 141, 144, 147, 148).

[194] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, Teddy Furon, and Ondrej Chum. "Efficient Diffusion on Region Manifolds: Recovering Small Objects with Compact CNN Representations." In: *CVPR*. 2017 (cit. pp. vii, 5, 56, 60, 61, 63, 70, 73, 75, 78, 85–88, 92, 104, 111, 135, 136, 139, 140, 148).

[195] L. Itti, C. Koch, and E. Niebur. "A Model of Saliency-Based Visual Attention for Rapid Scene Analysis." In: *PAMI* (1998), pp. 1254–1259 (cit. pp. 62, 129).

[196] Jörn-Henrik Jacobsen, Arnold W.M. Smeulders, and Edouard Oyallon. "i-RevNet: Deep Invertible Networks." In: *ICLR*. 2018 (cit. p. 143).

[197] M. Jain, R. Benmokhtar, P. Gros, and H. Jégou. "Hamming embedding similarity-based image classification." In: *ICMR*. June 2012 (cit. p. 37).

[198] M. Jain, H. Jégou, and P. Gros. "Asymmetric Hamming Embedding: Taking the best of our bits for large scale image search." In: *ACM Multimedia*. 2011 (cit. p. 37).

[199] H. Jégou, M. Douze, and C. Schmid. *Exploiting Descriptor Distances for Precise Image Search*. Tech. rep. 2011 (cit. p. 134).

[200] H. Jégou, M. Douze, and C. Schmid. "Product Quantization for Nearest Neighbor Search." In: *PAMI* 33.1 (2011) (cit. pp. 18, 39–42, 44).

[201] Hervé Jégou and Ondrej Chum. "Negative Evidences and Cooccurences in Image Retrieval: The Benefit of PCA and Whitening." In: *ECCV*. Oct. 2012 (cit. p. 108).

[202] Herve Jégou, Matthijs Douze, and Cordelia Schmid. "Hamming embedding and weak geometric consistency for large scale image search." In: *ECCV*. Oct. 2008 (cit. pp. 18, 33, 34, 36, 106, 134).

[203] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. "Improving bag-of-features for large scale image search." In: *IJCV* 87.3 (Feb. 2010), pp. 316–336 (cit. pp. 25, 31–34, 37, 39, 131).

[204] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. "Aggregating local descriptors into a compact image representation." In: *CVPR*. June 2010 (cit. pp. 18, 33, 34, 39, 81, 134).

[205] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. "Aggregating local descriptors into compact codes." In: *Trans. PAMI*. Sept. 2012 (cit. p. 18).

[206] Herve Jégou, Romain Tavenard, Matthijs Douze, and Laurent Amsaleg. "Searching in One Billion Vectors: Re-Rank with Source Coding." In: *ICASSP*. 2011 (cit. pp. 43, 44, 134).

[207] Albert Jimenez, Jose M Alvarez, and Xavier Giro-i-Nieto. "Class-Weighted Convolutional Features for Visual Instance Search." In: *BMVC* (2017) (cit. p. 87).

[208] Yushi Jing and Shumeet Baluja. "VisualRank: Applying PageRank to large-scale image search." In: *IEEE Trans. PAMI* 30.11 (2008), pp. 1877–1890 (cit. pp. 73, 136).

[209] Thorsten Joachims. "Transductive inference for text classification using support vector machines." In: *ICML*. Vol. 99. 1999 (cit. pp. 99, 129).

[210] Judson P Jones and Larry A Palmer. "An Evaluation of the Two-Dimensional Gabor Filter Model of Simple Receptive Fields in Cat Striate Cortex." In: *Journal of Neurophysiology* 58.6 (1987), pp. 1233–1258 (cit. p. 15).

[211] Bela Julesz. "Textons, the Elements of Texture Perception, and Their Interactions." In: *Nature* 290.5802 (1981), pp. 91–97 (cit. p. 16).

[212] F. Jurie and B. Triggs. "Creating Efficient Codebooks for Visual Recognition." In: *ICCV*. 2005 (cit. pp. 15, 16).

[213] Yannis Kalantidis and Yannis Avrithis. "Locally Optimized Product Quantization for Approximate Nearest Neighbor Search." In: *CVPR*. 2014 (cit. pp. vii, 4, 13, 18, 20, 39, 134, 135, 139, 148).

[214] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. "Cross-Dimensional Weighting for Aggregated Deep Convolutional Features." In: *ECCV Workshops*. 2016 (cit. pp. 63, 73, 75, 79, 81, 82, 87, 88, 137).

[215] Yannis Kalantidis, Giorgos Tolias, Yannis Avrithis, Marios Phinikettos, Evaggelos Spyrou, Phivos Mylonas, and Stefanos Kollias. "VIRaL: Visual Image Retrieval and Localization." In: *Multimedia Tools and Applications* 51.2 (2011), pp. 555–592 (cit. pp. vii, 4, 14, 19, 20, 45, 50, 51).

[216] Nandakishore Kambhatla and Todd K Leen. "Dimension Reduction By Local Principal Component Analysis." In: *Neural Computation* 9.7 (1997), pp. 1493–1516 (cit. p. 18).

[217] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. "Decoupling representation and classifier for long-tailed recognition." In: *arXiv preprint arXiv:1910.09217* (2019) (cit. p. 146).

[218] P. Kapsalas, K. Rapantzikos, A. Sofou, and Y. Avrithis. "Regions Of Interest for Accurate Object Detection." In: *CBMI*. 2008 (cit. pp. 130, 137).

[219] Leo Katz. "A New Status Index Derived From Sociometric Analysis." In: *Psychometrika* 18.1 (1953), pp. 39–43 (cit. pp. 60, 69, 70, 91).

[220] L. Kennedy, M. Naaman, S. Ahern, R. Nair, and T. Rattenbury. "How Flickr Helps Us Make Sense of the World: Context and Content in Community-Contributed Media Collections." In: *ACM Multimedia*. Vol. 3. 2007, pp. 631–640 (cit. pp. 19, 46, 47).

[221] Dahun Kim, Donghyeon Cho, Donggeun Yoo, and In So Kweon. "Two-Phase Learning for Weakly Supervised Object Localization." In: *ICCV*. 2017 (cit. p. 147).

[222] Gunhee Kim and Antonio Torralba. "Unsupervised detection of regions of interest using iterative link analysis." In: *NIPS*. 2009 (cit. p. 63).

[223] Seungryong Kim, Dongbo Min, Bumsub Ham, Sangryul Jeon, Stephen Lin, and Kwanghoon Sohn. "FCSS: Fully convolutional self-similarity for dense semantic correspondence." In: *CVPR*. 2017 (cit. p. 83).

[224] Tae Hoon Kim, Kyoung Mu Lee, and Sang Uk Lee. "Generative Image Segmentation Using Random Walks with Restart." In: *ECCV*. Springer. 2008, pp. 264–275 (cit. pp. 69, 111, 123).

[225] Diederik P Kingma and Max Welling. "Auto-encoding variational Bayes." In: *arXiv preprint arXiv:1312.6114* (2013) (cit. p. 95).

[226] Diederik Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *arXiv:1412.6980* (2015) (cit. p. 123).

[227] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks." In: *ICLR*. 2017 (cit. pp. 147, 149).

[228] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. "Overcoming Catastrophic Forgetting in Neural Networks." In: *Proceedings of the National Academy of Sciences* 114.13 (2017), pp. 3521–3526 (cit. p. 150).

[229] C Koch and S Ullman. "Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry." In: *Human Neurobiology* 4.4 (1985), pp. 219–227 (cit. p. 62).

[230] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." In: *ICMLW*. 2015 (cit. p. 115).

[231] Iasonas Kokkinos. "UberNet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory." In: *CVPR*. 2017 (cit. p. 96).

[232] Peter Kontschieder, Michael Donoser, and Horst Bischof. "Beyond pairwise shape similarity analysis." In: *ACCV*. 2009 (cit. p. 74).

[233] Martin Kostinger, Paul Wohlhart, Peter M Roth, and Horst Bischof. "Joint Learning of Discriminative Prototypes and Large Margin Nearest Neighbor Classifiers." In: *ICCV*. 2013 (cit. pp. 138, 150).

[234] Alex Krizhevsky and Geoffrey Hinton. *Learning multiple layers of features from tiny images*. Tech. rep. University of Toronto, 2009 (cit. p. 113).

[235] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *NIPS*. 2012 (cit. pp. 55, 57, 58, 101).

[236] BG Kumar, Gustavo Carneiro, Ian Reid, et al. "Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions." In: *CVPR*. 2016 (cit. p. 108).

[237] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world." In: *arXiv:1607.02533* (2016) (cit. pp. 101, 122, 125).

[238] Alexey Kurakin et al. "Adversarial Attacks and Defences Competition." In: *arXiv:1804.00097* (2018) (cit. p. 121).

[239] Alexey Kurakin, Ian Goodfellow, Samy Bengio, Yinpeng Dong, Fangzhou Liao, Ming Liang, Tianyu Pang, Jun Zhu, Xiaolin Hu, Cihang Xie, et al. "Adversarial attacks and defences competition." In: *arXiv:1804.00097* (2018) (cit. p. 125).

[240] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale." In: *arXiv preprint arXiv:1811.00982* (2018) (cit. p. 143).

[241] Samuli Laine and Timo Aila. "Temporal ensembling for semi-supervised learning." In: *ICLR*. 2017 (cit. pp. 99, 113).

[242] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. "Human-level concept learning through probabilistic program induction." In: *Science* 350.6266 (Dec. 2015), pp. 1332–1338 (cit. p. 143).

[243] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. "One shot learning of simple visual concepts." In: *Proceedings of the Annual Meeting of the Cognitive Science Society*. 2011 (cit. p. 100).

[244] Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. "Beyond Sliding Windows: Object Localization By Efficient Subwindow Search." In: *CVPR*. 2008 (cit. p. 143).

[245] Barbara Landau, Linda B Smith, and Susan S Jones. "The importance of shape in early lexical learning." In: *Cognitive development* 3.3 (1988), pp. 299–321 (cit. p. 100).

[246] Amy N Langville and Carl D Meyer. "Deeper Inside PageRank." In: *Internet Mathematics* 1.3 (2004), pp. 335–380 (cit. p. 76).

[247] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. "Autoencoding beyond pixels using a learned similarity metric." In: *arXiv preprint arXiv:1512.09300* (2015) (cit. p. 102).

[248] Marc Law, Nicolas Thome, and Matthieu Cord. "Quadruplet-Wise Image Similarity Learning." In: *ICCV*. 2013 (cit. p. 108).

[249] S. Lazebnik, C. Schmid, and J. Ponce. "Beyond bags of features: spatial pyramid matching for recognizing natural scene categories." In: *CVPR*. June 2006 (cit. pp. 131, 134).

[250] Y Le Cun, B Boser, JS Denker, D Henderson, Richard E Howard, W Hubbard, and Lawrence D Jackel. "Handwritten Digit Recognition with a Back-Propagation Network." In: *NIPS*. 1990 (cit. pp. 55, 56).

[251] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-Based Learning Applied to Document Recognition." In: *Proc. of the IEEE* 86.11 (1998) (cit. p. 56).

[252] Dong-Hyun Lee. "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks." In: *ICMLW*. 2013 (cit. pp. 99, 109, 110).

[253] John A Lee and Michel Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007 (cit. p. 97).

[254] B. Leibe and B. Schiele. "Interleaved Object Categorization and segmentation." In: *BMVC*. 2003 (cit. p. 17).

[255] Darui Li, Linjun Yang, Xian-Sheng Hua, and Hong-Jiang Zhang. "Large-Scale Robust Visual Codebook Construction." In: *ACM Multimedia*. 2010 (cit. pp. 21, 25).

[256] Dong Li, Wei-Chih Hung, Jia-Bin Huang, Shengjin Wang, Narendra Ahuja, and Ming-Hsuan Yang. "Unsupervised visual representation learning by graph-based consistent constraints." In: *ECCV*. 2016 (cit. pp. 107, 108).

[257] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. "Visualizing the loss landscape of neural nets." In: *NeurIPS*. 2018 (cit. p. 57).

[258] Yunpeng Li, David J. Crandall, and Daniel P. Huttenlocher. "Landmark Classification in Large-Scale Image Collections." In: *ICCV*. 2009 (cit. pp. 45, 46).

[259] Zhizhong Li and Derek Hoiem. "Learning without forgetting." In: *PAMI* 40.12 (2017), pp. 2935–2947 (cit. pp. 96, 144).

[260] Y. Lifchitz, Y. Avrithis, and S. Picard. "Few-shot Few-shot Learning and the role of Spatial Attention." Under review (cit. pp. 133, 140, 141).

[261] Y. Lifchitz, Y. Avrithis, S. Picard, and A. Bursuc. "Dense Classification and Implanting for Few-shot Learning." In: *CVPR*. 2019 (cit. pp. vii, 7, 96, 97, 101, 102, 115, 140, 141).

[262] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature pyramid networks for object detection." In: *CVPR*. 2017 (cit. p. 95).

[263] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. "Focal Loss for Dense Object Detection." In: *ICCV*. 2017 (cit. pp. 62, 143).

[264] T. Lindeberg. "Feature Detection with Automatic Scale Selection." In: *IJCV* 30.2 (1998), pp. 79–116 (cit. pp. 13, 14).

[265] T. Lindeberg and J. Garding. "Shape-Adapted Smoothing in Estimation of 3-D Shape Cues From Affine Deformations of Local 2-D Brightness structure." In: *Image and Vision Computing* 15.6 (1997), pp. 415–434 (cit. p. 14).

[266] Tony Lindeberg. "Scale-Space Theory: A Basic Tool for Analyzing Structures at Different Scales." In: *Journal of Applied Statistics* 21.1-2 (1994), pp. 225–270 (cit. p. 14).

[267] Chundi Liu, Guangwei Yu, Maksims Volkovs, Cheng Chang, Himanshu Rai, Junwei Ma, and Satya Krishna Gorti. "Guided Similarity Separation for Image Retrieval." In: *NeurIPS*. 2019 (cit. pp. 139, 144, 149).

[268] Hanxiao Liu, Karen Simonyan, and Yiming Yang. "DARTS: Differentiable architecture search." In: *arXiv preprint arXiv:1806.09055* (2018) (cit. p. 96).

[269] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. "SphereFace: Deep hypersphere embedding for face recognition." In: *CVPR*. 2017 (cit. pp. 96, 138).

[270] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. "Delving into Transferable Adversarial Examples and Black-box Attacks." In: *ICLR*. 2017 (cit. p. 101).

[271] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X. Yu. "Large-Scale Long-Tailed Recognition in an Open World." In: *CVPR*. 2019 (cit. p. 146).

[272] Jonathan L Long, Ning Zhang, and Trevor Darrell. "Do Convnets Learn Correspondence?" In: *NIPS*. 2014 (cit. pp. 61, 81, 83, 146).

[273] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation." In: *CVPR*. 2015 (cit. p. 118).

[274] David G Lowe. "Object recognition from local scale-invariant features." In: *ICCV*. 1999, pp. 1150–1157 (cit. pp. 13–15, 17, 81).

[275] D.G. Lowe. "Local Feature View Clustering for 3D Object Recognition." In: *CVPR*. 2001 (cit. p. 49).

[276] D.G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints." In: *IJCV* 60.2 (2004), pp. 91–110 (cit. pp. 13–15, 17, 27).

[277] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. "Deep photo style transfer." In: *CVPR* (2017) (cit. p. 102).

[278] B.D. Lucas and T. Kanade. "An Iterative Image Registration Technique With an Application to Stereo Vision." In: *IJCAI*. 1981 (cit. p. 14).

[279] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards deep learning models resistant to adversarial attacks." In: *arXiv preprint arXiv:1706.06083* (2017) (cit. p. 102).

[280] Dhruv Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. "Exploring the Limits of Weakly Supervised Pretraining." In: *ECCV*. 2018 (cit. p. 144).

[281] Prasanta Chandra Mahalanobis. "On the generalized distance in statistics." In: *Proceedings of the National Institute of Science of India* 2.1 (1936), pp. 49–55 (cit. p. 98).

[282] J. Malik, S. Belongie, J. Shi, and T. Leung. "Textons, Contours and Regions: Cue Integration in Image Segmentation." In: *ICCV*. 1999 (cit. p. 16).

[283] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. "Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights." In: *ECCV*. 2018 (cit. pp. 96, 115).

[284] Arun Mallya and Svetlana Lazebnik. "Packnet: Adding multiple tasks to a single network by iterative pruning." In: *CVPR* (2018) (cit. p. 115).

[285] BS Manjunath and WY Ma. "Texture Features for Browsing and Retrieval of Image Data." In: *PAMI* 18.8 (1996), pp. 837–842 (cit. pp. 13, 15).

[286] C. D. Manning, P. Raghavan, and H. Schütze. "Introduction to Information Retrieval." In: Cambridge University Press, 2008. Chap. Relevance feedback & query expansion (cit. pp. 19, 106).

[287] S Marcelja. "Mathematical Description of the Responses of Simple Cortical Cells." In: *Journal of Optical Society of America* 70.11 (1980), pp. 1297–1300 (cit. p. 15).

[288] D. Marr and E. Hildreth. "Theory of Edge Detection." In: *Proceedings of the Royal Society of London* 207.1167 (1980), pp. 187–217 (cit. p. 14).

[289] J. Matas, O. Chum, U. Martin, and T. Pajdla. "Robust wide baseline stereo from maximally stable extremal regions." In: *BMVC*. Sept. 2002, pp. 384–393 (cit. p. 83).

[290] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. "Metric Learning for Large Scale Image Classification: Generalizing to New Classes at Near-Zero Cost." In: *ECCV*. 2012 (cit. p. 100).

[291] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. "Distance-Based Image Classification: Generalizing to New Classes at Near-Zero Cost." In: *PAMI* 35.11 (2013), pp. 2624–2637 (cit. pp. 138, 146).

[292] Benjamin J. Meyer, Ben Harwood, and Tom Drummond. "Nearest Neighbour Radial Basis Function Solvers for Deep Neural Networks." In: *arXiv preprint arXiv:1705.09780* (2017) (cit. pp. 96, 100, 138).

[293] Thomas Miconi, Jeff Clune, and Kenneth O Stanley. "Differentiable plasticity: training plastic neural networks with backpropagation." In: *arXiv preprint arXiv:1804.02464* (2018) (cit. p. 150).

[294] K. Mikolajczyk and C. Schmid. "Scale & Affine Invariant Interest Point Detectors." In: *IJCV* 60.1 (2004), pp. 63–86 (cit. pp. 14, 81).

[295] K. Mikolajczyk and C. Schmid. "A Performance Evaluation of Local Descriptors." In: *PAMI* 27.10 (2005), pp. 1615–1630 (cit. p. 15).

[296] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L.V Gool. "A Comparison of Affine Region Detectors." In: *IJCV* 65.1 (2005), pp. 43–72 (cit. p. 14).

[297] Krystian Mikolajczyk and Jiri Matas. "Improving Descriptors for Fast Tree Matching By Optimal Linear Projection." In: *CVPR*. 2007 (cit. pp. 78, 85, 90, 108).

[298] A. Mikulík, M. Perdoch, O. Chum, and J. Matas. "Learning a fine vocabulary." In: *ECCV*. Sept. 2010 (cit. pp. 18, 33, 37, 39).

[299] Erik G Miller, Nicholas E Matsakis, and Paul A Viola. "Learning From One Example Through Shared Densities on Transforms." In: *CVPR*. 2000 (cit. p. 100).

[300] Marvin L Minsky and Seymour Papert. *Perceptrons: an Introduction to Computational Geometry, Expanded Edition*. MIT, 1969 (cit. p. 56).

[301] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. "A simple neural attentive meta-learner." In: *ICLR* (2018) (cit. p. 117).

[302] Eva Mohedano, Kevin McGuinness, Xavier Giro-i-Nieto, and Noel E O'Connor. "Saliency Weighted Convolutional Features for Instance Search." In: *arXiv preprint arXiv:1711.10795* (2017) (cit. pp. 62, 87).

[303] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. "On the number of linear regions of deep neural networks." In: *NIPS*. 2014 (cit. p. 57).

[304] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. "Universal Adversarial Perturbations." In: *CVPR*. 2017, pp. 86–94 (cit. pp. 101, 121, 122).

[305] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks." In: *CVPR*. 2016 (cit. p. 122).

[306] Konda Reddy Mopuri and R Venkatesh Babu. "Object level deep feature pooling for compact image representation." In: *CVPR*. 2015 (cit. pp. 62, 88).

[307] J Anthony Movshon, Ian D Thompson, and David J Tolhurst. "Spatial Summation in the Receptive Fields of Simple Cells in the Cat's Striate Cortex." In: *The Journal of Physiology* 283.1 (1978), pp. 53–77 (cit. p. 15).

[308] Yair Movshovitz-Attias, Alexander Toshev, Thomas K. Leung, Sergey Ioffe, and Saurabh Singh. "No Fuss Distance Metric Learning Using Proxies." In: *ICCV*. 2017 (cit. pp. 103, 104, 138, 150).

[309] M. Muja and D.G Lowe. "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration." In: *ICCV*. 2009 (cit. pp. 25, 39, 44).

[310] Naila Murray and Florent Perronnin. "Generalized max-pooling." In: *CVPR*. June 2014 (cit. pp. 75, 78).

[311] J. Mutch and D.G Lowe. "Multiclass Object Recognition With Sparse, Localized features." In: *CVPR*. 2006 (cit. p. 15).

[312] Seil Na, Sangho Lee, Jisung Kim, and Gunhee Kim. "A Read-Write Memory Network for Movie Story Understanding." In: *ICCV*. 2017 (cit. p. 144).

[313] Vinod Nair and Geoffrey E Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines." In: *ICML*. 2010 (cit. pp. 55, 57).

[314] M.E.J. Newman. *Networks: an introduction*. Oxford University Press, Oxford, 2010 (cit. pp. 90, 91).

[315] Quynh Nguyen, Mahesh Chandra Mukkamala, and Matthias Hein. "On the loss landscape of a class of deep neural networks with no bad local valleys." In: *arXiv preprint arXiv:1809.-10749* (2018) (cit. p. 57).

[316] David Nistér and Henrik Stewénius. "Scalable Recognition With a Vocabulary Tree." In: *CVPR*. 2006 (cit. pp. 17, 21, 81).

[317] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer, 2006 (cit. pp. 60, 65, 69, 70, 73, 76, 90, 104, 125).

[318] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. "Largescale image retrieval with attentive deep local features." In: *ICCV*. 2017 (cit. pp. 61, 63, 81, 82, 85–87, 146).

[319] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. "Learning deconvolution network for semantic segmentation." In: *ICCV*. 2015 (cit. p. 118).

[320] M. Norouzi and D. Fleet. "Cartesian k-Means." In: *CVPR*. 2013 (cit. pp. 41, 42, 44).

[321] Mohammad Norouzi, Ali Punjani, and David J Fleet. "Fast Search in Hamming Space with Multi-Index Hashing." In: *CVPR*. 2012 (cit. p. 44).

[322] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. "Deep metric learning via lifted structured feature embedding." In: *CVPR*. 2016 (cit. pp. 96, 98, 100, 103, 106–108, 138, 145, 150).

[323] A. Oliva and A. Torralba. "Building the Gist of a Scene: the Role of Global Image Features in Recognition." In: *Visual Perception* (2006) (cit. p. 15).

[324] Aude Oliva, Antonio B Torralba, Anne Guérin-Dugué, and Jeanny Hérault. "Global Semantic Classification of Scenes Using Power Spectrum Templates." In: *Challenge of Image Retrieval*. 1999 (cit. pp. 13, 15).

[325] C. Olsson, A.P. Eriksson, and F. Kahl. "Solving Large Scale Binary Quadratic Problems: Spectral Methods vs. Semidefinite Programming." In: *CVPR*. 2007 (cit. p. 29).

[326] Alan V Oppenheim and Ronald W Schafer. *Discrete-Time Signal Processing: Pearson New International Edition*. Pearson Higher Ed, 2010 (cit. pp. 65, 67).

[327] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. "Learning and transferring mid-level image representations using convolutional neural networks." In: *CVPR*. 2014 (cit. pp. 96, 144).

[328] Boris N Oreshkin, Alexandre Lacoste, and Pau Rodriguez. "TADAM: Task dependent adaptive metric for improved few-shot learning." In: *NeurIPS* (2018) (cit. pp. 117, 120).

[329] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. "The PageRank citation ranking: bringing order to the web." In: (1999) (cit. pp. 60, 66, 73).

[330] Jia-Yu Pan, Hyung-Jeong Yang, Christos Faloutsos, and Pinar Duygulu. "Automatic Multimedia Cross-Modal Correlation Discovery." In: *International Conference on Knowledge Discovery and Data Mining*. ACM. 2004 (cit. pp. 60, 68, 74).

[331] Dim P. Papadopoulos, Jasper R. R. Uijlings, Frank Keller, and Vittorio Ferrari. "Extreme Clicking for Efficient Object Annotation." In: *ICCV*. 2017 (cit. p. 143).

[332] Deepak Pathak, Ross B Girshick, Piotr Dollár, Trevor Darrell, and Bharath Hariharan. "Learning Features by Watching Objects Move." In: *CVPR*. 2017 (cit. pp. 96, 109, 144, 147).

[333] Karl Pearson. "On lines and planes of closest fit to systems of points in space." In: *Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572 (cit. p. 97).

[334] Michal Perdoch, Ondrej Chum, and Jiri Matas. "Efficient Representation of Local Geometry for Large Scale Object Retrieval." In: *CVPR*. June 2009 (cit. pp. 37, 81).

[335] Patrick Pérez, Michel Gangnet, and Andrew Blake. "Poisson Image Editing." In: *ACM Transactions on Graphics* 22.3 (2003), pp. 313–318 (cit. p. 69).

[336] F. Perronnin. "Universal and Adapted Vocabularies for Generic Visual Categorization." In: *PAMI* 30.7 (2008), pp. 1243–1256 (cit. p. 21).

[337] F. Perronnin, C. Dance, G. Csurka, and M. Bressan. "Adapted Vocabularies for Generic Visual Categorization." In: *ECCV*. 2006 (cit. p. 16).

[338] F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier. "Large-Scale Image Retrieval with Compressed Fisher Vectors." In: *CVPR*. June 2010 (cit. p. 33).

[339] Florent Perronnin and Christopher R. Dance. "Fisher Kernels on Visual Vocabularies for Image Categorization." In: *CVPR*. June 2007 (cit. pp. 18, 33, 134).

[340] Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. "Digital photography with flash and no-flash image pairs." In: *ACM Transactions on Graphics* 23.3 (2004), pp. 664–672 (cit. p. 123).

[341] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. "Object Retrieval With Large Vocabularies and Fast Spatial Matching." In: *CVPR*. 2007 (cit. pp. 16, 17, 19, 21, 25, 27, 31, 37, 45, 47, 48, 50, 60, 79–81, 84–87, 106, 130, 134, 139).

[342] James Philbin, Ondrej Chum, Josef Sivic, Michael Isard, and Andrew Zisserman. "Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases." In: *CVPR*. 2008 (cit. pp. 17, 19, 25, 32, 33, 37, 39, 60, 79, 80, 106).

[343] Philip Pritchett and Andrew Zisserman. "Wide Baseline Stereo Matching." In: *ICCV*. 1998 (cit. p. 14).

[344] Gilles Puy, Srdan Kitic, and Patrick Pérez. "Unifying local and non-local signal processing with graph CNNs." In: *arXiv preprint arXiv:1702.07759* (2017) (cit. p. 147).

[345] Gilles Puy and Patrick Pérez. "A Flexible Convolutional Solver for Fast Style Transfers." In: *CVPR*. 2019 (cit. p. 102).

[346] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation." In: *CVPR*. 2017 (cit. p. 147).

[347] Hang Qi, Matthew Brown, and David G Lowe. "Low-Shot Learning With Imprinted Weights." In: *CVPR*. 2018 (cit. pp. 101, 117, 118, 138, 150).

[348] Weichao Qiu and Alan Yuille. "UnrealCV: Connecting computer vision to unreal engine." In: *ECCV Workshops*. 2016 (cit. p. 144).

[349] T. Quack, B. Leibe, and L. Van Gool. "World-Scale Mining of Objects and Events From Community Photo Collections." In: *CIVR*. 2008, pp. 47–56 (cit. pp. 19, 46, 47).

[350] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. "Revisiting Oxford and Paris: Large-Scale Image Retrieval Benchmarking." In: *CVPR*. 2018 (cit. pp. 18, 20, 61, 63, 80, 82, 85, 86, 91, 134, 139, 140, 144).

[351] Filip Radenović, Giorgos Tolias, and Ondřej Chum. "CNN Image Retrieval Learns from BoW: Unsupervised Fine-Tuning with Hard Examples." In: *ECCV* (2016) (cit. pp. 55, 59, 75, 78, 85, 90, 96, 98, 103, 104, 106, 108, 150).

[352] Filip Radenović, Giorgos Tolias, and Ondřej Chum. "Fine-tuning CNN Image Retrieval with No Human Annotation." In: *IEEE Trans. PAMI* (2018) (cit. pp. 59, 62, 81–83, 85, 92, 109, 134).

[353] Adityanarayanan Radhakrishnan, Karren Yang, Mikhail Belkin, and Caroline Uhler. "Memorization in Overparameterized Autoencoders." In: *arXiv preprint arXiv:1810.10333* (2018) (cit. p. 144).

[354] Ilija Radosavovic, Piotr Dollar, Ross Girshick, Georgia Gkioxari, and Kaiming He. "Data Distillation: Towards Omni-Supervised Learning." In: *CVPR*. 2018 (cit. pp. 109, 145).

[355] Rajeev Ranjan, Carlos D Castillo, and Rama Chellappa. "L2-constrained softmax loss for discriminative face verification." In: *arXiv preprint arXiv:1703.09507* (2017) (cit. pp. 101, 138).

[356] Marc'Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann Lecun. "Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition." In: *CVPR*. 2007 (cit. pp. 57, 95).

[357] K. Rapantzikos, Y. Avrithis, and S. Kollias. "Dense saliency-based spatiotemporal feature points for action recognition." In: *CVPR*. 2009 (cit. p. 130).

[358] K. Rapantzikos, Y. Avrithis, and S. Kollias. "Detecting Regions from Single Scale Edges." In: *ECCV Workshops*. 2010 (cit. p. 130).

[359] K. Rapantzikos, Y. Avrithis, and S. Kollias. "Spatiotemporal features for action recognition and salient event detection." In: *Cognitive Computation* 3.1 (Mar. 2011), pp. 167–184 (cit. pp. 130, 137).

[360] K. Rapantzikos, Y. Avrithis, and S. Kollias. "Vision, Attention Control, and Goals Creation System." In: *Perception-Action Cycle: Models, Architectures and Hardware*. Ed. by V. Cutsuridis, A. Hussain, and J. G. Taylor. Springer, 2011, pp. 363–386 (cit. p. 130).

[361] K. Rapantzikos, N. Tsapatsoulis, Y. Avrithis, and S. Kollias. "Spatiotemporal Saliency for Video Classification." In: *Signal Processing: Image Communication* 24.7 (Aug. 2009), pp. 557–571 (cit. p. 130).

[362] Sachin Ravi and Hugo Larochelle. "Optimization as a model for few-shot learning." In: *ICLR* (2017) (cit. pp. 96, 101, 113, 120).

[363] Ali S Razavian, Josephine Sullivan, Stefan Carlsson, and Atsuto Maki. "Visual instance retrieval with deep convolutional networks." In: *ITE Transactions on Media Technology and Applications* 4 (2016), pp. 251–258 (cit. pp. 58, 75, 78, 80, 87, 88).

[364] Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. "Visual Instance Retrieval with Deep Convolutional Networks." In: *arXiv preprint arXiv:1412.6574* (2014) (cit. pp. 58, 60).

[365] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. "iCaRL: Incremental classifier and representation learning." In: *CVPR*. 2017 (cit. pp. 96, 144, 150).

[366] Angad S Rekhi, Brian Zimmer, Nikola Nedovic, Ningxi Liu, Rangharajan Venkatesan, Miaorong Wang, Brucek Khailany, William J Dally, and C Thomas Gray. "Analog/Mixed-Signal Hardware Error Modeling for Deep Learning Inference." In: *Proceedings of Annual Design Automation Conference*. ACM. 2019 (cit. p. 143).

[367] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks." In: *NIPS*. 2015, pp. 91–99 (cit. pp. 58, 59, 62, 130).

[368] Jerome Revaud, Jon Almazan, Rafael S. Rezende, and Cesar Roberto de Souza. "Learning With Average Precision: Training Image Retrieval With a Listwise Loss." In: *ICCV*. 2019 (cit. pp. 138, 145).

[369] Stephan R. Richter, Zeeshan Hayder, and Vladlen Koltun. "Playing for Benchmarks." In: *ICCV*. 2017 (cit. p. 144).

[370] Oren Rippel, Manohar Paluri, Piotr Dollar, and Lubomir Bourdev. "Metric learning with adaptive density discrimination." In: *arXiv*. 2015 (cit. pp. 103, 138, 146, 150).

[371] Herbert Robbins and Sutton Monro. "A stochastic approximation method." In: *The Annals of Mathematical Statistics* (1951), pp. 400–407 (cit. pp. 55, 56).

[372] Ignacio Rocco, Relja Arandjelovic, and Josef Sivic. "End-to-end weakly-supervised semantic alignment." In: *CVPR*. 2018 (cit. pp. 81, 83, 139, 146).

[373] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelović, Akihiko Torii, Tomas Pajdla, and Josef Sivic. "Neighbourhood Consensus Networks." In: *NeurIPS*. 2018 (cit. p. 81).

[374] Vladimir Rokhlin, Arthur Szlam, and Mark Tygert. "A Randomized Algorithm for Principal Component Analysis." In: *SIAM Journal on Matrix Analysis and Applications* 31.3 (2009), pp. 1100–1124 (cit. pp. 70, 77).

[375] Frank Rosenblatt. *Principles of Neurodynamics*. Spartan Books, 1962 (cit. pp. 56, 57).

[376] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning Representations By Back-Propagating Errors." In: *Nature* 5.3 (1986), p. 1 (cit. pp. 55, 56).

[377] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. "ImageNet large scale visual recognition challenge." In: *IJCV* 115.3 (2015), pp. 211–252 (cit. pp. 55, 57, 120).

[378] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. "Progressive neural networks." In: *arXiv preprint arXiv:1606.04671* (2016) (cit. pp. 115, 150).

[379] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. "Déjà Vu: an empirical evaluation of the memorization properties of ConvNets." In: *arXiv preprint arXiv:1809.06396* (2018) (cit. p. 144).

[380] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. "Spreading vectors for similarity search." In: *ICLR*. 2019 (cit. p. 139).

[381] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. "Regularization with stochastic transformations and perturbations for deep semi-supervised learning." In: *NIPS*. 2016 (cit. p. 109).

[382] Amaia Salvador, Xavier Giró-i-Nieto, Ferran Marqués, and Shin'ichi Satoh. "Faster R-CNN features for instance search." In: *CVPR Workshops*. 2016 (cit. pp. 62, 87, 88).

[383] Aliaksei Sandryhaila and Jose MF Moura. "Discrete Signal Processing on Graphs." In: *IEEE Signal Processing Magazine* 61.7 (2013), pp. 1644–1656 (cit. pp. 60, 65, 73, 76).

[384] Aliaksei Sandryhaila and José MF Moura. "Discrete Signal Processing on Graphs: Frequency Analysis." In: *IEEE Trans. Signal Processing* 62.12 (2014), pp. 3042–3054 (cit. pp. 66–68).

[385] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. "Meta-learning with memory-augmented neural networks." In: *ICML*. 2016 (cit. p. 115).

[386] L. Saul and S. Roweis. "Think globally, fit locally: unsupervised learning of low dimensional manifolds." In: *JMLR* 4 (2003), pp. 119–155 (cit. p. 103).

[387] G. Schindler, M. Brown, and R. Szeliski. "City-Scale Location Recognition." In: *CVPR*. 2007 (cit. p. 19).

[388] Jürgen Schmidhuber. "Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook." MA thesis. Technische Universität München, 1987 (cit. p. 100).

[389] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Nonlinear component analysis as a kernel eigenvalue problem." In: *Neural computation* 10.5 (1998), pp. 1299–1319 (cit. pp. 60, 97).

[390] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering." In: *CVPR*. 2015 (cit. pp. 96, 98, 107).

[391] Christian Schuldt, Ivan Laptev, and Barbara Caputo. "Recognizing Human Actions: a Local SVM Approach." In: *ICPR*. 2004 (cit. p. 130).

[392] H Scudder. "Probability of error of some adaptive pattern-recognition machines." In: *IEEE Transactions on Information Theory* 11.3 (1965), pp. 363–371 (cit. pp. 98, 99).

[393] John R Seeley. "The Net of Reciprocal Influence. A Problem in Treating Sociometric Data." In: *Canadian Journal of Experimental Psychology* 3 (1949), p. 234 (cit. p. 59).

[394] Ozan Sener and Silvio Savarese. "Active Learning for Convolutional Neural Networks: A Core-Set Approach." In: *ICLR*. 2018 (cit. pp. 96, 132).

[395] T. Serre, L. Wolf, and T. Poggio. "Object Recognition with Features Inspired By Visual Cortex." In: *CVPR*. 2005 (cit. pp. 15, 56).

[396] Burr Settles. *Active learning literature survey*. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences, 2009 (cit. pp. 96, 132).

[397] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R Stanley Williams, and Vivek Srikumar. "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars." In: *ACM SIGARCH Computer Architecture News* 44.3 (2016), pp. 14–26 (cit. p. 143).

[398] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. "CNN features off-the-shelf: an astounding baseline for recognition." In: *CVPR Workshops*. 2014 (cit. pp. 55, 57).

[399] Mahmood Sharif, Lujo Bauer, and Michael K. Reiter. "On the Suitability of L_p-Norms for Creating and Preventing Adversarial Examples." In: *arXiv:1802.09653* (2018) (cit. p. 121).

[400] Miaojing Shi, Yannis Avrithis, and Herve Jegou. "Early Burst Detection for Memory-Efficient Image Retrieval." In: *CVPR*. 2015 (cit. p. 38).

[401] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. "The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains." In: *IEEE Signal Processing Magazine* 30.3 (2013), pp. 83–98 (cit. pp. 60, 65, 76).

[402] R. Sicre, Y. Avrithis, E. Kijak, and F. Jurie. "Unsupervised part learning for visual recognition." In: *CVPR*. 2017 (cit. p. 132).

[403] R. Sicre, J. Rabin, Y. Avrithis, T. Furon, F. Jurie, and E. Kijak. "Automatic discovery of discriminative parts as a quadratic assignment problem." In: *ICCV Workshops*. 2017 (cit. p. 131).

[404] O. Siméoni, M. Budnik, Y. Avrithis, and G. Gravier. "Rethinking deep active learning: Using unlabeled data at model training." In: *arXiv preprint arXiv:1911.08177* (2019) (cit. pp. 132, 148).

[405] Oriane Siméoni, Yannis Avrithis, and Ondrej Chum. "Local Features and Visual Words Emerge in Activations." In: *CVPR*. 2019 (cit. pp. vii, 6, 56, 62, 63, 81, 82, 87, 134, 136, 139, 140, 146, 149).

[406] Oriane Simeoni, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. "Unsupervised object discovery for instance recognition." In: *WACV*. 2018 (cit. pp. vii, 6, 56, 63, 70, 87, 88, 92, 137, 140, 147).

[407] Oriane Siméoni, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. "Graph-based Particular Object Discovery." In: *Machine Vision and Applications* 30.2 (Mar. 2019), pp. 243–254 (cit. pp. vii, 6, 56, 63, 70, 87, 88, 92, 137, 140, 144).

[408] I. Simon, N. Snavely, and S.M Seitz. "Scene Summarization for Online Image Collections." In: *ICCV*. 2007 (cit. pp. 19, 47, 49).

[409] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." In: *ICLR* (2014) (cit. pp. 57, 78, 82, 85, 92, 103, 108).

[410] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, and Francesc Moreno-Noguer. "Fracking deep convolutional image descriptors." In: *arXiv*. 2014 (cit. p. 105).

[411] Chawin Sitawarin and David Wagner. "Defending Against Adversarial Examples with K-Nearest Neighbor." In: *arXiv preprint arXiv:1906.09525* (2019) (cit. p. 150).

[412] J. Sivic, B.C. Russell, A. Efros, A. Zisserman, and W.T Freeman. "Discovering Object Categories in Image Collections." In: *ICCV*. 2005 (cit. p. 63).

[413] J. Sivic and A. Zisserman. "Video Google: A Text Retrieval Approach to Object Matching in videos." In: *ICCV*. 2003 (cit. pp. 13, 16, 27, 31, 34, 37, 81, 87, 95).

[414] N. Snavely, S.M. Seitz, and R. Szeliski. "Photo Tourism: Exploring Photo Collections in 3D." In: *Computer Graphics and Interactive Techniques*. 2006, pp. 835–846 (cit. pp. 19, 45).

[415] Jake Snell, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning." In: *NIPS*. 2017 (cit. pp. 96, 100, 116, 117, 119).

[416] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. "Deep Metric Learning via Facility Location." In: *CVPR*. 2017 (cit. p. 107).

[417] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. "Striving for simplicity: The all convolutional net." In: *arXiv preprint arXiv:1412.6806* (2014) (cit. p. 135).

[418] U. Steinhoff, D. Omercevic, R. Perko, B. Schiele, and A. Leonardis. "How Computer Vision Can Help in Outdoor Positioning." In: *European Conference on Ambient Intelligence*. 2007 (cit. p. 19).

[419] David Stutz, Matthias Hein, and Bernt Schiele. "Disentangling Adversarial Robustness and Generalization." In: *CVPR*. 2019 (cit. p. 102).

[420] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. "End-To-End Memory Networks." In: *NIPS*. 2015 (cit. p. 144).

[421] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era." In: *ICCV*. 2017 (cit. pp. 96, 143, 144).

[422] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. "Inception-v4, Inception-ResNet and the impact of residual connections on learning." In: *AAAI*. 2017 (cit. p. 57).

[423] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In: *CVPR*. 2015 (cit. pp. 57, 103, 108).

[424] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In: *CVPR*. 2016 (cit. p. 125).

[425] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks." In: *arXiv:1312.6199* (2013) (cit. pp. 96, 101, 121, 123).

[426] Antti Tarvainen and Harri Valpola. "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results." In: *NIPS*. 2017 (cit. pp. 96, 99, 109, 113, 114).

[427] Marvin Teichmann, Andre Araujo, Menglong Zhu, and Jack Sim. "Detect-to-retrieve: Efficient regional aggregation for image search." In: *CVPR*. 2019 (cit. pp. 18, 137, 139).

[428] Matus Telgarsky. "Representation benefits of deep feedforward networks." In: *arXiv preprint arXiv:1509.08101* (2015) (cit. p. 57).

[429] J. Tenenbaum. "Mapping a manifold of perceptual observations." In: *NIPS*. 1997 (cit. p. 97).

[430] J. Tenenbaum, V. de Silva, and J. Langford. "A Global Geometric Framework for Nonlinear Dimensionality Reduction." In: *Science* 290.5500 (2000), pp. 2319–2323 (cit. p. 103).

[431] Sebastian Thrun. "Lifelong learning algorithms." In: *Learning to learn*. 1998 (cit. p. 115).

[432] M. Tipping and B. Schölkopf. "A Kernel Approach for Vector Quantization with Guaranteed Distortion Bounds." In: *Artificial Intelligence and Statistics*. 2001, pp. 129–134 (cit. pp. 19, 45, 46).

[433] G. Tolias and Y. Avrithis. "Speeded-Up, Relaxed Spatial Matching." In: *ICCV*. 2011 (cit. pp. vii, 4, 13, 17, 20, 25, 27, 32, 81, 87, 130, 131, 139, 140).

[434] G. Tolias, Y. Kalantidis, and Y. Avrithis. "SymCity: Feature Selection by Symmetry for Large Scale Image Retrieval." In: *ACM Multimedia*. 2012 (cit. pp. 131, 137).

[435] G. Tolias, Y. Kalantidis, Y. Avrithis, and S. Kollias. "Towards large-scale geometry indexing by feature selection." In: *CVIU* 120 (2014), pp. 31–45 (cit. pp. 131, 137).

[436] Giorgos Tolias, Yannis Avrithis, and Hervé Jégou. "To Aggregate or not to Aggregate: Selective Match Kernels for Image Search." In: *ICCV*. 2013 (cit. pp. vii, 4, 13, 18, 20, 33, 38, 39, 58, 59, 74, 81, 134, 139, 140).

[437] Giorgos Tolias, Yannis Avrithis, and Hervé Jégou. "Image search with selective match kernels: aggregation across single and multiple images." In: *IJCV* 116.3 (2016), pp. 247–261 (cit. pp. vii, 4, 13, 18, 20, 33, 38, 85, 86, 134, 139, 140).

[438] Giorgos Tolias and Herve Jegou. "Visual Query Expansion with or Without Geometry: Refining Local Descriptors By Feature Aggregation." In: *Pattern Recognition* (2014) (cit. p. 73).

[439] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. "Particular object retrieval with integral max-pooling of CNN activations." In: *ICLR* (2016) (cit. pp. 58, 59, 62, 73, 75, 78, 79, 81–83, 85, 87, 90, 91, 107, 108).

[440] C. Tomasi and T. Kanade. *Detection and Tracking of Point Features*. Tech. rep. 1991 (cit. p. 14).

[441] Hanghang Tong, Christos Faloutsos, and Jia Y Pan. "Fast Random Walk with Restart and Its Applications." In: *ICDM*. 2006, pp. 613–622 (cit. p. 78).

[442] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. "The Space of Transferable Adversarial Examples." In: *arXiv:1704.03453* (2017) (cit. p. 102).

[443] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*. SIAM, 1997 (cit. pp. 76, 77).

[444] Anne M Treisman and Garry Gelade. "A Feature-Integration Theory of Attention." In: *Cognitive Psychology* 12.1 (1980), pp. 97–136 (cit. p. 62).

[445] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. "Robustness may be at odds with accuracy." In: *arXiv preprint arXiv:1805.12152* (2018) (cit. p. 102).

[446] P. Turcot and D. G. Lowe. "Better matching with fewer features: The selection of useful features in large database recognition problems." In: *ICCV Workshops*. 2009 (cit. pp. 38, 63).

[447] Mark R Turner. "Texture Discrimination By Gabor Functions." In: *Biological Cybernetics* 55.2 (1986), pp. 71–82 (cit. pp. 13, 15).

[448] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. "Deep domain confusion: Maximizing for domain invariance." In: *arXiv preprint arXiv:1412.3474* (2014) (cit. p. 96).

[449] Evgeniya Ustinova and Victor Lempitsky. "Learning deep embeddings with histogram loss." In: *NIPS*. 2016 (cit. pp. 100, 108, 138).

[450] Koen EA Van De Sande, Jasper RR Uijlings, Theo Gevers, and Arnold WM Smeulders. "Segmentation As Selective Search for Object Recognition." In: *ICCV*. 2011 (cit. pp. 58, 62, 130).

[451] Vladimir Vapnik. *Statistical learning theory*. 1998 (cit. p. 99).

[452] Vladimir Vapnik and Alexey Chervonenkis. *Theory of pattern recognition*. 1974 (cit. p. 99).

[453] C. Varytimidis, K. Rapantzikos, and Y. Avrithis. "WαSH: Weighted α-Shapes for Local Feature Detection." In: *ECCV*. 2012 (cit. p. 130).

[454] C. Varytimidis, K. Rapantzikos, Y. Avrithis, and S. Kollias. "α-shapes for local feature detection." In: *Pattern Recognition* 50.1 (Feb. 2016), pp. 56–73 (cit. p. 130).

[455] A. Vedaldi and B. Fulkerson. *VLFeat: An Open and Portable Library of Computer Vision Algorithms*. http://www.vlfeat.org/. 2008 (cit. p. 85).

[456] Yannick Verdie, Kwang Yi, Pascal Fua, and Vincent Lepetit. "TILDE: a temporally invariant learned detector." In: *CVPR*. 2015 (cit. p. 61).

[457] Paul Vernaza and Manmohan Chandraker. "Learning Random-Walk Label Propagation for Weakly-Supervised Semantic Segmentation." In: *CVPR*. 2017 (cit. p. 123).

[458] Sebastiano Vigna. "Spectral Ranking." In: *arXiv preprint arXiv:-0912.0238* (2009) (cit. pp. 60, 65, 68, 91).

[459] Ricardo Vilalta and Youssef Drissi. "A perspective view and survey of meta-learning." In: *Artificial Intelligence Review* 18.2 (2002), pp. 77–95 (cit. p. 100).

[460] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. "Extracting and Composing Robust Features with Denoising Autoencoders." In: *ICML*. 2008 (cit. p. 95).

[461] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. "Matching networks for one shot learning." In: *NIPS*. 2016 (cit. pp. 96, 100, 113, 115, 117, 120).

[462] Paul Viola and Michael Jones. "Rapid object detection using a boosted cascade of simple features." In: *CVPR*. 2001 (cit. pp. 14, 143).

[463] Nisheeth K Vishnoi. "Laplacian Solvers and Their Algorithmic Applications." In: *Theoretical Computer Science* 8.1-2 (2012), pp. 1–141 (cit. p. 76).

[464] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. *The Caltech-UCSD Birds-200-2011 dataset*. Tech. rep. California Institute of Technology, 2011 (cit. p. 106).

[465] Georg Waltner, Michael Opitz, and Horst Bischof. "Bacon: Building a Classifier From Only N Samples." In: *Computer Vision Winter Workshop* (2016) (cit. p. 100).

[466] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. "NormFace: L2 hypersphere embedding for face verification." In: *ACM Multimedia*. 2017 (cit. p. 101).

[467] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. "CosFace: Large margin cosine loss for deep face recognition." In: *CVPR*. 2018 (cit. p. 138).

[468] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. "Learning Fine-Grained Image Similarity with Deep Ranking." In: *CVPR*. 2014 (cit. pp. 59, 98, 103–105).

[469] Shuang Wang and Shuqiang Jiang. "INSTRE: a new benchmark for instance-level object retrieval and recognition." In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 11 (2015), p. 37 (cit. pp. 79, 92, 106).

[470] Xiaolong Wang and Abhinav Gupta. "Unsupervised learning of visual representations using videos." In: *CVPR*. 2015 (cit. pp. 96, 98, 103, 104, 144, 147).

[471] Xiaolong Wang, Kaiming He, and Abhinav Gupta. "Transitive invariance for selfsupervised visual representation learning." In: *ICCV*. 2017 (cit. pp. 103, 104, 109).

[472] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. "Low-Shot Learning from Imaginary Data." In: *CVPR* (2018) (cit. p. 101).

[473] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. "Learning to model the tail." In: *NIPS*. 2017 (cit. pp. 115, 146).

[474] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. "Multi-Similarity Loss with General Pair Weighting for Deep Metric Learning." In: *CVPR*. 2019 (cit. p. 138).

[475] Yaming Wang, Jonghyun Choi, Vlad Morariu, and Larry S Davis. "Mining discriminative triplets of patches for fine-grained classification." In: *CVPR*. 2016 (cit. pp. 103, 106).

[476] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. "Dynamic graph CNN for learning on point clouds." In: *ACM Transactions on Graphics (TOG)* 38.5 (2019), pp. 1–12 (cit. pp. 147, 149).

[477] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. "Distance Metric Learning for Large Margin Nearest Neighbor Classification." In: *NIPS*. 2006 (cit. p. 98).

[478] Philippe Weinzaepfel, Hervé Jégou, and Patrick Pérez. "Reconstructing an Image From Its Local Descriptors." In: *CVPR*. 2011 (cit. p. 101).

[479] Robert Edwin Wengert. "A simple automatic derivative evaluation program." In: *Communications of the ACM* 7.8 (1964), pp. 463–464 (cit. p. 56).

[480] Paul John Werbos. "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences." PhD thesis. Applied Mathematics, Harvard University, MA, 1974 (cit. p. 56).

[481] Davis Wertheimer and Bharath Hariharan. "Few-Shot Learning With Localization in Realistic Settings." In: *CVPR*. 2019 (cit. p. 140).

[482] Jason Weston, Frédéric Ratle, and Ronan Collobert. "Deep Learning via Semi-Supervised Embedding." In: *ICML*. 2008 (cit. pp. 96, 99, 109, 111).

[483] Tobias Weyand and Bastian Leibe. "Discovering Favorite Views of Popular Places with Iconoid Shift." In: *ICCV*. 2011 (cit. p. 149).

[484] A.P Witkin. "Scale-Space Filtering." In: *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*. Morgan Kaufmann, 1987 (cit. pp. 13, 14).

[485] Ian H Witten, Ian H Witten, Alistair Moffat, Timothy C Bell, Timothy C Bell, and Timothy C Bell. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, 1999 (cit. p. 16).

[486] Rafi Witten and Emmanuel Candes. "Randomized Algorithms for Low-Rank Matrix Factorizations: Sharp Performance Bounds." In: *arXiv preprint arXiv:1308.5697* (2013) (cit. p. 77).

[487] Paul Wohlhart, Martin Kostinger, Michael Donoser, Peter M Roth, and Horst Bischof. "Optimizing 1-Nearest Prototype Classifiers." In: *CVPR*. 2013 (cit. p. 138).

[488] Shu-Fai Wong and Roberto Cipolla. "Extracting Spatiotemporal Interest Points Using Global Information." In: *ICCV*. 2007 (cit. p. 130).

[489] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. "Sampling matters in deep embedding learning." In: *ICCV*. 2017 (cit. pp. 98, 108, 138, 145).

[490] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. "Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination." In: *CVPR* (2018) (cit. pp. 96, 109, 138, 145).

[491] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. "Distance Metric Learning with Application to Clustering with Side-Information." In: *NIPS*. 2003 (cit. pp. 97, 129).

[492] I Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. "Billion-scale semi-supervised learning for image classification." In: *arXiv preprint arXiv:1905.00546* (2019) (cit. p. 144).

[493] Ziang Yan, Jian Liang, Weishen Pan, Jin Li, and Changshui Zhang. "Weakly- and semi-supervised object detection with expectation-maximization algorithm." In: *arXiv preprint arXiv:1702.08740* (2017) (cit. p. 132).

[494] Liu Yang and Rong Jin. *Distance Metric Learning: A Comprehensive Survey*. Tech. rep. 2006 (cit. p. 98).

[495] Z. Yang, M. Shi, Y. Avrithis, C. Xu, and V. Ferrari. "Training Object Detectors from Few Weakly-Labeled and Many Unlabeled Images." In: *arXiv preprint arXiv:1912.00384* (2019) (cit. pp. 132, 146).

[496] Alfred L Yarbus. *Eye movements and vision*. Plenum Press, 1967 (cit. p. 62).

[497] Mang Ye, Xu Zhang, Pong C. Yuen, and Shih-Fu Chang. "Unsupervised Embedding Learning via Invariant and Spreading Instance Feature." In: *CVPR*. 2019 (cit. pp. 138, 139, 145).

[498] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. "LIFT: Learned invariant feature transform." In: *ECCV*. 2016 (cit. pp. 61, 81, 82, 146).

[499] Kenneth Yip and Gerald Jay Sussman. "Sparse Representations for Fast, One-Shot Learning." In: *AAAI*. 1997 (cit. pp. 100, 143).

[500] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. "Lifelong Learning with Dynamically Expandable Networks." In: *ICLR*. 2018 (cit. pp. 96, 144, 150).

[501] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. "How transferable are features in deep neural networks?" In: *NIPS*. 2014 (cit. pp. 96, 144).

[502] Yang You, Igor Gitman, and Boris Ginsburg. "Large batch training of convolutional networks." In: *arXiv preprint arXiv:1708.03888* (2017) (cit. p. 143).

[503] Gale Young and Alston S Householder. "Discussion of a set of points in terms of their mutual distances." In: *Psychometrika* 3.1 (1938), pp. 19–22 (cit. p. 97).

[504] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. "Deep sets." In: *NIPS*. 2017 (cit. p. 136).

[505] H. Zhang, Y. Avrithis, T. Furon, and L. Amsaleg. "Smooth Adversarial Examples." In: *arXiv preprint arXiv:1903.11862* (Mar. 2019) (cit. pp. vii, 7, 71, 97, 102, 121, 125, 126, 140, 141).

[506] H. Zhang, Y. Avrithis, T. Furon, and L. Amsaleg. "Walking on the Edge: Fast, Low-Distortion Adversarial Examples." In: *arXiv preprint arXiv:1912.02153* (2019) (cit. pp. 133, 140, 141).

[507] Hongguang Zhang, Jing Zhang, and Piotr Koniusz. "Few-Shot Learning via Saliency-Guided Hallucination of Samples." In: *CVPR*. 2019 (cit. p. 140).

[508] Jian Zhang, Chenglong Zhao, Bingbing Ni, Minghao Xu, and Xiaokang Yang. "Variational Few-Shot Learning." In: *ICCV*. 2019 (cit. p. 150).

[509] Shaoting Zhang, Ming Yang, Timothee Cour, Kai Yu, and Dimitris N Metaxas. "Query specific fusion for image retrieval." In: *ECCV*. 2012 (cit. p. 74).

[510] W. Zhang and J. Kosecka. "Image Based Localization in Urban Environments." In: *International Symposium on 3D Data Processing, Visualization and Transmission*. 2006 (cit. p. 19).

[511] Xu Zhang, Felix X Yu, Sanjiv Kumar, and Shih-Fu Chang. "Learning spread-out local feature descriptors." In: *ICCV*. 2017 (cit. p. 139).

[512] Xu Zhang, Felix Xinnan Yu, Svebor Karaman, Wei Zhang, and Shih-Fu Chang. "Heated-up softmax embedding." In: *arXiv preprint arXiv:1809.04157* (2018) (cit. pp. 96, 101, 138).

[513] Zhengli Zhao, Dheeru Dua, and Sameer Singh. "Generating Natural Adversarial Examples." In: *ICLR*. 2018 (cit. p. 121).

[514] Yantao Zheng, Ming Zhao, Yang Song, Hartwig Adam, Ulrich Buddemeier, Alessandro Bissacco, Fernando Brucher, Tat-Seng Chua, and Hartmut Neven. "Tour the World: Building a Web-Scale Landmark Recognition Engine." In: *CVPR*. 2009 (cit. pp. 45–47).

[515] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. "Learning Deep Features for Discriminative Localization." In: *CVPR*. 2016 (cit. p. 118).

[516] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. "Learning with local and global consistency." In: *NIPS*. 2003 (cit. pp. 66, 68, 69, 71, 74, 99, 102, 109, 111).

[517] Dengyong Zhou, Jason Weston, Arthur Gretton, Olivier Bousquet, and Bernhard Schölkopf. "Ranking on Data Manifolds." In: *NIPS*. 2003 (cit. pp. 60, 66, 68, 70, 71, 73, 74, 123, 124).

[518] Wen Zhou, Xin Hou, Yongjun Chen, Mengyun Tang, Xiangqi Huang, Xiang Gan, and Yong Yang. "Transferable Adversarial Perturbations." In: *ECCV*. 2018 (cit. p. 102).

[519] Yanzhao Zhou, Yi Zhu, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. "Weakly Supervised Instance Segmentation Using Class Peak Response." In: *CVPR*. 2018 (cit. pp. 96, 144).

[520] Yin Zhou and Oncel Tuzel. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection." In: *CVPR*. 2018 (cit. p. 147).

[521] Xiaojin Zhu and Zoubin Ghahramani. *Learning From Labeled and Unlabeled Data with Label Propagation*. Tech. rep. 2002 (cit. pp. 69, 99).

[522] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions." In: *ICML*. 2003 (cit. pp. 69, 99, 109, 123).

[523] Zhuotun Zhu, Lingxi Xie, and Alan L Yuille. "Object Recognition with and without Objects." In: *arXiv preprint arXiv:1611.06596* (2016) (cit. p. 144).

[524] C. Lawrence Zitnick and Piotr Dollar. "Edge Boxes: Locating Object Proposals From Edges." In: *ECCV*. Vol. 8693. Springer, 2014 (cit. pp. 58, 62, 130).

[525] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. "Learning transferable architectures for scalable image recognition." In: *CVPR*. 2018 (cit. p. 96).

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/