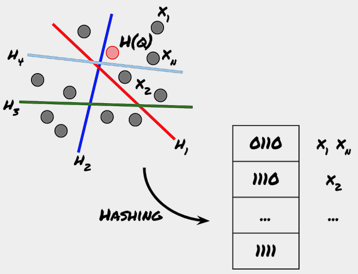
Computer Aided Medical Procedures
Prof. Dr. Nassir Navab



Dissertation

Learning to Hash for Large-Scale Medical Image Retrieval

Sailesh Conjeti

Fakultät für Informatik
Technische Universität München

Technische Universität München
Fakultät für Informatik
Lehrstuhl für Informatikanwendungen in der Medizin

Learning to Hash for Large-Scale Medical Image Retrieval

Sailesh Conjeti

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzende(r): Prof. Dr. Julien Gagneur

Prüfer der Dissertation: Prof. Dr. Nassir Navab

Prof. William Wells III, Ph.D.

Die Dissertation wurde am 13.03.2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 19.06.2018 angenommen.

Sailesh Conjeti
Learning to Hash for Large-Scale Medical Image Retrieval
Dissertation, Version 1.0

Technische Universität München
Fakultät für Informatik
Lehrstuhl für Informatikanwendungen in der Medizin
Boltzmannstraße 3
85748 and Garching bei München

Abstract

The task of similarity search refers to fetching an item that is closest to the query item from a reference database under the notion of some distance measure. In critical applications of large-scale search and pattern matching, exhaustive comparison is often not possible due to prohibitive computational complexity and memory overheads. Towards mitigating this, hashing has been adopted as a popular approach for performing computationally efficient approximate nearest neighbor search. It is an approach of encoding data items into sequence of bits, called hash codes, such that the nearest neighbor search in the coding space is efficient and accurate. With explosive growth in big data in medical imaging, there is a particular need for designing efficient indexing and search methods that can be leveraged for pattern exploration and knowledge discovery.

Towards learning optimized hashing functions for efficient indexing, we propose hashing forests (HF), which are an ensemble of code-efficient trees for recursive partitioning of the feature space. We propose two variants of HF, namely unsupervised HF (uHF) and metric HF (mHF) that are targeted at distance approximation and semantic-preserving hashing scenarios respectively. In uHF, trees are trained to parse the feature space into balanced well-clustered subspaces, whereas in mHF, the trees use localized metric learning to parse the space into semantically similar local neighborhoods. The versatility and efficacy of both these variants is demonstrated for the challenging task of pattern exploration within highly heterogeneous neuron databases. Extending to medical images, we leverage deep learning for end-to-end learning of convolutional neural networks for simultaneous representation learning and hashing. For enforcing code consistency within such networks, we design optimization objectives inspired by neighborhood component analysis (NCA) for minimizing the similarity weighted Hamming distance, with additional losses and regularizations such as bit-balance, bit-uncorrelation and quantization loss to improve hash code quality. We demonstrate the ability of such networks to perform semantics-preserving similarity search on a large scale chest X-ray image database with co-occurring disease patterns. We also investigate extension to multiple instance (MI) retrieval through the introduction of the MI pooling layer and robustness within NCA for improved learning. This was demonstrated on large-scale databases of breast mammography and histology for carcinoma assessment.

In conclusion, this thesis explores aspects of code-consistent training of hashing forests and deep learning models for end-to-end learning of hash codes and demonstrates that such hashing models can be leveraged to perform efficient and accurate large-scale content-based medical image retrieval.

Zusammenfassung

Die Aufgabe der Ähnlichkeitssuche besteht darin, ein Element aus einer Referenzdatenbank auszuwählen, das am nähesten des Suchelementes bezüglich einer Distanzmetrik ist. In kritischen Such- oder Mustererkennungsanwendungen ist es meistens wegen Speicher- und Rechen-Overhead nicht möglich, eine exhaustive Suche durchzuführen. Hashing ist eine Methode, um die Nearest-Neighbor-Suche laufzeit-effizient zu lösen, in der man das Element in einer Bitsequenz, genannt Hashcode, kodiert, so dass die Vergleiche im Kodierungsraum effizient und präzise durchgeführt werden können. Der wachsenden Datenmenge in der Bildgebung zugrunde müssen Algorithmen für effizientes Indizieren und Suchen entwickelt werden, die für Mustererkennung und Wissenentdeckung genutzt werden.

Um optimierte Hashing Funktionen für effizientes Indizieren zu lernen, schlagen wir Hash-Forests (HF) vor, die ein Ensemble von code-effizienten Bäume für rekursives Partitionieren des Feature-Raums sind. Wir schlagen zwei Arten von HF vor, nämlich unüberwachtes HF (uHF) und metrisches HF (mHF), die der Distanzapproximierung bzw. dem semantik-bewahrenden Hashing gewidmet sind. Die trees im uHF werden zur Zerlegung des Feature-Raums in ausgeglichene gut-gruppierte Unterräume traininert, während die mHF Bäume lokales metrisches Lernen verwenden, um den Raum in semantisch ähnliche lokale Nachbarschaften zu zerlegen. Die Vielseitigkeit und Wirksamkeit dieser beiden Varianten wird anhand anspruchsvoller Mustererkennung in sehr heterogenen Neuronendatenbanken beweist. Auf Bildgebung erweiternd, wir setzen Deep Learning in Form von Convolutional Neural Networks (CNNs) für das simultanes Ende-zu-Ende-Lernen von automatischen Features und Hashing ein. Zur Durchsetzung von Code-Konsistenz in CNNs entwerfen wir Optimierungsziele, die von der Nachbarschaftskomponentenanalyse (NCA) inspiriert sind. Um die Qualität der Codes zu verbessern, werden die Ähnlichkeits-gewichteten Hamming-Distanz minimiert und andere Ziele und Regularisierungen einführt, wie Bit-Balance, Bit-Unkorrelation und Quantisierungsverlust. Wir zeigen die Fähigkeit solcher CNNs, semantik-erhaltende Ähnlichkeitssuche in einer Large-Scale Röntgen-Brust Bilddatenbank mit zusammen auftretenden Krankheitsmuster auszuführen. Wir untersuchen sowohl die Erweiterung auf Multiple Instance (MI) Retrieval durch die Einführung der MI-Pooling-Layer als auch die Robustheit innerhalb NCA für verbessertes Lernen auf Large-Scale Datenbanken der Brust Mammographie und Histologie zur Karzinom-Einschätzung.

Zusammenfassend untersucht diese Dissertation Aspekte des code-konsistenten Trainings von Hash-Forests und Deep Learning Modellen für das Ende-zu-Ende-Lernen von Hashwerte und zeigt, dass diese Hashing Modelle dazu genutzt werden können, effizientes und präzises Large-Scale medizinisches inhaltsbasiertes Image Retrieval durchzuführen.

Acknowledgments

Doing research at CAMP has been one of the most enriching and rewarding periods of my life. I am forever grateful to my advisor Nassir Navab for his boundless enthusiasm and constant belief in me, which allowed me to take up a niche topic and hammer it hard. I would like to whole-heartedly thank Amin Katouzian for his constant support and mentorship. In the initial stages of my doctoral studies, Amin helped shape my perspective and more importantly guided me on the path of pursing research in the direction that excited me. I learned a lot from his way of approaching problems and am very grateful for the constant support he offered through this journey. I would like to thank Prof. Dr. Julien Gagneur and Prof. Dr. William Wells III for enthusiastically reviewing my thesis and for the very insightful discussion during my oral examination.

Both Abhijit Guha Roy and Magda Paschali deserve a special shout-out. They were constant companions over this journey. Starting with forests and transitioning into deep learning was done together with Abhijit and I am ever grateful for his steadfast friendship and collaboration. Together with Magda, I dared to venture into newer and exciting facets of deep learning, which was an incredible experience. I am very proud of and grateful to Sepideh Mesbah, Abhijit Guha Roy, Anees Kazi, Magda Paschali, Deepa Gunasekhar, Muneer Ahmad Dedmari, Fernando Navarro, Andrei Costinescu, Phalgun Chintala, Amit Pandita and Santiago Estrada for choosing to do their theses with me and I will forever cherish the experience of working with them. Together with them, I was able to explore a large spectrum of topics and develop insights into various aspects of medical imaging way beyond the scope and confines of my core PhD research.

Having insightful discussions and working with colleagues at CAMP including Shadi Albarquoni, Mehmet Yigitsoy, Tingying Peng, Maximilian Baust, Federico Tombari, Diana Mateus, Huseyin Coskun, Christian Rupprecht, Loic Peter, Sebastian Pölstrel, Nicola Reike, Wadim Kehl, Fausto Milletari, Paul Huang, Hemal Naik, Robert DiPietro and many other CAMPers was an incredible experience. I am also very grateful to Martina Hilla for her help and support with all the administrative demands. My journey of doing a PhD has been a highly collaborative adventure and I would like to acknowledge the contributions of Phillip Rautenberg, Debdoot Sheet, Shaoting Zhang, Arcot Sowmya, Gustavo Caneiro, Christian Wachinger, Martin Reuter, Stephane Carlier, Phani Krishna Karri, Sebasitian Pölstrel, Lichao Wang and other wonderful collaborators. I am ever grateful for their support. I had already moved to Bonn during the time of my defense. I would like to thank my colleagues Martin Reuter, Kersten Diers, Santiago Estrada, Amit Pandita, Leonie Herschel and Muneer Ahmad Dedmari for supporting me through the last few months.

Finally, I would like to express my deepest gratitude towards my parents, Srilakshmi and Sudhir, my dear sister Supraja and my partner for life Sindhuja for always believing in me and keeping my morale high through the roller-coaster ride of a PhD.

Contents

Part I

Introduction

Introduction

1

> *Medicine is a science of uncertainty and an art of probability.*
>
> — **William Osler**
> (Founding Professor, Johns Hopkins Hospital)

1.1 Motivation

Content-based Image Retrieval (CBIR) is an image search technique that is driven by quantifiable features extracted directly from the images as the search criterion [181]. In contrast to depending on manually assigned search keywords, these features are often automatically / semi-automatically extracted from the images, thereby making the retrieval system more objective and quantifiable. Essentially, a CBIR system measures image similarity based on the underlying visual components and is ideal for large-scale mining of image repositories. The goal of such a system is to answer two primal questions:

- *What ?*: Retrieve a set of images that are similar to the query image
- *Why ?* Explain why the retrieved results are similar to the query in an objective, non-qualitative manner.

In the recent past, with the deluge of digital images (e.g. ImageNet [49], MS-COCO [115], PASCAL VOC [59] etc.), CBIR has moved towards Big Data, with a new generation of algorithms targeting retrieval at a massive scale. Towards this fast and efficient indexing and compression methods have been proposed including product quantization techniques [90], tree-based indexing structures [88, 97] and hashing [193] to name a few. The design elements for a CBIR system the scale well are closely related to the concepts of efficiency and accuracy of the system. These include:

Defining similarity: CBIR system requires application-specific definition of inter-image similarity such that the extracted representations preserve the relevant semantics and factor out irrelevant information.

Efficiency: The CBIR system has to efficiently organize the representations into indices such that search and retrieval from large repositories is time-efficient and has a low-computational footprint.

Similarity search, also known as *nearest neighbor search* aims at finding items within a database $\mathcal{X}$ that are nearest to a query q, under the definition of some distance measure. For large-scale databases, the distance computation in a linear fashion (*i.e.* exhaustive search) is expensive and is often computationally infeasible due to associated linear search-time complexity $\mathcal{O}(|\mathcal{X}|)$. Besides limited scalability of such an approach, since semantics-preserving representations are often very high-dimensional, the *curse of dimensionality* and associated storage constraints become important bottlenecks to consider and mitigate [16]. Often, retrieving a set of *Approximate Nearest Neighbors* (ANN) is sufficient for practical applications. Hence, methods that achieve sub-linear $o(|\mathcal{X}|)$, logarithmic $\mathcal{O}(|\mathcal{X}|)$ or even constant $\mathcal{O}(1)$ query time is desired for retrieving ANNs.

With increasing volume of modern image repositories and high feature dimensionality of the descriptors, the question of how to efficiently mine them is important to address. In many cases, it is not economical to exhaustive compare the query to every element in the database. To achieve real-time retrieval, efficient indexing schemes are needed that store and partition the database, such that the data can be accessed and explored quickly by filtering out irrelevant data. This has given rise to approximate search schemes which aim at ranking images within a dataset in accordance to their relevance to the query.

Tree-based indexing approaches employ hierarchical database structures that recursively partition the feature space and inverted files to improve retrieval efficiency. Prior art of such approaches includes KD tree [131], ball tree [18], metric tree [210], vantage point tree [139], vocabulary tree [140] *etc.* to name a few. However, these approaches have a significant overhead of storage and are reported to dramatically degrade in performance while handling high dimensional data. In contrast to these methods, hashing methods repeatedly partition the entire dataset and derive a single hash 'bit' from each partitioning. We extensively discuss and present hashing based approximate nearest neighbor search in this chapter. Other popular approaches towards handling scalability in large-scale retrieval include feature compression (or dimensionality reduction) [27], re-ranking [138] and use of high-performance computing environments [113].

1.2 Outline

This thesis, in its entirety, is dedicated to developing advanced hashing models and demonstrating their usability for the task of CBIR in medical image databases. This thesis is structured in four parts. In Part I, **Chapter 1**, the essentials of hashing and content based image retrieval are introduced, including a detailed overview of learning and optimizing hash functions, brief overview of the state-of-the art in hashing and subsequently evaluating such functions. The chapter also presents a brief overview of image retrieval specific to medical imaging discussing the challenges associated with it and presents the prior art methods developed to address the same. This chapter aims to serve as a basis for the subsequent chapters and lays the premise for the contributions presented in the thesis.

Part II contains the scientific contributions of this thesis, which are outlined as follows:

- **Chapter 2** introduces a machine learning model based on random forests for unsupervised hashing, termed as *Hashing Forests*. This model is an ensemble of independently trained hashing trees that recursively parse the feature space into balanced clusters and encode them with binary codes. We introduce a novel cluster validity and balanced partitioning based cost-function for learning the hash trees and demonstrate their applicability to a challenging scenario of neuron image retrieval.

- **Chapter 3** introduces supervision into the training of hashing forests, termed as *metric Hashing Forests* (mHF). Key elements are the introduction of local metric learning and a novel cost function inspired by code-consistent hashing to learn trees that kernelize and parse the feature space recursively into semantically similar local neighborhoods that are encoded with binary codes. mHF is extensively validated and compared against state of the art hashing and metric learning methods for the task of similarity-preserving neuron image retrieval.

- **Chapter 4** delves into leveraging deep learning for the task of end-to-end learning of hash codes and we introduce a novel deep architecture, termed *Deep Residual Hashing* (DRH). To enforce code-consistency, optimization based on neighborhood component analysis (NCA) was performed with additional losses and regularizations such as bit balance, bit orthogonality and quantization being introduced to improve hash code quality. DRH was extensively validated on the task of similarity-preserving retrieval within a highly heterogeneous Chest X-ray database with a significant fraction of co-morbid samples.

- **Chapter 5** presents extensions of deep hashing concepts discussed in Chapter 4 into the domain of multiple instance image retrieval and introduces a novel hashing paradigm termed as *Robust Multiple Instance Hashing* (RMIH). Given a bag of images, RMIH generates an hash code at the bag-level preserving the semantics associated with such a bag. This is achieved with the introduction of the multiple instance pooling layer to aggregate information selectively from across the bag. To learn in the presence of label noise, we extended NCA loss discussed in Chapter 4 into robust-NCA with the introduction of robust statistics. RMIH was extensively validated for retrieval on two large-scale datasets related to assessment of breast carcinoma (mammography and histology).

Substantial parts of this thesis have already been published, and the respective publications are clearly indicated at the beginning of each chapter. Although the presented work can be considered my own if not explicitly declared otherwise, the usage of the first-person plural form indicates that many efforts where only possible as a team. Finally, Part III, Chapter 6, concludes the thesis and outlines potential directions of future work. The appendix (Part IV) contains additional information about pattern exploration in neuron image databases, the list of awards and scholarships, abstracts of related publications not presented within this thesis, lists of publications, figures, tables, and the bibliography.

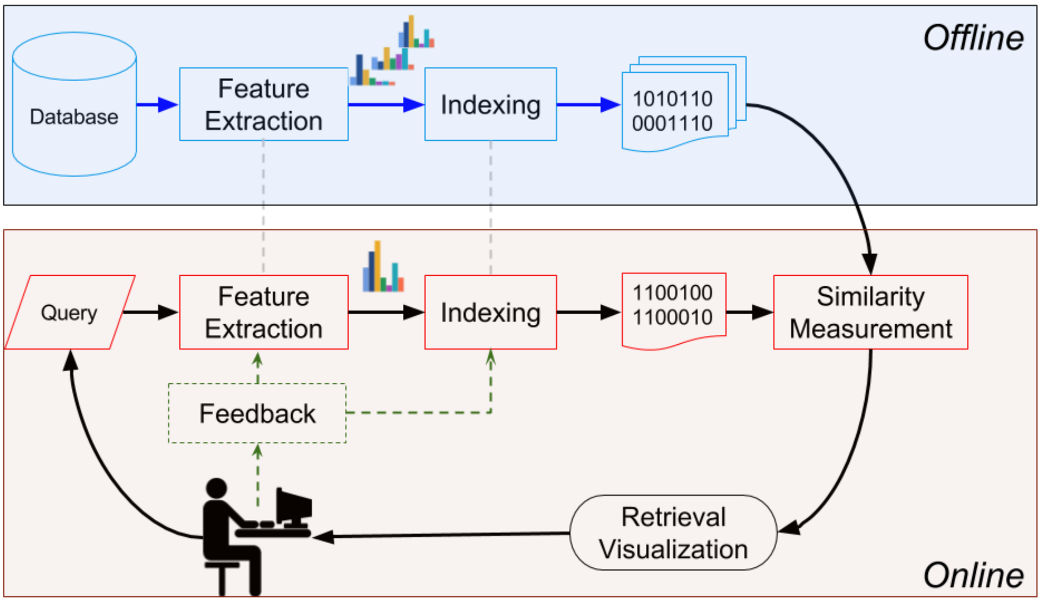


Fig. 1.1. *Schematic illustrating the overall process of Content based Image Retrieval using Hashing*: In the training phase (performed offline), give a reference database, with associated semantics in case of supervised retrieval, firstly we extract (or alternatively learn) features that encode these semantics. Post feature extraction, hash functions are learnt to parse the feature space into semantically similar neighborhoods which are indexed with a sequence of bits, called hash codes. The hash codes across the database is organized into a hash table which is light-weight and is used during retrieval. In the querying phase (performed online), the query item undergoes an identical feature extraction routine followed by index generation through a learnt hash function. The hash code of the query is compared against the hash table using computationally efficient inverse lookup and the database items are ranked. The top-ranked data items are presented to the user and any available feedback re-incorporated into in the query phase.

1.3 Essentials of Hashing

To make retrieval fast without compromising of the performance, generalized hashing based methods have been widely investigated. Hashing refers to the approach of mapping the query to the target in a database using quantized representations such that similar nearest neighbors can be searched and retrieved efficiently and accurately. This is usually achieved by transforming the data items into a low-dimensional representation, or equivalently compact codes consisting of a sequence of bits. Ideally, such a mapping has to preserve similarity in which similar items are mapped to closer hash codes than dissimilar counterparts. The distance computed on short codes is highly efficient and such encoding is scalable to large-scale databases as the codes are much smaller than the input features (or data items) thus resulting in the disk I/O cost reduction for cases where the original features are too large for the available memory [193, 195, 201]. For instance, 120 thousand chest X-ray images, such as Chest-XRay8 [198] (1000×1000, 72 DPI, 12-bit encoding) would cost around 55G bytes in storage space, but can be presented into 64-bits binary codes requiring only 7.2M bytes!

1.3.1 Exact *vs.* Approximate Nearest Neighbor Search

Given query $\mathbf{q}$, the goal of nearest neighbor search is to retrieve item NN($\mathbf{q}$) from within a database $\mathcal{X}$, such that

$$\text{NN}(\mathbf{q}) = \arg\ \text{min}_{\mathbf{x}\in\mathcal{X}} \text{dist}(\mathbf{x}, \mathbf{q}) \tag{1.1}$$

where $\text{dist}(\mathbf{x}, \mathbf{q})$ is the distance computed between $\mathbf{q}$ and $\mathbf{x}$. Typically, if $\mathcal{X}$ lies in d-dimensional space $\mathbb{R}^d$ the distance is induced by an l_s norm (usually $s = 2$), given by:

$$\|\mathbf{x} - \mathbf{q}\|_s = \left(\sum_{i=1}^{d} |x_i - q_i|^s\right)^{1/s} \tag{1.2}$$

For a fixed-radius (say R) nearest neighbor problem, the goal is to retrieve items $\mathcal{R}$ that satisfy $\mathcal{R} = \{\mathbf{x} \| \text{dist}(\mathbf{x}, \mathbf{q}) \le R, \mathbf{x} \in \mathcal{X}\}$. As discussed in the Sec. 1.1, the naïve solution based on linear scan is computationally expensive and for most practical applications, an approximate nearest neighbor search suffices. Formally, the $(1+\epsilon)$ -approximate NN, for $\epsilon > 0$ is defined as $\text{dist}(\mathbf{x}, \mathbf{q}) \le (1+\epsilon)\, \text{dist}(\mathbf{x}^*, \mathbf{q})$, where $\mathbf{x}^*$ is the true nearest neighbor.

1.3.2 Definition of Hashing

Hashing aims to map the database items into short codes so that approximate NN search can be efficiently performed. Formally, the hash function is defined as: $y = h(\mathbf{x})$, where y is the hash code and $h(\cdot)$ is the hash function. When represented with sequence of bits (say N in number), the hash code is $\mathbf{y} = \mathcal{H}(\mathbf{x})$, where $\mathbf{y} = [y_1 y_2 \cdots y_N]^T$ and $\mathcal{H} = [h_1(\mathbf{x})\, h_2(\mathbf{x}) \cdots h_N(\mathbf{x})]^T$. Depending on the application, one may either approximate the distance using a hash table lookup, wherein the hashing approach aims to maximize the probability of collision of near items. Alternatively, instead of exhaustive search, an non-exhaustive approach would be to fetch approximate nearest neighbors using the hash-codes and the fetched items are re-ranked using the true distance. In either scenarios, the true distance can be approximated using the Hamming distance computed between the hash-codes.

1.4 Learning to hash

The task of learning a compound hash function $\mathbf{y} = h(\mathbf{x})$ is to compute mapping of $\mathbf{x}$ to a compact code $\mathbf{y}$ such that the coding space is efficient and similarity-preserving *i.e.* can serve as an effective approximation to the true nearest neighbor search. There are three fundamental building blocks that define an instance of a *learning to hash* approach:

- **Nature of hash function**: The hash function can be based on linear projections [180], kernels [120], neural networks, non-parametric, and so on. A popular hash function is the linear model, wherein each hash bit is associated with a projection vector $\mathbf{w}$ and a quantization function (typically signum). For *e.g.*, $y = \text{sign}(\mathbf{w}^T\mathbf{x}) \in \{0, 1\}$, where $\text{sign}(\mathbf{w}^T\mathbf{x}) = 1$ if $\mathbf{w}^T\mathbf{x} \ge 0$ and 0 otherwise. The choice of hash function is a key decision that influences the efficacy of computing hash codes and its flexibility of partitioning the feature space. We discuss this in greater detail in Sec. 1.4.1.

- **Feature Representation and Similarity Measure**: For binary encodings, the Hamming distance (HD) is widely adopted as a similarity measure between two different hash codes and hence deemed as an *approximation* of the true distance between their respective data items. Given a hashing functions $\mathcal{H} = \{h_k(\cdot)\}_{k=1}^{K}$, the hamming distance $d_{\mathcal{H}}$ between two binary codes $\mathbf{y}_i = \mathcal{H}(\mathbf{x}_i)$ and $\mathbf{y}_j = \mathcal{H}(\mathbf{x}_j)$ is defined as:

  $$d_{\mathcal{H}}(\mathbf{y}_i, \mathbf{y}_j) = |\mathbf{y}_i - \mathbf{y}_j| = \sum_{k=1}^{K} |h_k(\mathbf{x}_i) - h_k(\mathbf{x}_j)| \tag{1.3}$$

  HD can be efficiently calculated as the bitwise `xor` operation and it must be noted that conducting exhaustive search in the Hamming space is significantly faster than doing the same in the original feature space due to its low memory footprint and computational cost. Variants such as normalized HD [205] asymmetric HD [67], query-adaptive HD [220] have also been explored [193]. We discuss aspects related to feature representation for encoding inter-item similarity in Sec. 1.4.2.

- **Nature of Supervision**: The optimization to learn hash functions attempts at achieving an approximate nearest neighbor search result closest to the true search result with the order of similarity preserved. For unsupervised settings, this translates to a scenario where the distance computed in the encoded space closely mimics the distance computed in the input space. For the case of semantic hashing, the neighborhood estimated using approximate NN should preserve the semantic similarities that we intend to encode. This loosely motivates the *Code Consistency* criterion that penalizes larger distances in the coding space but with larger similarities in the input space (either semantic similarity or closeness in Euclidean distance). Besides similarity preservation, another widely adopted optimization criterion is code balance that refers to hash codes to be uniformly distributed. We present a brief primer on the nature of supervision in scenarios of learning to hash in Sec. 1.4.3.

1.4.1 Nature of hash function

The goal of learning to hash is to learn learn data-dependent and task-specific hash functions that yield compact binary representations to achieve good search performance. Depending on the nature of the hash function $\mathcal{H}$, hashing methods can be broadly categorized into two groups *viz.* linear and non-linear. We briefly discuss the pros and cons of such hash function design as follows:

- **Linear**: Due to their computational efficacy, these methods aim at learning optimal projections to map original feature space to the Hamming space. Earliest approaches in this direction include unsupervised hashing methods such as PCA hashing [214] which performed principal components analysis on the data to derive projections that preserve the largest variance; under supervised settings, methods include Linear Discriminant Analysis (LDA) Hashing [185] which are designed to learn more discriminative hash codes. In addition to variance of projections, often additional constraints like orthogonality [66], equal-variance projections [100], correlated projection learning with error-corrective encoding [197] have also been investigated. Despite their simplicity and low-computational requirement for hash code generation, linear hashing

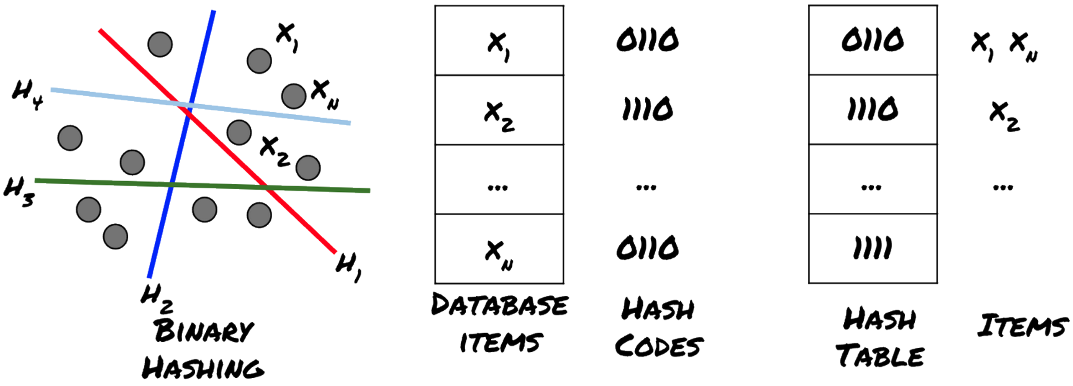


Fig. 1.2. *Illustrative example of training phase of linear projection based binary hashing*: The input feature space ($\{\mathbf{X}_i\}_{i=1}^{N}$) is parsed into sub-spaces using oblique hyperplane based hashing functions ($\{\mathbf{H}_i\}_{i=1}^{M}$). Depending on the membership of each of the data points with respect to the learnt hashing functions, binary hash codes are allotted. The database items are re-organized into the hash table, with items belonging to the same hashed sub-space grouped into the same hash bucket (characterized by an unique hash code). This hash table is representative of distribution of the data items in the original input space and is used for calculating approximate nearest neighbors during querying.

methods suffer from insufficient discriminative power due to insufficiency in model complexity and are of limited application in scenarios wherein differences amongst data are not subtle and linearly inseparable. Fig. 1.2 and Fig. 1.3 present illustrative examples of the training and testing phases of linear projection based hashing respectively.

- **Nonlinear**: To override the limitations of linear hashing, methods that non-linearly map the feature space into binary codes have been investigated. Prior art in this direction include spectral hashing [201] that uses sinusoidal hash functions, methods using kernelized encodings such as Kernel Sensitive Hashing [120], Compressed Hashing [116], Anchor Graph Hashing [119], Optimized Kernel Hashing [72] *etc.* In parallel, methods that aim at learning optimized discriminative kernels such as Metric Learning Hashing [141], Supervised Kernel Sensitive Hashing [120], Maximum Inner Product Hashing [114] *etc.* have been investigated.

1.4.2 Feature Representation

Visual feature extraction aims at representing low-level image content in a meaning fashion such that it links to high-level perceptive concepts associated with the images. A good feature representation is critical for good performance of CBIR systems as it is directly correlated with the definition of image similarity. This has been an active area of research within the machine learning and computer vision communities and has produced seminal contributions that can be broadly categorized into hand-crafted and learned features. The main distinguishing factor between the two types are whether the features are obtained through domain expert knowledge or purely learnt from the data itself.

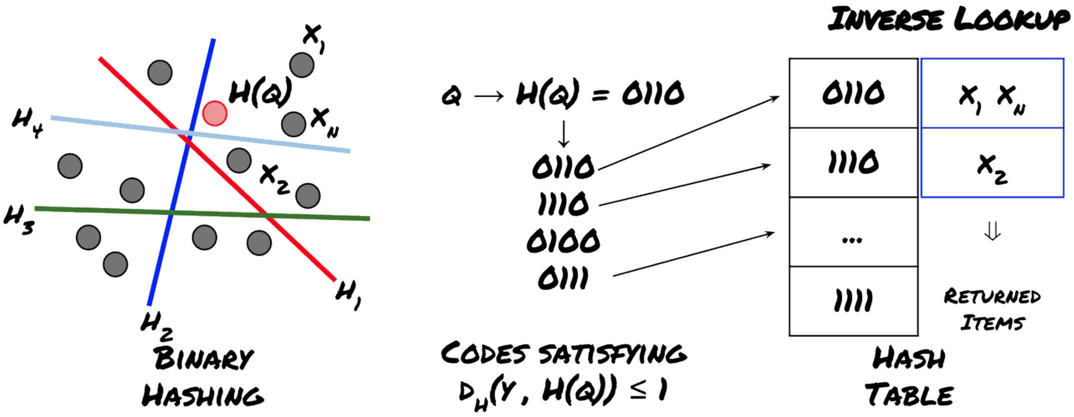


Fig. 1.3. *Illustrative example of inverse lookup in a hash table*: Given a query $\mathbf{Q}$ that maps to the subspace represented with hash code $\mathbf{H}(\mathbf{Q})$, we perform approximate nearest neighbor search by comparing with the hash table of the database. Illustratively, say $\mathbf{H}(\mathbf{Q}) = 0110$ and we intend to fetch database items that are at most 1 Hamming distance from $\mathbf{H}(\mathbf{Q})$, the potential neighbors would belong to the buckets indexed as 0110, 1110, 0100 or 0111. We fetch the items belonging to the aforementioned hash codes directly from the hash table, *sans* the need to perform pairwise comparisons. The latter aspect of retrieval with hashing contributes significantly to speed-up.

The definition of similarity is closely related to the task of extracting feature representations that preserve the semantics of the task at hand. Such a choice needs to bridge the following gaps in representation:

- *Sensory gap*: This is the difference between the real-world representation of the object to what is described by the features. The chosen features should ideally be robust towards less relevant aspects of the image such as noise, low illumination, partial occlusion to name a few. This gap is further compounded when information is lost when 2D images are used for physical 3D objects due to choice of view points, acquisition settings such as imaging quality, exposure, digitization and compression methods used.

- *Semantic Gap*: This gap arises due to inability of features to accurately represent the underlying semantics within the image and the associated *intent* of the user.

In the rest of the thesis, learning based hashing methods that employ hand-crafted features (discussed in Sec. 1.4.2) are referred to as *Shallow*-learning based hashing methods (or Shallow Hashing for short) as they decouple feature representation and binarization into two stages. In contrast to shallow hashing, methods that learn representations directly from the data (representation learning, discussed in Sec. 1.4.2) and hash them (also learnt end-to-end from the data) are referred to as *Deep*-learning based hashing methods (or Deep Hashing for short).

Hand-Crafted Features

Hand-crafted features are sequentially extracted for each image (globally or locally) and are generally based on expert knowledge wherein the feature encodes specific image content such as color, texture, histograms *etc.* Some popular descriptors employed in prior literature for the task of CBIR can be categorized as follows:

- **Local Features**: These features are descriptive of information around interest points such as corners, edges *etc.* Local features have the advantage of being robust to occlusion or clutter, highly descriptive, distinctive and efficient to extract. They are often modeled using a Bag-of-Words representation to obtain a global description of the whole image typically by counting the frequency of the generated visual words within the image. Popular features under this category include: Pixel Intensities, Haralick's Gray Level Co-occurence Matrix based features [82], Wavelets [121], Scale Invariant Feature Transform [111, 123], Speeded Up Robust Features [14], Local Binary Patterns [215], Gabor texture features [219], Tamura textural features [82], to name a few.

- **Holistic Features**: These features directly represent the global information in the entire image. Popular features under this category include GIST [142], Histogram of Oriented Gradients [83], Color Histogram [117], Moments [213], Fourier Descriptors [60], to name a few.

Despite achieving significant strides in CBIR through informed choice of hand-crafted features, they have short-comings when extended to large-scale retrieval. Domain knowledge to choose appropriate features is often limited and does not work particularly well when the database scales including potential outliers and cases that do not conform with standardized rules. These methods are computationally intensive and contribute to the retrieval time overhead which could critically limit retrieval time efficiency. Further, these features are often designed with specific use cases and cannot be easily generalized on heterogeneous image databases.

Similarity amongst data items can be measured in a multitude of ways. If the features are represented as a vector, distance metrics such as the Euclidean distance are used; for subtle geometric differences, elastic deformations can be used [47]. If the spatial relationships amongst objects of interest is to be encoded, then posing similarity estimation as a graph matching problem is a potential solution [11]. Finally, statistical classifiers can be used to automatically extract keywords and search terms from within query images, which in turn can be used to fetch examples with similar attributes.

Representation Learning

Of late, approaches to hierarchically learn semantically relevant representations directly from the data, collectively termed as Deep Learning, have become increasingly popular. In contrast to hand-crafting features using domain knowledge, deep learning performs feature discovery in a self-taught manner. Deep Networks are designed in a non-linear and hierarchical fashion, such that they map features from fine to abstract with multiple layers of neural networks and a large number of learned parameters. From a general perspective, deep learning was facilitated by the availability of large training datasets (such as ImageNet [49], MS-COCO [115], PASCAL VOC [59] *etc.*) making it possible for parameter optimization. Within the medical domain, with availability of large-scale medical image databases (such as TGCA [70], DDSM [78], Chest-XRay8 [198], ImageCLEF [135], TCIA [30] *etc.*), deep learning paradigms, both supervised and unsupervised, are being increasingly adopted for end-to-end learning.

Deep neural networks (DNNs) consist of an input layer which interfaces with the input feature vector, followed by multiple hidden layers of neurons with nonlinear connections (through non-linear activation functions such as sigmoid, tangent hyperbolic *etc.* [55])

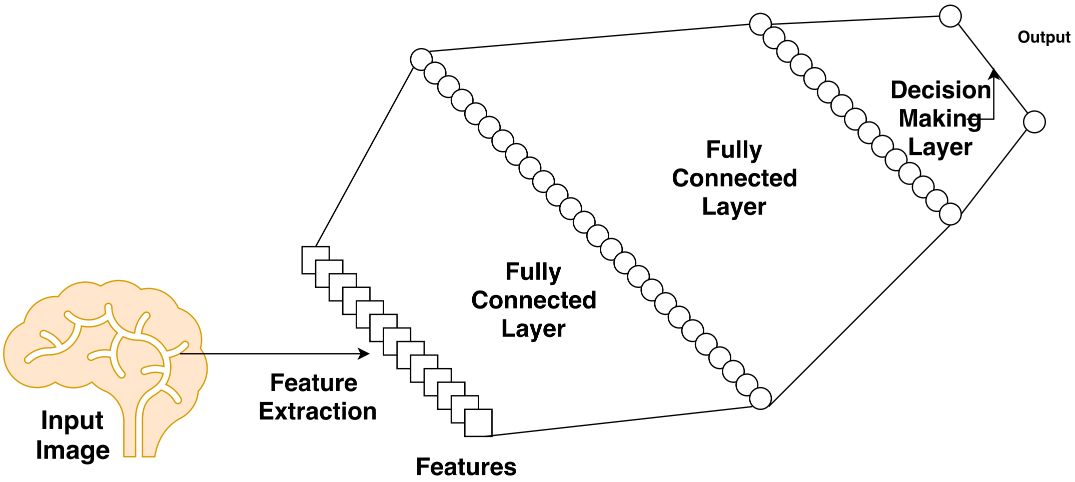


Fig. 1.4. *Illustrative example of architecture of deep neural networks*: DNNs consist of an input layer which interfaces with the input feature vector, followed by multiple hidden layers of neurons with nonlinear connections (through non-linear activation functions such as sigmoid, tangent hyperbolic *etc.*) terminating in a decision-making layer (such as *softmax* for classification; *binary encoding* for hashing *etc.*)

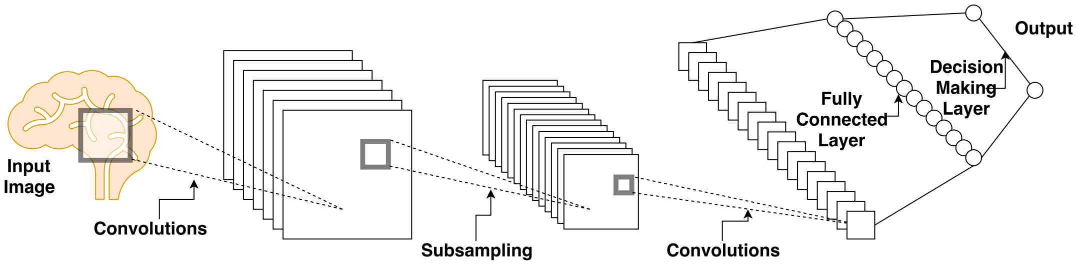


Fig. 1.5. *Illustrative example of architecture of convolutional neural networks*: Convolutional layers apply convolution operation to input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli. Pooling operator combine the outputs of neuron clusters at one layer into a single neuron in the next layer. Fully connected layers connect every neuron in one layer to every neuron in another layer and work on principals similar to a DNN. CNNs share weights in convolutional layers, which means that the same filter is used for each receptive field in the layer; this reduces memory footprint and improves performance.

terminating in a decision-making layer (such as *softmax* for classification; *binary encoding* for hashing *etc.*)(schematically illustrated in Fig. 1.4). Typically, each hidden layer consists of a set of neurons that are completely and independently connected to every other neuron in the previous layer (without any shared connections) [165]. In contrast to this, Convolutional Neural Networks (CNN) are tailored for fixed size inputs that are convolved with multiple shared kernels used in conjunction with non-linear activation functions (such as Rectified Linear Units (ReLU) [137]; parametric ReLU (pReLU) [74]; exponential Linear Units (eLU) [31] *etc.*)(schematically illustrated in Fig. 1.5). Additional layers worth mentioning include: pooling layers to down-sample representations (typically max- / mean-pooling); batch-normalization layers to compensate for internal covariate shifts [87]; fully-connected layers that are akin to DNNs and ultimately terminating in a decision making layer. Parameter optimization within CNNs is typically done through back-propagating error gradients in a reverse fashion by greedily updating parameters in directions that optimize the task-specific loss function [79]. Towards this a number of gradient-based optimization techniques have been proposed including popular approaches such as stochastic gradient descent [79];

ADAM [98]; RMSProp [80] *etc.* To prevent model overfitting or memorization of the training data, often regularizations such as weight decay [104], dropout [184] *etc.* are adopted.

Starting from the earliest variant of CNNs for hand-written number recognition (termed LeNet [110]), seminal advances towards training very deep networks with advanced network architectures have been made notably, AlexNet [103], GoogLeNet [187], VGG Net [26] and ResNets [73]. Supervised training of CNNs requires a large amount of labeled training data. Within domain of medical imaging, despite the deluge of data, obtaining high-quality manual annotations is challenging and severely limits the size of the available training data-base. Some popular approaches adopted to accommodate medical image analysis with deep learning include:

- *Transfer Learning*: Instead of training CNN models from scratch, pre-trained architectures from related tasks are adopted as a starting point and fine-tuned for the task at hand with limited training data. Potentially one could use networks trained for image classification on natural images (say, ImageNet [49]) or using auxiliary labels generated through pre-existing approaches [160].

- *CNN as a feature extractor*: In scenarios with severe training data limitations, pre-trained CNN models can be employed to extract features which can in turn be used complementary to hand-crafted features [170].

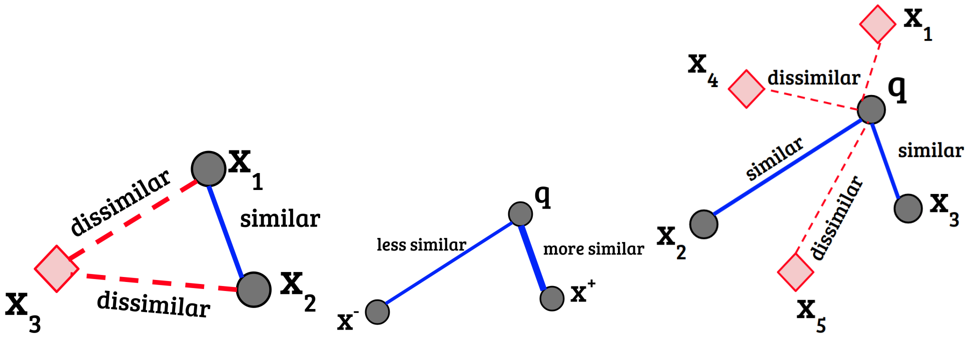
- *Data Augmentation*: Synthetically transforming training data with class-preserving transformations (such as geometric transformations, color variations, deformable transformations *etc.*) can significantly boost the exploratory aspect of CNNs [202]. Additionally, synthetic images from simulations could also be potentially used to augment training data [53].

- *Unsupervised Pre-training*: Leveraging vast amounts of unsupervised data to learn dataset specific latent features by minimizing loss functions such as reconstruction error is also widely adopted. Some noteworthy deep architectures under this category include auto-encoders including variants such as denoising autoencoders [191], variational autoencoders [99], convolutional autoencoders [128] *etc.*; generative models such as retricted Boltzmann machines [189], deep belief networks [109] *etc.*.

Clearly with a positive trend of increasing amounts of labeled training data within medical image analytics, the approaches of data-driven representation learning is the way of the future.

1.4.3 Nature of Supervision

Like machine learning paradigms, emerging hashing methods employ different levels of supervision ranging from unsupervised and supervised to semi-supervised variants. Broadly, unsupervised methods target integration of properties of the data manifold and distributions to design compact hash codes. Representative unsupervised methods include spectral hashing [201], anchor graph hashing [118], manifold hashing [172], iterative



(a) Pairwise (b) Triplet (c) Higher-order

Fig. 1.6. *Nature of Supervision in learning to hash*: The neighborhood relationships amongst data base items can be pair-wise (as shown in Fig. 1.6a) or include ranked-order such as in triples (as shown in Fig. 1.6b). Alternatively, higher order relationships including multiple neighbors can also be used for supervised hashing (as shown in Fig. 1.6c).

quantization [66], self-taught hashing [218], isotropic hashing [100] *etc.* to name a few. In addition, supervised hashing methods exploit a range of techniques from kernelization to metric learning to deep learning to learn binary codes and popular approaches under this category include metric learning hashing [141], kernel sensitive hashing (KSH) [120], supervised discrete hashing (SDH) [173], deep hashing network (DHN) [224], simultaneous feature learning and hashing (SFLH) [107] among others. Finally, semi-supervised learning paradigms employ both labeled and unlabeled data and aim at obtaining accurate and balanced hash codes [196].

The relationship amongst the data items in a supervised (or semi-supervised) hashing setting can be further used to sub-categorize hashing methods as follows:

- *Pointwise*: These approaches use instance-level semantic attributes to design hash functions *e.g.* parameter-sensitive hashing [169], predictable discriminative binary-coding [154] *etc.*

- *Pairwise*: Pairwise supervision is used to learn hash functions that preserve them in the Hamming space. As shown in Fig. 1.6a, the pair $(\mathbf{x}_1, \mathbf{x}_2)$ contains similar points, while the other two pairs $(\mathbf{x}_1, \mathbf{x}_3)$ and $(\mathbf{x}_2, \mathbf{x}_3)$ contain dissimilar points. Such hashing methods are the most widely investigated and aim at preserving relations between data samples in the learned Hamming space. Learning methods employing such a supervision include fast similarity search [218], kernel sensitive hashing [120], hamming-distance metric learning [141], semi-supervised hashing [196], non-transitive hashing [143] *etc.* It must be noted that as ranking information is not fully utilized, the performance of pairwise supervision might be sub-optimal for nearest neighbor search.

- *Triplet and Higher-Order Relationships*: Triplets encode pairwise proximity information amongst three data points. As demonstrated in Fig. 1.6b, the point $\mathbf{x}^+$ is more similar to query point $\mathbf{q}$ than point $\mathbf{x}^-$. The methods that utilize ranking information such as triplets (comparison amongst three data points), list-wise information (rank-order of similarities amongst samples) [197], lifted-structure embedding (simultaneous use of positives or negatives samples as illustrated in Fig. 1.6c) [183] *etc.* are gaining popularity due to their ability to more efficiently preserve ranking in the Hamming Space.

1.5 Code Consistent Hashing

Formally, code-consistent hashing refers to a category of hash functions that are based on minimizing the similarity-weighted distance $s_{ij}\text{dist}(\mathbf{y}_i, \mathbf{y}_j)$ (and possibly maximizing $d_{ij}\text{dist}(\mathbf{y}_i, \mathbf{y}_j)$, to formulate the objective function. Here, s_{ij} is the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$ defined either in the input feature space or extracted from semantic relationships. In this section, we will explore various prior art learning based code-consistent hashing techniques to design data-specific indexing schemes, specifically looking into their technical merits, advantages and drawbacks. It must be noted that the performance of each of these methods depends on practical settings, including learning parameters and the dataset itself. In general, methods under the category of non-linear and supervised hashing tend to learn more optimized and discriminative hash codes over linear and unsupervised methods, while trading off computational efficacy. In Table 1.1 and Table 1.2 we summarize the state of the art shallow hashing and deep hashing methods along with the contributions of this thesis within the respective categories.

1.5.1 Projection based Hashing

One of the earliest works on partitioning the input space was Locality Sensitive Hashing (LSH) proposed by Slaney and Casey [180]. The idea behind the LSH is that if two points are similar and closely embedded, then upon randomized linear projections they will remain close to each other. The LSH generates binary encoding by partitioning feature space with randomly generated hyperplanes [**slaney2008**, 64]. The hash functions for LSH can be parameterized as:

$$h_k(\mathbf{x}) = \text{sgn}\left(\mathbf{w}_k^T\mathbf{x} + b_k\right) \tag{1.4}$$

where $\{\mathbf{w}_k, b_k\}_{k=1}^K$ are parameters of the hash function representing the projection vector $\mathbf{w}_k$ and the corresponding intercept b_k. The LSH is a data independent method since it randomly generates hashing functions regardless of the data distribution. To mitigate the randomness of projections while simultaneously induce data-dependence, LSH was extended by projecting the input space along uncorrelated dimensions via PCA and learning a hashing function that minimizes the error between the new feature matrix and the corresponding binarized feature matrix [214]. Despite being highly efficient, linear projection based methods suffer from the same pitfalls discussed in Section 1.4.1 of insufficient model complexity and poor generalizablity.

1.5.2 Spectral Hashing

Spectral Hashing (SH) aims at preserving neighborhoods in the input space to neighborhoods in the learned Hamming space and requires hash codes to be balanced and uncorrelated [201]. To find the best code for a dataset, SH draws analogy with the problem of graph partitioning and additionally introduces relaxed orthogonality and balanced partitioning constraints. Like in spectral graph analysis, SH obtains solutions that are a subset of thresholded eigenvectors of the graph Laplacian. Let $\{\mathbf{y}_n\}_{n=1}^{N} = 1^N$ be the hash-codes of N data items, with each $\mathbf{y}_n$ of M bit size. Let s_{ij} be the similarity that correlates with the Euclidean distance. The Spectral Hashing formulation is given as:

$$\min_{\mathbf{Y}} \text{Trace}(\mathbf{Y}(\mathbf{D}-\mathbf{S})\mathbf{Y}^T) \tag{1.5}$$

$$\begin{aligned} s.t. \qquad \mathbf{Y1} &= \mathbf{0} \\ \mathbf{YY}^T &= \mathbf{I} \\ y_{im} &\in \{-1, 1\} \end{aligned} \tag{1.6}$$

where $\mathbf{Y} = [\mathbf{y}_1\mathbf{y}_2 \cdots \mathbf{y}_N]$, $\mathbf{S} = [s_{ij}]$ is the similarity matrix of size $N \times N$, $\mathbf{D}$ is a diagonal matrix Diag$(d_{11}d, \cdots, d_{NN})$, and $d_{nn} = \sum_{i=1}^{N} s_{ni}$. $\mathbf{D}-\mathbf{S}$ is the Laplacian matrix and

$$\text{Trace}(\mathbf{Y}(\mathbf{D}-\mathbf{S})\mathbf{Y}^T) = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} \left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2^2 \tag{1.7}$$

The constraints $\mathbf{Y1} = \mathbf{0}$ and $\mathbf{YY}^T = \mathbf{I}$ correspond to bit balance and bit uncorrelated requirements respectively. Instead of solving Eq. 1.5 directly, under the assumption of uniform data distribution, Weiss et al. [201] demonstrated that the spectral solution can be approximated and efficiently computed using 1D-Laplacian eigenfunctions. The algorithm is as follows:

1. Find the principal components of the N-d data using PCA.

2. Compute the M 1d Laplacian eigenfunctions with the smallest eigenvalues along each PCA direction. For the case of uniform distribution defined on bounds $[r_1, r_2]$, the 1d eigen function $\phi_f(x)$ is

$$\phi_f(x) = \sin\left(\frac{\pi}{2} + \frac{f\pi}{r_2 - r_1}x\right) \tag{1.8}$$

   and the corresponding eigenvalue λ_f is

$$\lambda_f = 1 - \exp\left(-\frac{\epsilon^2}{2}\left|\frac{f\pi}{r_2 - r_1}\right|^2\right) \tag{1.9}$$

3. Pick the M eigen functions with the smallest eigen values amongst the Md eigenfunctions.

4. Threshold the eigenfunction $\phi_f(x)$ at 0 to binarize into hash codes.

The computed projects are data-dependent and are learnt in an unsupervised fashion. However, this method suffers from low-quality binary encodings when using low-variance projections. To alleviate performance degradation upon extension to longer hash codes, a variant of SH was

Hashing Method	Input Similarity	Hash Function	Optimization Criteria
Locality Sensitive Hashing [180]	Euclidean	Linear	Random Projections
Spectral Hashing [201]	Euclidean	Laplacian Eigenmaps	Coding Consistency Bit Balance Bit Uncorrelation
Hashing with Graphs [119]	Semantic	Kernel	Code Consistency Bit Balance
Multi-dimensional Spectral Hashing [200]	Euclidean Semantic	Kernel	Coding Consistency Bit Balance Bit Uncorrelation
Linear Discriminant Analysis Hashing [185]	Semantic	Linear	Code Consistency Projection Uncorrelation
Supervised Hashing with Kernels [120]	Semantic Euclidean	Linear Kernel	MDS
Unsupervised Hashing Forests [42] (Chapter 2)	Euclidean	Tree-based Nodewise Linear	Code Consistency Projection Uncorrelation Bit Balance MDSD
Metric Hashing Forests [40] (Chapter 3)	Semantic	Tree-based Nodewise Linear	Code Consistency Projection Uncorrelation MDSD MD^3 Maximum Margin

Tab. 1.1. *Summary of shallow-learning based hashing algorithms*: This table tabulates state of the art shallow hashing methods along with the contributions made in this thesis and summarizes the underlying input similarity, underlying hash function and optimization criterion used. (*Abbreviation*: MDS - Minimizing differences between similarities; MDSD - Minimizing differences between similar distributions; MD^3 - Maximizing differences between dissimilar distributions; NCA - Neighborhood Component Analysis)

proposed (termed as Multidimensional Spectral Hashing) which employed kernalization [200]. It must be noted that a critical bottleneck with this approach was the assumption of uniform data distribution which hardly holds for real-world data.

1.5.3 Anchor Graph Hashing

The main challenge of Spectral Hashing in solving Eq. 1.5 and Eq. 1.7, other than the uniform distribution assumption, was defining w_{ij}, the computation of associated graph Laplacian and the associated solving of the eigen-system which had quadratic complexity. Following on similar lines as SH, Anchor Graph Hashing by Liu et al. [119] alleviates the assumption of uniform distribution and creates an efficient approximation of the graph structure by building the pairwise similarity graph $\mathbf{A}$, using a small set of anchor points such that similarity between any two points can be approximated using point-to-anchor similarities. Such an approximation is both computationally efficient and representative of the neighborhood structure.

In particular, the truncated point-to-anchor similarity $\mathbf{Z} \in \mathbb{R}^{N\times M}$ gives the similarity between N database items to M anchor points. The approximated similarity matrix $\hat{\mathbf{A}}$ is calculated as $\hat{\mathbf{A}} = \mathbf{Z}\Lambda\mathbf{Z}^T$, where $\Lambda = \text{diag}(\mathbf{Z}1)$ is the degree matrix of anchor graph $\mathbf{Z}$. Based on such an approximation the eigen-system of the matrix can be obtained by solving a much smaller eigen-system with an $M \times M$ matrix $\Lambda^{\frac{1}{2}}\mathbf{Z}\mathbf{Z}^T\Lambda^{\frac{1}{2}}$ and the final binary codes are obtained through binarizing the spectral embedding as:

$$\mathbf{Y} = \text{sgn}\left(\mathbf{Z}\Lambda^{\frac{1}{2}}\mathbf{V}\Sigma^{\frac{1}{2}}\right) \tag{1.10}$$

where, $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_k, \cdots, \mathbf{v}_K] \in \mathbb{R}^{M \times K}$ and $\Sigma = \text{diag}\left(\sigma_1, \cdots, \sigma_k, \cdots, \sigma_K\right) \in \mathbb{R}^{K \times K}$, where $\{\mathbf{v}_k, \sigma_k\}$ are the eigenvector-eigenvalue pairs. For out-of-sample extension, AGH extrapolates the graph Laplacian eigenvectors to eigenfunctions and binarizes via hierarchical thresholding.

1.5.4 Hashing with Kernels

To extend hashing methods into handling high-dimensional features and learn complex distance functions into a low-dimensional Hamming space, Kernelized Locality-Sensitive Hashing generalizes LSH to handle arbitrary kernel functions to accommodate linearly inseparable data [120]. This work was extended into the supervised domain by considering pairwise similarity/ dissimilarity information while encoding and minimizing the Hamming affinity over hash codes and the similarity over data items. The similarity ($s_{ij} = 1$) or dissimilarity ($s_{ij} = -1$) is given by label information or Euclidean distance. The hash function is defined as follows:

$$y_{mn} = h_m\left(\mathbf{x}_n\right) = \text{sign}\left(\sum_{t=1}^{T_m} w_{mt} K\left(\mathbf{s}_{mt}, \mathbf{x}\right) + b\right) \tag{1.11}$$

where $\{\mathbf{s}_{mt}\}_{t=1}^{T_m}$ are sampled data points forming the hash function $h_m\left(\cdot\right) \in \{h_1\left(\cdot\right), \cdots, h_M\left(\cdot\right)\}$, $K\left(\cdot, \cdot\right)$ and $\{w_{mt}\}$ are weights to be learnt and b is the bias. The objective function is derived as follows:

$$\min \sum_{(i,j) \in \mathcal{L}} \left(s_{ij} - \text{affinity}\left(\mathbf{y}_i, \mathbf{y}_j\right)\right)^2 \tag{1.12}$$

where $\mathcal{L}$ is a set of labeled pairs and $\mathbf{y} \in \{1, -1\}^M$ The Hamming affinity between hash codes $\mathbf{y}_i$ and $\mathbf{y}_j$ is defined as:

$$\text{affinity}\left(\mathbf{y}_i, \mathbf{y}_j\right) = M - \|\mathbf{y}_i - \mathbf{y}_j\|_1 \tag{1.13}$$

This was achieved by posing Hamming distances with their equivalent code inner products and using an efficient greedy algorithm to sequentially solve for the target hash functions bit by bit.

1.5.5 Deep Hashing

Of late, deep learning based approaches are being increasingly adopted for end-to-end learning of representations tailored for hashing, effectively mitigating the two-stage approach of prior shallow hashing based methods (*i.e.* extraction of features followed by binarization). The main purpose of such methods is to learn robust and powerful feature presentations which are binarized to generate compact hash codes. In this section we review a few popular prior art deep hashing methods, discussing their contributions, pros and cons.

Semantic Hashing by Salakhutdinov and Hinton [164] is one of the earliest works in using deep learning for hashing. The approach builds a deep generative model based on stacked

Hashing Method	Input Similarity	Hash Function	Optimization Criteria
Semantic Hashing [164]	Semantic	Neural Networks	MDS
Deep Hashing [58]	Euclidean Semantic	Neural Networks	MDD (in Euclidean) MDS (in Semantic) Bit Balance Quantization Error Equal Variance
Simultaneous Feature Learning and Hashing [107]	Semantic	Convolutional Neural Networks	Triplet Loss Quantization Error
Deep Hashing Network [224]	Semantic	Convolutional Neural Networks	MDS Quantization Error
Deep Residual Hashing [39] (Chapter 4)	Semantic	Convolutional Neural Networks	NCA Code Consistency MDS Bit Balance Quantization Error Bit Uncorrelation
Robust Multiple Insance Hashing [37] (Chapter 5)	Semantic	Convolutional Neural Networks	Robust NCA Code Consistency MDS Quantization Error

Tab. 1.2. *Summary of deep learning based hashing algorithms*: This table tabulates state of the art deep hashing methods along with the contributions made in this thesis and summarizes the underlying input similarity, underlying hash function and optimization criterion used. (*Abbreviation*: MDS - Minimizing differences between similarities; MDSD - Minimizing differences between similar distributions; MD^3 - Maximizing differences between dissimilar distributions; NCA - Neighborhood Component Analysis)

Restricted Boltzmann Machines (RBM) with hidden binary units (*i.e.* latent topic features). The hash codes were generated by thresholding the output of the deepest layer and were shown to preserve semantically similar relationships in the code space. This was an unsupervised learning paradigm without end-to-end learning of features. In a supervised extension, Torralba et al. [189], introduced the idea of Neighborhood Component Analysis (NCA) [65] to incorporate supervised information (neighbor / non-neighbor relationships) between the training examples. Then, the objective function of NCA was optimized on top of a deep RBM, making the deep RBM yield discriminative hash codes.

With strong efforts within the machine learning and artificial intelligence communities, kicking off with Krizhevsky et al. [103] in 2012, convolutional neural networks (CNNs) have found increasing applicability for end-to-end learning of hash-codes. In the work titled *Deep Hashing*, Erin Liong et al. [58] proposed to learn multiple hierarchical non-linear transformations that map the original image to compact binary hash codes. The model was supervised by introduction of the reconstruction error between original real-valued feature vector and the resulting binary codes. Additional constraints to generate balanced and independent hash codes were imposed. Within the same work, to incorporate supervision when available, an additional discriminative term incorporating pairwise supervised information was added to the objective function. In the work of Lai et al. [107], they use a stack of convolutional layers to produce effective intermediate features which are further stratified via a divide-and-encode module to compress and encode into one hash bit each to simultaneously learn features and hashing functions on top of them. The model was trained *via* a triplet ranking loss and a piece-wise threshold function was used for binarization. In the work, titled *Deep Hashing Network*, Zhu et al. [224] jointly learn image representations tailored for hash coding and formally control the quantization error to improve hash code quality. Their approach proposed using a stack of convolutional layers terminating in a fully-connected hashing layer, which is trained end-to-end with a pairwise cross-entropy and a pairwise quantization loss. In a recent work, Yao et al. [209] propose the idea of co-training for hashing by jointly learning projections from image representations to hash codes and classification. The hash codes were learnt to preserve multilevel semantic similarities amongst images with multiple semantic annotations. They employed the triplet ranking loss with orthogonality constraints to learn discriminative and independent hash bits.

1.6 Evaluation of CBIR Systems

Post retrieving images, the task of evaluating the quality of retrieval within CBIR systems is also critical. Towards this, several efforts have been made to benchmark dedicated retrieval specific databases and tasks (such as ImageCLEF [95], VISCERAL [188] *etc.*). In this section, we present the evaluation metrics and criterion popularly used to quantify retrieval performance.

1.6.1 Evaluation Metrics

Evaluation metrics used to quantitatively evaluate the efficacy of CBIR systems. These include precision, recall and retrieval time-related metrics.

Precision: This refers to the fraction of images retrieved that are relevant to the query image, thus effectively evaluating the ability of CBIR systems to fetch similar or relevant samples.

$$\text{precision} = \frac{|\{\text{Relevant Images} \cap \text{Retrieved Images}\}|}{|\{\text{Retrieved Images}\}|} \tag{1.14}$$

Recall: This refers to the fraction of relevant images retrieved within all the relevant images in the database, *i.e.*:

$$\text{recall} = \frac{|\{\text{Relevant Images} \cap \text{Retrieved Images}\}|}{|\{\text{Relevant Images}\}|} \tag{1.15}$$

Recall evaluates the sensitivity of the retrieval system *i.e.* whether the system is able to fetch all relevant samples in a top-K ranked list, while keeping K as small as possible. Beyond precision and recall, joint measures that combine both the metrics are also popularly adopted, such as $\mathrm{F}_{\text{score}}$, G_{mean}:

$$\mathrm{F}_{\text{score}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{1.16}$$

$$\mathrm{G}_{\text{mean}} = \sqrt{\text{Precision} \times \text{Recall}} \tag{1.17}$$

To obtain a global view of the performance of the CBIR system it is common to associate precision with recall typically in the form of a Precision-Recall curve (PR-curve). Every point in the PR-curve is evaluated varying the scope of retrieval (*i.e.* K). From the PR-curve, we can derive the mean average precision (MAP) metric which is akin to the area under the curve of the Receiver Operating Characteristics used in evaluating classification performance. MAP is a relatively stable metric and is defined as the mean of the average precision scores over positions of all relevant database items (averaged over all queries). The MAP can be formulated as:

$$\text{MAP} = \frac{1}{|M|}\sum_{m=1}^{M}\frac{1}{|K|}\sum_{k=1}^{K}\text{precision}\left(Q_{m,k}\right) \tag{1.18}$$

where M is the number of query images, K indicated the top-K ranked relevant images for each query, and $Q_{m,k}$ denotes the top-k retrieval precision of the mth query. MAP can be used during massive testing and particularly alleviates the bias during precision evaluation.

To efficiency of the feature-indexing schemes the average time for retrieval and theoretical comparison of the time-complexity are often used. The average time typically accumulates the time for feature extraction on the query image, time to index the extracted features and more importantly time for inter-item distance computation and ranking. In addition to the aforementioned metrics, several other commonly employed measures include the precision after the first N_R images are retrieved, recall at 0.5 precision, rank-correlation between ranked-lists *etc.* A detailed analysis of these metrics were discussed in previous articles [133, 134].

1.6.2 Evaluation Criterion

In addition to the evaluation metrics discussed in Sec. 1.6.1, the criterion of decision similarity / relevance is important. In unsupervised retrieval settings, where the goal is to perform approximate nearest neighbor search in an efficient fashion, the similarity can be measured using distance metrics defined in the input space and the representation space aims at mimicking relationships encoded in the input space. In a supervised setting, two kinds of evaluation criterion are commonly employed:

Annotation-based Criteria: Here, annotations associated with images (such as class labels, key-words *etc.*) are used as a surrogate for retrieval performance. This is particularly suited for large-scale evaluation tasks where generating expert-defined ranked-lists for each query image by exhaustively exploring the database is expensive and subjective. The metrics of precision and recall are measured sequentially comparing the labels between each query image and the retrieved images.

User-based Criteria: In this setting, an expert user is often employed to manually evaluate retrieval query typically through feedback on similarity/relevance between the query and fetched neighbors. Such an approach can be adopted on smaller-scales especially for some analytical tasks where simply using the class labels is not an adequate representation of relevance.

Compared to annotation-based criterion, the use of domain experts to evaluate CBIR systems can result in more fine-grained evaluations in form of relevance judgments for specific tasks. However, such a criterion heavily relies on the user's domain knowledge and may be partly subjective based on the user's background. In such a setting, employing multiple users (in a blind fashion) and aggregating their opinions could be a more reliable evaluation. Markonis et al. [127] performed an user-oriented evaluation of CBIR systems with a team of 16 radiologists to provide insights into the required specifications and the potential short-coming of such systems. They concluded that designing user-oriented interactive systems is preferred and proposed future directions targeted at filtering by imaging modality and cross-modal retrieval by search for articles using image examples.

1.7 Image Retrieval in Medicine

Image analytics is a fundamental component of modern medicine and plays a central role in clinical diagnosis [52], treatment planning [216], image-guided surgery [125] and assessment of treatment-response [126]. Traditional decision making in diagnostics involves seeking evidence from a patient's data (both image and non-image factors) coupled with prior knowledge of similar cases [81]. In such a setting, the question of objective definition of image similarity is important as reliable interpretation and understanding of images within the clinical work-flow means access to relevant stored data for making a well-informed decision.

In a modern healthcare setup, data generated is often digitized, indexed and archived via the Picture Archiving and Communication Systems (termed as PACS). Such systems have enabled storage of large digital repositories of data and enables physicians to perform patient-specific diagnosis through their longitudinal data by presenting all images related to a particular patient [84]. In addition to this, PACS systems have opened up new opportunities for enhanced image understanding, pattern discovery, teaching and research based on contrastive analysis via inter-patient comparisons [133]. Towards achieving the aforementioned goals, we need an objective way of searching and indexing the repository such that when a new query is presented data sharing similar characteristics is efficiently retrieved. Currently deployed PACS systems are heavily dependent on search via textual keywords including patient name, identifiers like RadLex / ICD-10 codes and image device. This significantly limits the exploratory capacity of such a search engine due to limited and often confounding vocabulary employed. This also requires that the user to know the identifiers and the characteristics of data that they wish to fetch. This becomes an important bottleneck to overcome when systems scale and given the massive volumes of imaging data being stored in modern PACS systems a search by textual keywords or manually assigned nomenclature (or labels) is not viable [132]. As systems scale, both is size and number, manual annotation for retrieval is often impractical and uneconomical. The high degree of underlying subjectivity owing to the high dependence on the skill, training, experience and alertness of the expert could have potentially detrimental effects on the quality of the retrieval system driven by manual annotations.

Retrieval *vs.* Classification: The term Retrieval is often associated with information search wherein the user has an information need to be fulfilled by fetching data items that are relevant to the query at hand. This is often performed in a class-agnostic setting and the definition of relevance is user- and situation-dependent. Efficacy of retrieval algorithms are evaluated in terms of precision and recall and the ability of the algorithm to rank the fetched items in order of relevance. In contrast, with classic machine learning settings classification is tasked with assigning the correct label to an unseen sample from a finite set of classes. The performance measure in such a setting is typically the classification accuracy. From a broader perspective, both retrieval and classification fall under the umbrella of image understanding which aims at extracting higher level content from the image to help understand the global content within the image. Within the context of computer aided diagnosis (CAD), image retrieval can be positioned as an interactive way of image understanding as the clinician can pose a query with a case (or with a selected region of interest) and find visually similar cases from the search database.

1.7.1 Challenges within Image Retrieval in Medicine

Despite advances in large-scale methods in generic image retrieval problems their seamless adoption to the case of large scale medical image databases is still a challenge and open topic for investigation [106, 133]. The major domain-specific challenges associated with medical multimedia retrieval can be detailed as follows:

1. **Context**: In contrast to generic image databases, medical images always have a context and the associated meta-data such as why was the image acquired, complex imaging parameters (contrast agents, acquisition settings), demographic and patient information (typically as an electronic health record) are very important to provide overall context.

2. **Anatomical Variability**: Despite being acquired via fairly standardized protocols, anatomical differences between humans is very important as disease manifestations often seen in images are a result of complex link between multiple inter-dependent phenomena.

3. **Complexity in image acquisition**: Medical images are not acquired via digital cameras with direct sensors detecting light and the underlying physics behind their acquisition involves multiple stages such as sensing, detection, reconstruction, de-noising and digitization. These steps are often lossy in nature due to constraints of storage and time of acquisition and could be prone to artefacts associated with patient motion, registration errors and scanner variabilities which implies that images of the same patient acquired with the same type of scanner (exact model / manufacturer) can be significantly different.

4. **High Dimensionality**: The size of medical images can be extremely large with thousands of tomographic slices per patient to large gigapixel images for whole-slide histopathology. In addition to this high dimensionality, the relevant disease manifestations can be very subtle and requires dedicated routines targeted at localization and detection of these changes which requires a fine-grained and detailed analysis approach.

5. **Availability**: The biggest challenge for large-scale analysis of medical images is their availability owing to a variety of reasons including privacy, restrictions on data use or patient selection, associated questions of ethics and high cost to obtain quality annotations from medical experts. The above factors compound to create a scenario that despite ever-increasing volume of data being generated, little training data is available further complicating the development of image analytics solutions.

These challenges factor out as two important variables that define the usefulness of a medical CBIR system viz. system efficacy and accuracy. A medical CBIR system should be able to handle massive amounts of images via efficient indexing and search schemes and must not resort to exhaustive search across millions of images as this will be a critical bottleneck when scaling up. In addition to this, the system has to achieve a high retrieval precision in reasonable time and is trained to be sensitive to small, local anomalies and work under large intra- and inter-class variation.

1.7.2 Prior art within Medical Image Retrieval

Hospital information systems curate a large variety of information, ranging from patient demographics, clinical measurements, standardized / unstructured text reporting, test results and images. Within images, a range of modalities ranging from 2D modalities like cytology, histology, X-rays to volumetric images including Computed Tomography (CT), Ultrasound, Positron Emission Tomography (PET), Magnetic Resonance Imaging (MRI) *etc.* In such a setting developing CBIR systems that can handle this heterogeneous nature of patient information is a daunting effort and often targeted application-specific solutions are explored. In this section, we briefly review CBIR systems for medical image retrieval organized according to the nature of the search database. One of the earliest CBIR sytems was proposed by Comaniciu et al. [32] that supports decision making in clinical pathology. Muller et al. [134] explored the GIFT - GNU Image Finding tool and investigated a range of feature extraction methods targeted at the retrieval of medical images within the CasImage medical case database of the university hospitals of Geneva. We organize the prior art CBIR systems within the domain of medical imaging under the following types depending on the type of data within the database.

2D Image Retrieval

Major efforts of CBIR for medical images has been directed towards 2D images such as radiographs [69], histology [23, 54], dermatology images [20] *etc.* The techniques developed in this category reflect on how non-medical CBIR approaches have be adapted and tailored for medical applications. In a work on CBIR system targeted at dermatology applications Bunte et al. [20] proposed using concepts of limited rank matrix learning vector quantization for feature compression coupled with a Large Margin Nearest Neighbor approach for learning the distance metric. Within CBIR for histology applications, Doyle et al. [54] employed manifold learning for retrieval within digitized histopathology slides of prostate cancer. They investigated hand-crafted features descriptive of the architecture, morphology and texture of these images and embedded them into low-dimensional representation which encode object similarity using PCA and Laplacian eigenmaps. Caicedo et al. [23] proposed CBIR systems for use in digital pathology which aims at retrieving histopathology images from a large collection using an example image as query. They integrated multiple hand-crafted features with a selection mechanism and learnt to perform automatic semantic image annotation on query images. Zhang et al. [221] investigated CBIR techniques for computer aided detection of actionable manifestations in breast histopathology images. They adopted a supervised kernel sensitive hashing approach to learn compact informative signatures of high dimensional BOW representation of SIFT features. They demonstrated that adopting an hashing based approach was computationally efficient over dimensionality reduction and feature selection approaches and achieves a promising throughput of 800 queries in under 0.01 seconds. Haas et al. [69]

used superpixels within the bag of visual words framework to effectively local descriptors into representative and discriminative visual words. They targeted the task of categorization of radiographs and localization of the lung within thorax CTs through retrieval.

Within hashing for 2D medical image retrieval, Sze-To et al. [186] learned deep denoising auto-encoders with binary latent variables for tagging x Ray images with compact similarity preserving codes. They demonstrated in an extensive study with 14,000 chest X Rays that such an approach adaptively learn non-linear relationships amongst pixels to generate binary codes that encode high-level features efficiently. Liu et al. [118] introduced a novel hashing algorithm, termed as Composite Anchor Graph Hashing with Iterative Quantization to compress mammographic regions of interest into compact binary codes that captures similarities between the masses in an computationally efficient fashion by ensuring linear complexity of the training procedure and constant time for query. Jiang et al. [93] investigated joint kernel-based supervised hashing for fusion of multiple hand-crafted features towards the task of categorization of breast cancer histopathological images into actionable and benign cases.

3D+ Image Retrieval

Of late, with a deluge of 3D+ (including 3D, 2D+t, 4D data) acquired in clinical settings, many CBIR algorithms have been adapted for use in such modalities. The key design aspect in such extensions was first choosing *key slices* from the volume or extracting interesting regions of interests or representing 3D images with abstract representations such as graphs and indexing the same. André et al. [5] proposed a system for endomicroscopy video retrieval which uses the Dense SIFT descriptor with bag-of-words to compute a visual signature for each video. This visual signature was demonstrated to be concise and able to communicate high-level medical knowledge consistently. Lan and Zhou [108] introduced a novel discriminative hand-crafted feature termed as histogram of compressed scattering coefficients for the task of CBIR particularly suited for medical computed tomography images. Murala et al. [136] proposed a novel image descriptor combining binary wavelet transform with local binary patterns on binary bit-plane representations of 8-bit gray-scale images. This descriptor was evaluated on multiple medical image databases of MRI and CT and demonstrated significant improvements over LBP and LBP features with Gabor transform. Cai et al. [22] investigated 3D CBIR systems for functional imaging with applications in clinical dementia studies. The user query consisted of pathology-centric masks which was represented with textural features related to the cerebral metabolic rate of glucose consumption from neurological FDG PET images. Li et al. [112] introduced an asymmetric binary coding strategy based on the maximum inner product search (MIPS) for pattern exploration in large-scale neuronal databases and demonstrated potential use cases in identification and analysis of neuron characteristics.

Multimodal and Heterogeneous Databases

Under scenarios of retrieval from diverse databases, the CBIR system must have the capacity to differentiate between modalities while performing retrieval. Güld et al. [68] used global features for describing the contents within the image for the task of modality classification in addition to using downscaled representation of the original images for preserving spatial information and learning distance measures that are robust to variations in radiation dose, translation and local deformations. Caicedo et al. [24] targeted the task of modality classification within a large heterogeneous image database (ImageCLEFmed [95]) and proposed an categorization index to perform ranking of similar image modalities. They also demonstrated that combining holistic and low-level features as a better strategy towards learning semantics.

Extending CBIR to multiple images and modalities is also an emerging area of research, wherein CBIR systems adapt to use complementary information from different images and discover and exploit relationships between them. Rahman et al. [152] proposed a CBIR framework for handling heterogeneous collection of medical images from multiple imaging modalities, anatomical regions and biological systems. They deploy probabilistic multi-class support vector machine (SVM) and fuzzy c-mean (FCM) clustering to perform pre-filtering of the database to extract a subset of potentially relevant candidates. Following which an user interaction driven relevance feedback approach is adopted to incorporate perception subjectivity into the querying process and adjust image similarity matching functions. Depeursinge and Müller [50] extensively evaluated multiple information fusion strategies for fusing visual and textual information which were further categorized into early and late fusion strategies, with a few capable of inter-media query expansion. Cao et al. [25] designed a multi-modal CBIR system for cancer research. They investigated the use of a novel probabilistic Latent Semantic Analysis model to integrate visual and textual information. Interestingly for cases of missing modalities, they trained a deep Boltzmann machine-based multi-modal learning model to impute the missing modality.

System Solutions

Targeted at developing efficient and intelligent systems that can transform healthcare, the academia and industry has started developing and providing dedicated CBIR systems integrated within their PACS / data archiving systems. Antani et al. [6] investigated issues of interfacing multi-location CBIR systems *via* an XML-based data and resource exchange framework and demonstrated that the resultant framework was system portable and offered richer functionality to the user through a web interface. A few existing deployments of successful CBIR systems that impacted clinical practice and biological applications include:

- *CervigramFinder*: Xue et al. [206] developed a prototype CBIR system for retrieval within the uterine cervix image databases of the National Cancer Institute as a part of a multi-year longitudinal effort to study origins of cervical pre-cancer and cancer. The system is build with a distributed architecture with the core indexing and retrieval algorithms at the backend.

- *ViewFinder*: Agarwal and Mostafa [1] proposed the ViewFinder Medicine CBIR application for retrieval in Alzheimer's disease incorporating an multi-tier architecture with flexibility of relevance feedback.

- *ImageMiner*: Foran et al. [61] proposed the ImageMiner software platform for performing comparative analysis on expression patterns in tissue microarrays (TMA). The proposed pipeline had tools for analysis and data management for rapidly analyzing large ensembles of TMA data and further supports deployment in a multi-institutional collaborative environment through grid-enabled web-services.

- *Khresmoi Radiology Search System*: The system proposed by Hanbury et al. [71] aims at retrieval of similar volumes from tomographic images in radiology. Upon querying with a volume to be diagnosed, a selection of volumes containing similar regions of interest and associated radiology reports are fetched for multi-modal retrieval enriched by semantics.

- *IBM Watson Health* [28]: This is a modular system designed for deep content analysis and evidence based reasoning. In addition to these, there are routines available for providing personalized recommendations, learning from past experiences and an interactive routine via chat bots that engage in dialog and perform relevance feedback.

- *ASSERT* [175]: This system is deployed for retrieval of high-resolution CT lung images and was demonstrated to improve the accuracy of diagnosis made by the physicians. Based on key pathology-bearing slices marked by the physician, the system retreived images of the same type of lung pathology (*e.g.* emphysema, cysts, metastases *etc.*).

Part II

Contributions

Unsupervised Hashing Forests 2

> *There is no scientific study more vital to man than the study of his own brain. Our entire view of the universe depends on it.*
>
> — **Francis Crick**
> (Nobel Prize for Medicine 1962 for discovering double-helix structure of DNA)

2.1 Overview and Publications

This chapter presents the contributions of this thesis concerning learning to hash with random forests in an unsupervised fashion, termed as *Hashing Forests*. The chapter is established upon hashing (search and retrieval) technique by employing multiple unsupervised random trees, collectively called as Hashing Forests (HF). The HF are trained to parse the input space hierarchically and preserve the neighborhoods in input space while encoding with compact binary codewords. We further introduce the inverse-coding formulation within HF to effectively mitigate pairwise similarity comparisons, thus allowing scalability to massive databases with little additional time overhead. The proposed hashing tool has superior approximation of the true neighborhood with better retrieval and ranking performance in comparison to existing generalized hashing methods. With steady growth of digital neuroscientific data, there is an increasing demand for a reliable, systematic, and computationally effective retrieval algorithm. In this chapter, we present hashing forests as a potential tool for fast and accurate reference-based retrieval within neuron image databases. This is exhaustively validated by quantifying the results over 31266 neuron reconstructions from Neuromorpho.org dataset curated from 147 different archives. We envisage that finding and ranking similar neurons through reference-based querying *via* such an algorithm would assist neuroscientists in objectively understanding the relationship between neuronal structure and function for applications in comparative anatomy or diagnosis.

The chapter is organized as follows: in Sec. 2.2 we introduce the concept of neuron image retrieval and lay the premise for hashing with forests. The detailed mathematical formulation is exposéd in Sec. 2.3 with discussions on the key elements of the hashing model, namely cluster validity and tree balance during training of the hashing forests; and inverse lookup during testing. In the next section Sec. 2.4, we present detailed experiments and results of such an approach for the task of pattern exploration in neuron image databases. We evaluate the methods against a number of ablative baselines and state of the art unsupervised hashing based comparative methods. Finally, we conclude the chapter in Sec. 2.5, we discuss how well the proposed hashing forests approximates true neighborhood and how does the

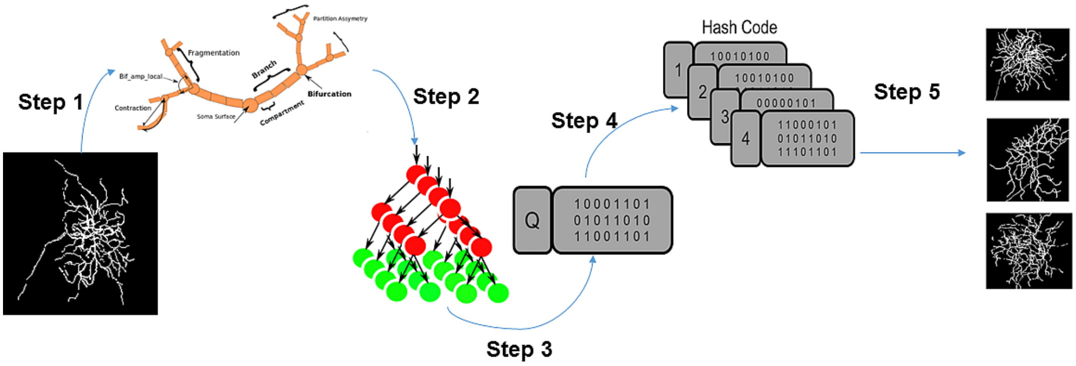


Fig. 2.1. *Schematic of the proposed method for neuron image retrieval with Hashing Forests*: For a query neuron, we extract neuromorphological features (Step 1), which are then fed into the learnt Hashing Forests (Step 2). This results in a similarity-preserving binary query hash code (Step 3). Comparing this hash code to the Hash Table (codes of neurons in the search database) through the proposed inverse coding scheme (Step 4), we find and rank neurons that are morphologically similar to the query neuron (Step 5). Reprint from [42], with permission of Springer.

performance fare as the dimensionality of the encoding space varies. We also present scenarios of incremental training as the database evolves to demonstrate the scalability of such an approach.

Substantial parts of this chapter have already been published in the following articles and are quoted verbatim:

[42] **Conjeti, Sailesh**, Sepideh Mesbah, Mohammadreza Negahdar, Philipp L. Rautenberg, Shaoting Zhang, Nassir Navab, and Amin Katouzian. "Neuron-Miner: An Advanced Tool for Morphological Search and Retrieval in Neuroscientific Image Databases." *Neuroinformatics 14, no. 4 (2016): 369-385.*

Copyright Statement. ©Springer Science+Business Media New York 2016.

[130] **Conjeti, Sailesh**, Sepideh Mesbah, Ajayrama Kumaraswamy, Philipp Rautenberg, Nassir Navab, and Amin Katouzian. "Hashing forests for morphological search and retrieval in neuroscientific image databases." *In International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 135-143. Springer, Cham, 2015.*

Copyright Statement. ©Springer International Publishing Switzerland 2015.

2.2 Introduction

Projection based hashing methods like PCA Hashing [214], iterative quantization [66], LDAH [185] *etc.* assume zero-centered datasets, with hash function h_k consisting of a projection matrix $\mathbf{w}_k$ and such that:

$$h_k(\mathbf{x}) = \text{sgn}(\mathbf{w}_k^T \mathbf{x}) \tag{2.1}$$

These methods are data-dependent, however their maximal code length is limited to intrinsic dimension of the data (typically, the original feature dimension if the features are de-correlated). As a consequence of this property, such methods are not natively scalable to large code sizes which sometimes limits their performance in large-scale datasets with low-dimensionality. Alternatively, one can add a bias factor b_k to Eq. 2.1 to threshold the projected values and modify it as: $h_k(\mathbf{x}) = \text{sgn}(\mathbf{w}_k^T\mathbf{x} + b_k)$. However, this can lead to hash functions that are non-independent and can thus be counter-productive to computational efficiency. In this context, learning random hyperplanes that parse the data (like in LSH [180]) has a favorable property of improved performance with increasing code length without the bottle-neck of limiting the code-size to data's intrinsic dimension. In this case, the number of hyperplanes has to be large enough to guarantee good performance which results in reduced speed of hashing.

Trees have been generally accepted as well-suited data structures for indexing large-scale datasets (*e.g.* KD-tree [17], VP-tree [210]). In effect, they act at as non-linear projection but carry the computational advantage of a simple linear projection as they have linear decision models organized hierarchically. Extending to an ensemble of trees, forests offer a higher degree of flexibility as they can be seamlessly applied on high and low dimensional data and easily scalable to larger code sizes by growing the trees deeper or by adding more trees. However, the traditional approach of training random forests for classification (typically by maximizing information gain at each node) is agnostic to the notion of nearest neighbors (and class similarity within them) and can thus result in inconsistency within the hash codes generated by a particular tree for the same class data [212]. Towards this end, we propose to leverage cluster validity as a principled way of bringing similar class examples closer in the projected space, thus softly enforcing hash consistency. Further, as the trees are grown independently forest-based hashing schemes have good asymptotic properties and with additional bagging of the training samples, each tree explores a different facet of the dataset, which makes forest based hashing preferable for highly heterogeneous datasets (like neuron data) over methods that span the whole feature space.

For better scalability of retrieval to a growing heterogeneous database, we design hashing functions established upon unsupervised random forests called Hashing Forests (HF). The HF are trained to parse the input space in a hierarchical fashion and we demonstrate that HF can generate more sensitive code words by effectively utilizing its tree-structure and ensemble nature. Trees in HF are trained independently and they are more easily augmented for evolving databases. In comparison to random forest hashing method proposed by Yu and Yuan [212], we introduce an inverse coding scheme which effectively mitigates database-wide pairwise comparisons, which is better suited for fast large-scale hashing. Fig. 2.1 schematically illustrates the different methodological steps involved in using HF for neuron retrieval.

2.3 Mathematical Formulation

Hashing functions are used to encode records into hash codes, such that simple binary distance defined on the hash codes preserves similarity amongst the encoded records. Ideally, the hash function should generate compact and easy to compute representations, which can then be used for accurate search and fast retrieval [201]. In this work, we model hashing functions through *unsupervised* randomized forests ($\mathcal{H}$), which generates compact binary code blocks (*say,* $C_{\mathcal{H}}$) encoding the input feature space. We hypothesize that computationally cheaper binary distance measures (*such as hamming distance etc.*), defined between the generated code blocks correlates well with the inter-item similarity, and thus aiding in effective hashing. In the following sections, we discuss in detail the different stages involved in using hashing forests for encoding and retrieval within large-scale image databases.

2.3.1 Training phase

The hashing forest ($\mathcal{H}$) is an ensemble of binary decision trees which partitions the feature space hierarchically based on learnt binary oblique split functions. We introduce randomness through feature subspace bagging and bootstrapping to generate maximally decorrelated trees. The goal of a decision forest is to combine the predictions of several decorrelated trees built with different components in order to achieve high robustness in regards to noisy features.

Each hashing tree h^k is grown in an *unsupervised* fashion by recursively partitioning the feature set X_n, which reaches a particular node n into two subsets X_{2n} and X_{2n+1}. At each split node n, split functions ϕ_n are generated as shown in Eq. 2.2, which are randomly selected hyperplanes that split the feature space into two subsets. The hyperplane ϕ_n is parametrized by the parameter set θ_n, which comprise of the node-level feature-wise mean vector μ_n, the feature-wise standard deviation vector σ_n, and a vector of coefficients of individual features α_n along with an intercept scalar α_n^0. The values of μ_n and σ_n are estimated locally from training data X_n that reaches the node n.

The oblique split $\phi_n(\mathbf{x}, \theta_n)$ is defined as follows:

$$\phi(\mathbf{x}, \theta_n) = \left(\frac{\mathbf{x} - \mu_n}{\sigma_n}\right) . \alpha_n + \alpha_n^0 \tag{2.2}$$

In this work, we use randomized node optimization, generating a family of candidate splits ($\mathcal{F}_n$), where each split (say $\theta_c \in \mathcal{F}_n$) is multivariate and assigned randomly generated coefficient values (say α_c computed from a parameter hypersphere of radius 1 centred at the origin (*i.e.* $\sqrt{\sum |\alpha_c|.^2} = 1$). The intercept α_c^0 is generated as a random value between the minimum and maximum of $\left(\frac{\mathbf{x}-\mu_n}{\sigma_n}\right) . \alpha_c$. The coefficients are standardized by normalizing them to make their l_2 norm = 1 (*i.e.* $\sqrt{|\alpha_c|.^2 + |\alpha_c^0|^2} = 1$). Here, $\mathcal{F}_n$ is generated by randomly selecting `numVar` features from X_n at each split node and the candidate split function, which maximizes the node scoring function (given by Eq. (2.6)) is assigned to the split node in which:

$$\theta_n = \text{argmax}_{\theta_c \in \mathcal{F}_n} \mathcal{E}(X_n, \theta_c) \tag{2.3}$$

Using the above oblique split, the data set X_n is split into left and right subsets X_{2n} and X_{2n+1} by corresponding split functions as follows:

$$X_{2n} = \{\mathbf{x} | \mathbf{x} \in X_n \wedge \phi_n(\mathbf{x}, \theta_c) \leq 0\} \tag{2.4}$$

$$X_{2n+1} = \{\mathbf{x} | \mathbf{x} \in X_n \wedge \phi_n(\mathbf{x}, \theta_c) > 0\} \tag{2.5}$$

The choice of the optimal node split function is an interplay of two major factors namely, tree balance ($\mathcal{E}_B$) and cluster validity ($\mathcal{E}_C$), which are unified in defining $\mathcal{E}$ as shown below:

$$\mathcal{E}(X_n, \theta_c) = \underbrace{\mathcal{E}_C(X_n, \theta_c)}_{\text{Cluster Validity}} \times \underbrace{\mathcal{E}_B(X_n, \theta_c)}_{\text{Tree Balance}} \tag{2.6}$$

Cluster Validity

Recursive splitting of the input space can be modeled as a clustering operation, where similar data points are grouped together as we traverse down the tree. It is therefore appropriate to evaluate the splitting functions on how well they partition the feature space such that data elements within particular child node are more similar than across the other children nodes. This translates into two measurement criterion for cluster validity: (1) *Compactness* within the a child node and (2) *Separation* across other children nodes. The compactness is often associated with dispersion within a dataset allocated to a particular child node and the separation is measured as a distance measure between the datasets across other children nodes.

Towards this end, we evaluate the cluster validity $\mathcal{E}_C(X_n, \theta_c)$ of split, generated by a candidate split function ϕ_c, using the Krzanowski and Lai Index [2, 51, 101]. In Krzanowski and Lai Index, the compactness and separation of the candidate split function ϕ_c is evaluated together by minimizing the empirical distortion induced due to splitting the dataset into the children nodes as follows:

$$\mathcal{E}_C^{\text{KL}}(\theta_c, X_n) = 2^{2/\texttt{numFeat}} \left(\frac{1}{|X_n|} \sum_{\mathbf{x}_i \in X_n} \min_{j \in (X_{2n}, X_{2n+1})} (d_M(\mathbf{x}_i, \mathbf{c}_j)) \right) \tag{2.7}$$

where, $d_M(\mathbf{x}_i, \mathbf{c}_j)$ is the Mahalanobis distance. It is defined as

$$d_M(\mathbf{x}_i, \mathbf{c}_j) = \sqrt{(\mathbf{x}_i - \mathbf{c}_j)\Gamma^{-1}(\mathbf{x}_i - \mathbf{c}_j)^T} \tag{2.8}$$

where Γ is the covariance matrix estimated from X_n. $\mathbf{c}_j$ refers to the node specific centroid evaluated from the dataset allocated to a particular child node. This measures were evaluated for each candidate split function at the split nodes and contributed as the $\mathcal{E}_C$ term in the node scoring function (Eq. 2.6).

Tree Balance

Imbalance in an unsupervised tree is induced if the split divides the datasets into the children nodes in a skewed fashion, resulting in one child node encoding a larger data subset than the other node *i.e.* $|X_{2n}|> |X_{2n+1}|$ or $|X_{2n+1}|> |X_{2n}|$. We measure the degree of tree balance using a sigmoid function as follows:

$$\mathcal{E}_B(X_n, \theta_c) = \frac{2}{1 + e^{\gamma \cdot \tau(X_n, \theta_c)}} \tag{2.9}$$

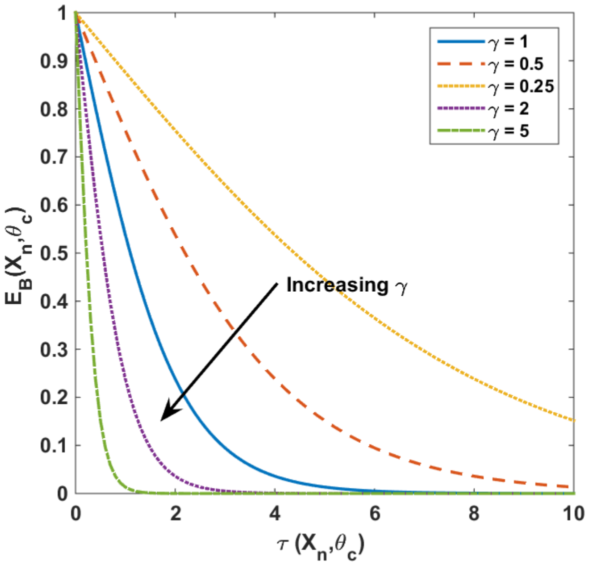


Fig. 2.2. *Variation of* $\mathcal{E}_B(X_n, \theta_c)$: The tree-balance factor is inversely related with increasing tree imbalance $\tau(X_n, \theta_c)$. For a high γ factor, $\mathcal{E}_B(X_n, \theta_c)$ penalizes the overall cost $\mathcal{E}(X_n, \theta_c)$ in Eq. 2.6 steeply with tree-imbalance. Reprint from [42], with permission of Springer.

where

$$\tau(X_n, \theta_c) = \max\left\{\left(\frac{|X_{2n}|}{|X_{2n+1}|} - 1\right), \left(\frac{|X_{2n+1}|}{|X_{2n}|} - 1\right)\right\} \tag{2.10}$$

In Eq. 2.9, γ is a hyper parameter that controls the importance of imposing tree balance while evaluating oblique splits. Increasing γ implies higher importance placed on tree-balance in Eq. 2.6 and this is illustrated in Fig. 2.2, where increasing γ penalizes more as tree imbalance increases.

The tree is grown through recursive splitting of the training dataset until the maximum defined tree-depth (treeDepth) is reached. We create an ensemble of such independently grown trees to create the hashing forest $\mathcal{H}$. Such a forest of `numTrees` binary trees with maximum depth of `treeDepth`, requires `numTrees` × `treeDepth` bits to encode each data-item. The time complexities to grow a tree and ensemble a forest are tabulated in Table 2.2 (S1-S2) [122].

Extension to non-Euclidean distances

In the proposed formulation, the oblique split defined in Eq. 2.2 falls under the family of hyperplane hashing based locality sensitive hashing methods. The theoretical guarantees of such methods applies only to certain metrics such as $l_p \in (0, 2]$ [195]. For extension of the proposed HF method to more complex metric spaces like weighted distance, power distance and other l_p distances, the splitting function has to be suitably defined to split data in that particular metric space. Typically, this can be done by considering a random sample reaching the split node as a pivot element and evaluating the metric distance of all other samples about this pivotal element. The split function can then be defined as a simple threshold over the obtained metric distances. The subsequent encoding and retrieval schemes proposed for the current HF formulation can be seamlessly extended for the non-Euclidean variants of HF.

S1	**Building tree**	$\mathcal{O}(\sqrt{N} * M * d) + \mathcal{O}(M * 2^{d-1})$	Training
S2	**Building forests**	$\mathcal{O}(T * (\sqrt{N} * M * d + M * 2^{d-1}))$	
S3	**Generating hash table**	$\mathcal{O}(T * d) + \mathcal{O}(M * S)$	
S4	**Generating Single Query Code**	$\mathcal{O}(T * d)$	Retrieval
S5	**Calculating Inter-item similarity with Forward Code**	$\mathcal{O}(M * S)$	
S6	**Calculating Inter-item similarity with Inverse Code**	$\mathcal{O}(T * d)$	
S7	**Quick sort**	$\mathcal{O}(M \log M)$	

Symbols: Code word Size $S = T(2^{d+1} - 2))$; Number of Trees T; Number of Features N; Tree Depth d; Retrieval Database Size M.

Tab. 2.2. *Time Complexity Analysis* of training and retrieval with unsupervised hashing forests. It must be noted that for the complexity for calculating the inter-item similarity with inverse coding is independent of database size M, which is the main factor contributing to scalability to large repositories. Reprint from [42], with permission of Springer.

2.3.2 Hash Table Generation

Given a trained tree (h^k) of the hashing forest $\mathcal{H}$, each data item n_i (*characterized by the input feature vector* $\mathbf{x}_i$) in the database is passed through it till it reaches the leaf node. For a tree h^k of depth treeDepth, the split and the leaf nodes are assigned breadth-first order indices (say n_k), which are associated with binary bit b_n^k in the code word $C_k(\mathbf{x}_i)$. For a particular data item, if node n_k is part of its path, then b_n^k is set to 1, otherwise, to 0. This leads to a $(2^{\texttt{treeDepth}+1} - 2)$ bit sparse code word $C_k(\mathbf{x}_i)$. It must however be noted that only treeDepth bits are required to generate the codeword as there are only $2^{\texttt{treeDepth}}$ possible traversal paths, each leading to a unique leaf node. We repeat the same process for every other tree in the forest to generate the sparse code block $C_{\mathcal{H}}(\mathbf{x}_i)$ of size $S = (\texttt{numTrees} \times (2^{\texttt{treeDepth}+1} - 2))$ for each data item.

For faster retrieval, we pre-compute the code blocks for all M data items in retrieval/training database $\mathcal{D}$ and generate a hash table of size $M \times S$. This is stored using $(M * (\texttt{numTrees} * \texttt{treeDepth}))$ bits along with traversal paths saved in a $(2^{\texttt{treeDepth}} * (2^{\texttt{treeDepth}+1} - 2))$ binary look-up table. However, as the database gets bigger, so will the time required for calculating the pairwise hamming distance between the codewords of all the data points in the dataset. The time complexity of generating the hash table for all the data items in the database is shown in Table 2.2 (S3). To address this problem, we further propose to generate the inverse codewords to improve the retrieval speed performance.

2.3.3 Inverse Coding

Each bit b_n^k in $C_{\mathcal{H}}$ encodes a unique input sub-space, which is constrained by the split functions of tree h^k leading to node n_k. In order to avoid pair-wise comparisons between the data items during retrieval in large databases, we formulate an inverse coding scheme. We transpose the hash table to generate the inverted hash table $\mathcal{I}$, which is a sparse $(S \times M)$ dimensional matrix. This implies that for feature vector $\mathbf{x}_i$, if bit b_n^k in $C_{h^k}(\mathbf{x}_i)$ is 1, then $\mathcal{I}(n_k, i) = 1$, and it belongs to the feature subspace encoded by b_n^k. Given a new *query* data item, instead of calculating the pairwise-similarity between all data items in $\mathcal{D}$, we extract the corresponding hash code from the hash table, which is a representation of similarity vector between the new point and all the other data points. Through the generation of the inverse hash table $\mathcal{I}$, we have effectively encoded the input subspaces along with associated data items.

2.3.4 Testing Phase

The path in which a data item traverses through the trained trees is used to define inter-item similarity. For a given *query* item n_q (with feature vector $\mathbf{x}_q$), the corresponding code block $C_{\mathcal{H}}(\mathbf{x}_q)$ is generated in a similar fashion to the Hash Table Generation phase. In the direct retrieval formulation, pairwise comparisons (*through hamming distance*) between $C_{\mathcal{H}}(\mathbf{x}_q)$ and code blocks of items (say $C_{\mathcal{H}}(\mathbf{x}_i)$ for item n_i) in the retrieval database $\mathcal{D}$ are made to evaluate inter-item similarity $\mathcal{S}(n_q, n_i)$ *i.e.*

$$\mathcal{S}(n_q, n_i) = \frac{1}{S} \sum_{\forall bits} (C_{\mathcal{H}}(\mathbf{x}_q) == C_{\mathcal{H}}(\mathbf{x}_i)) \tag{2.11}$$

If n_q generates the same code block as an item in $\mathcal{D}$ (*i.e. both belong to the same input subspace*), we assign perfect similarity to them ($\mathcal{S} = 1$). However, the pairwise comparison for large scale databases is computationally expensive as seen from its time complexity in Table 2.2 (S5). To mitigate this, in the inverse coding, we formulate the similarity function as $S_{\mathcal{I}} = \texttt{numTrees} * (\texttt{treeDepth} - 1)$ dimensional similarity accumulator cell $\mathcal{A}_{n_q}$. Given the code block $C_{\mathcal{H}}(\mathbf{x}_q)$ for the query item n_q, $\mathcal{A}_{n_q}$ is calculated as:

$$\mathcal{A}_{n_q}(i) = \frac{1}{S_{\mathcal{I}}} \sum_{\forall n_k} \mathcal{I}(n_k, i) \text{ if bit } b_n^k \text{ in } C_{\mathcal{H}}(\mathbf{x}_q) = 1 \tag{2.12}$$

The inter-item similarity $\mathcal{S}(n_q, n_i)$ is related to $\mathcal{A}_{n_q}$ as $\mathcal{S}(n_q, n_i) = \mathcal{A}_{n_q}(i)$. Such an inverse formulation is computationally more efficient for large databases (as seen from the order complexity in Table 2.2 (S6)), than the forward scheme (Table 2.2 (S5)). The inverse coding evaluates inter-item similarity by accumulating the membership of the database items from the columns of the inverse hash table corresponding to the tree nodes reach by that of the query item in HF. This effectively mitigates the need for pair-wise comparisons, leading to a time complexity that is independent of the database size M.

In the task of retrieving an ordered set of K most similar items from $\mathcal{D}$, we sort the items of the database in ascending order of inter-item similarity using quick-sort (time complexity of $\mathcal{O}(M \log M)$). The top $(2K)$ nearest neighbor items are further re-ranked according to their normalized Euclidean distance from the *query* item for better consistency (with an additional time complexity of $\mathcal{O}(KN + K \log K)$). For validation purposes, this re-ranking using normalized Euclidean distance is performed on all comparative methods and baselines considered in Section 2.4.

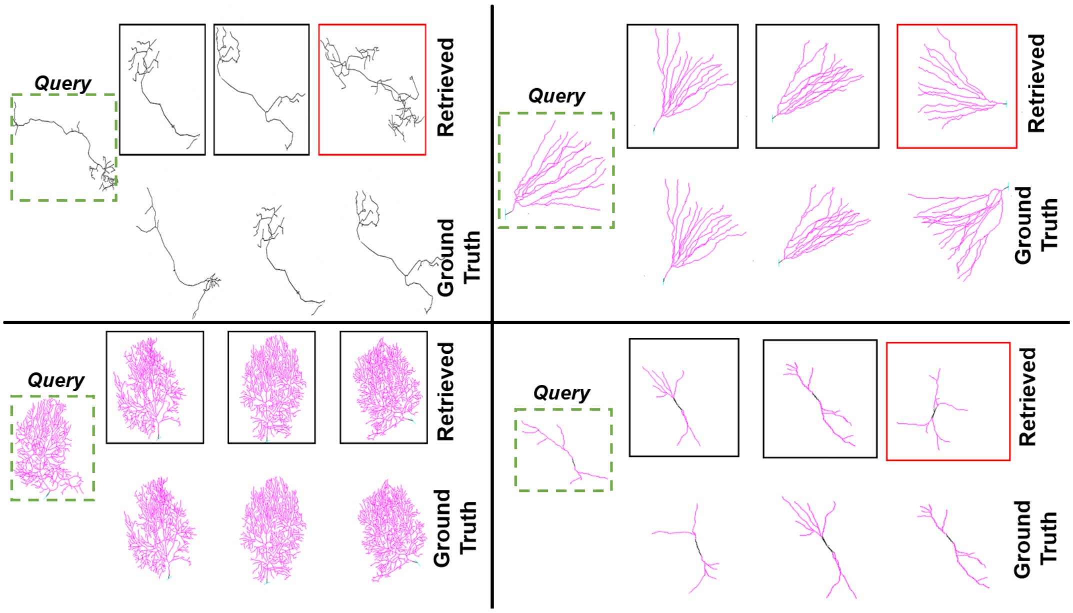


Fig. 2.3. *Visual evaluation of neuron retrieval*: For each query neuron on the left (boxed in green), the top-three neighbors retrieved with the proposed HF algorithm are shown along with ground truth neurons (using normalized Euclidean distance) are shown. The incorrect results are marked by red boxes. Reprint from [42], with permission of Springer.

2.4 Experiments and Results

2.4.1 Neuron Image Retrieval

In the context of neuroscientific databases, the desired morphological similarity preserving aspect of the hashing function implies that *morphologically similar neurons are encoded with similar binary codes*. This implies that for a particular *query* neuron (*say* n_q), the bucket of K morphologically similar neurons retrieved from the database $\mathcal{D}$ through hashing should be ideally as same as the K-nearest neighbors calculated using standardized Euclidean distance over the whole neuromorphological space. We refer the reader to Appendix A for further exposition on the task of pattern exploration in neuroscientific image databases. We used 31266 3D reconstructions of neurons extracted from 147 different archives, which curated from multiple laboratories and are publicly available on `http://neuromorpho.org` [9]. All archives listed as '*In the Repository*' in the list of archives in the Neuromorpho.org repository have been included in this study [226]. We employed the Lmeasure toolbox to extract 3D neuromorphological features, which characterize different aspects of neuron structure and topology [44, 166]. Fig. 2.3 illustrates the results of retrieval with hashing forests over the target database.

2.4.2 Evaluation Metrics

Successful similarity-preserving hashing in large-scale databases depends on the efficacy of the code word to compactly parse and represent the neuromorphological space as well as efficiently compute inter-neuron similarity using the generated hash codes. As part of validations, we use the following evaluation metrics:

Neighborhood Approximation

We introduced the *Neighborhood Approximation* (NA) graph in Mesbah et al. [130] to model how close the estimated neighborhood, computed from code words, (*from the hashing method*), approximates the true neighborhood around a item in the input space. For a particular hashing method, the NA for the j^{th} neighbor is defined as the average of the normalized Euclidean distances between the item and retrieved j^{th} neighbor for all items in $\mathcal{D}$. Let, for an item n_i (with feature vector $\mathbf{x}_i^0$), the j^{th} neighbor have a feature vector $\mathbf{x}_i^j$, then

$$\text{NA}(j) = \frac{1}{M_{\text{test}}} \left(\sum_{i=1}^{M_{\text{test}}} \varepsilon(\mathbf{x}_i^0, \mathbf{x}_i^j) \right) \tag{2.13}$$

where

$$\varepsilon(\mathbf{x}_i^0, \mathbf{x}_i^j) = \left(\sqrt{\frac{1}{N} \sum_{a=1}^{N} \left(\frac{x_{ia}^0 - x_{ia}^j}{s_a} \right)^2} \right) \tag{2.14}$$

$\varepsilon(\mathbf{x}_i^0, \mathbf{x}_i^j)$ is the standardized Euclidean distance between $\mathbf{x}_i^0$ and its j^{th} neighbor $\mathbf{x}_i^j$ with a^{th} feature standard deviation s_a estimated over the whole database (which is chosen for invariance to scales of different features). NA-graph is averaged over M_{test} test items from the testing dataset.

Retrieval Performance

We evaluate the retrieval performance by computing two metrics: Kendall's rank correlation coefficient κ and G_{mean}. The G_{mean} is often used in information retrieval algorithms to better understand the trade-off between precision and recall. Let $\mathcal{N}_\varepsilon(n_i)$ represents the set of items 'relevant' to the query item, which is the top K nearest neighbors defined upon the normalized Euclidean distance in the input space and $\mathcal{N}_\mathcal{H}(n_i)$ represents the retrieved items through hashing as a set of k similar items. The G_{mean} is calculated as an average over the test database and is calculated as:

$$\begin{align} G_{mean} &= \sqrt{Precision \times Recall} \tag{2.15} \\ &= \frac{1}{M_{\text{test}}} \sum_{i=1}^{M} \sqrt{\frac{|\mathcal{N}_\varepsilon(n_i) \bigcap \mathcal{N}_\mathcal{H}(n_i)|^2}{|\mathcal{N}_\mathcal{H}(n_i)| \times |\mathcal{N}_\varepsilon(n_i)|}} \tag{2.16} \end{align}$$

Kendall's rank correlation coefficient κ is used to measure the association between two ranked lists. We use this metric to evaluate the efficacy of ranking of relevant items. Given a pair of ranking lists

$$\left(r_1^R, r_1^T\right), \left(r_2^R, r_2^T\right), \cdots, \left(r_n^R, r_n^T\right)$$

(here, retrieved list (R) *vs.* true neighborhood list (T)). A pair of observations $\left(r_i^R, r_i^T\right)$ and $\left(r_j^R, r_j^T\right)$ are said to be concordant if the ranks on both lists agree *i.e.* $r_i^R > r_i^T$ and $r_j^R > r_j^T$ or $r_i^R < r_i^T$ and $r_j^R < r_j^T$. The pairs are deemed discordant if $r_i^R > r_i^T$ and $r_j^R < r_j^T$ or

$r_i^R < r_i^T$ and $r_j^R > r_j^T$. If $r_i^R = r_i^T$ and $r_j^R = r_j^T$, they are neither concordant nor discordant. The κ for the retrieval performance on test dataset is evaluated as follows:

$$\kappa = \frac{1}{M_{\text{test}}} \sum_{i=1}^{M_{\text{test}}} \frac{n_c^i - n_d^i}{n_c^i + n_d^i} \tag{2.17}$$

where n_c^i and n_d^i are the number of concordant and discordant pairs extracted from the respective R and T lists for each item n_i. In case of total agreement and disagreement between the two paired lists, the coefficient value is $\kappa = 1$ and under $\kappa = -1$, respectively.

Retrieval Time

Hashing aims at minimizing the time for retrieval by reducing expensive pairwise distance computations to cheaper binary operations defined over the hash codes (like `xor` for Hamming distance computation). As discussed in Section 2.3, we consider two different strategies for hash code comparison: (1) Forward Coding and (2): Inverse Coding. For a particular hashing method, the training time includes time required to train the hashing functions and generate the hash table for database. The testing time includes the time required for generating the hash codes for the query items, time for comparison (forward / inverse coding), sorting and ranking the approximate nearest neighbors. We incur an additional time overhead during testing, if the fetched neighbors are re-ranked according to their normalized Euclidean distance from the query item.

2.4.3 Comparative Methods

The main contribution of the hashing forests formulation presented in this chapter to design hashing forests is the introduction of oblique split functions and improvised node-scoring with cluster validity measures. Further, we formulate tree-traversal path based coding scheme as opposed to leaf-based scheme proposed in Yu and Yuan [212] for more efficient hierarchical parsing of the input space. These propositions lead us to four baselines to test the hypothesis that introducing these contributions improve hashing performance. The baselines are tabulated in Table 2.3. Each baseline differs in terms of the choice of the encoding scheme (leaf node/tree path encoding), the inclusion/exclusion of cluster validity and the type of the splitting function.

Comparisons to these baselines would support our hypothesis that oblique splits with cluster validity leads to better parsing of the input space, resulting in higher code efficiency. In addition, we validate the performance of our proposed algorithm (HF) by comparing it against popular large scale hashing methods discussed in Chapter 1, including Locality Sensitive Hashing (LSH) [180], Spectral Hashing (SH) [201], and Self taught hashing (STH) [218]. Additionally, as a baseline for comparison against hashing based approaches we include dimensionality reduction based retrieval methods, including Principal Component Analysis (PCA) and Neighborhood Preserving Embedding (NPE) [76]. In case of PCA and NPE, we used single-precision floating point representation for the embedding and retrieval was done by pairwise computation of Euclidean distance in the embedding space between the query item's embedding and that of the target database.

Baselines	**Encoding** **LN - Leaf node** **TP - Tree-path**	**Cluster Validity**	**Split Type**
Baseline 1 (BL1)	LN	×	Axis-aligned
Baseline 2 (BL2)	TP	×	Axis-aligned
Baseline 3 (BL3)	TP	×	Oblique
Baseline 4 (BL4)	LN	✓	Oblique
Proposed	TP	✓	Oblique

Tab. 2.3. *Hashing Forest Baselines*: To ablatively contrast the contributions within this chapter, baselines BL1-4 are defined with variations on the scheme of encoding (either using leaf-node encoding or tree-path encoding), if cluster validity was employed for choosing the optimal split function and the nature of the split function (linear or oblique). Reprint from [42], with permission of Springer.

γ	2.0	1.0	0.5	0.25	0.1	0.05
HF-S	51.09	**54.25**	51.34	52.87	53.80	52.96
HF-M	65.27	**68.04**	66.31	65.63	65.71	65.57
HF-D	67.18	**67.65**	67.15	63.51	64.20	64.10

Tab. 2.4. *Hyperparameter Selection for HF*: Contrasting three variants of HF *i.e.* HF-S, HF-M and HF-D, on basis of the tree-depth, while simultaneously fixing the code size at 128 bits, we observe the retrieval performance using G_{mean} metric as the tree-balance parameter γ is varied from 0.05 to 2. We observe a relative stability of performance at $\gamma = 1$ across the configurations and performance is best at moderate depth in contrast to shallow or very-deep trees. Reprint from [42], with permission of Springer.

2.4.4 Hyperparameter Selection for Hashing Forests

The main hyper parameters to be optimized for hashing forests include: tree balance parameter (γ), number of trees (`numTrees`), and their depth (`treeDepth`). For hashing forests, the hash code word size is given by `numTrees`×`treeDepth`. Fixing the code-size, we first optimize γ to be used in further analysis. For this, we fix the code-size at 128 bits and optimize γ for three configurations of HF: Shallow Trees (HF-S with `treeDepth` = 2), Moderately Deep Trees (HF-M with `treeDepth` = 4), and Very Deep Trees (HF-D with `treeDepth` = 8). The `numTrees` are chosen accordingly as 64, 32 and 16 respectively. The hyperparameter γ was varied as $[2.0, 1.0, 0.5, 0.25, 0.1$ and $0.05]$ with decreasing importance towards tree-balance. The G_{mean} for each of these configurations is tabulated in Table 2.4.

From Table 2.4, comparing HF-S, HF-M and HF-D, we infer that for sufficient depth, the performance of HF is invariant to choice of `numTrees` and `treeDepth`. We observe consistent optima at $\gamma = 1.0$ for all three tested configurations. Therefore, for the rest of validations, we fix the tree balance parameter γ at 1.0 and `treeDepth` at 4, corresponding to moderately deep

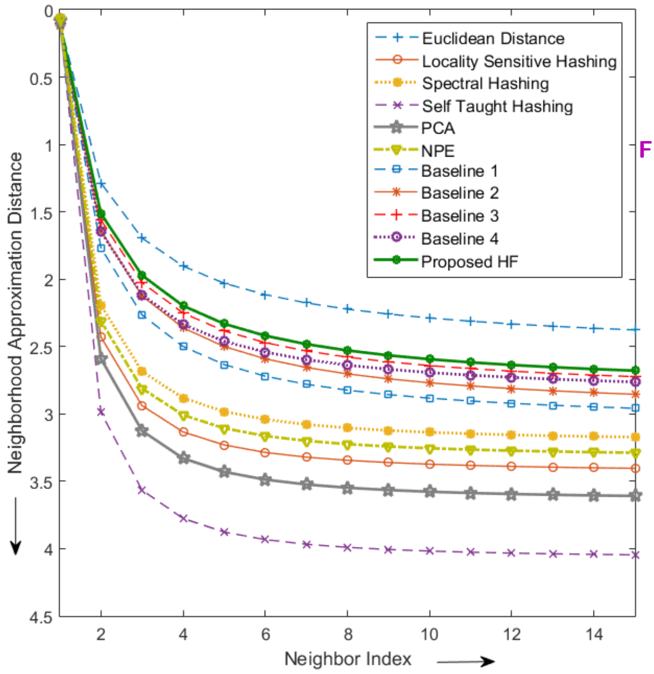


Fig. 2.4. *Neighborhood Approximation (NA) graph*: For a fixed codeword size (32 bytes), we observe that the proposed HF approximate the true neighborhood best. The true neighborhood is defined with the scaled Euclidean distance on the input space. Amongst the comparative methods, Spectral Hashing is observed to be the closest to HF. Reprint from [42], with permission of Springer.

trees. This observation is extendable to other code-sizes as trees are grown independently in a decorrelated fashion.

2.4.5 Neighborhood Approximation

In an unsupervised hashing setting, the distance function defined on the original input feature space (say, normalized Euclidean distance (NA$_{EUC}$)) is deemed to have the best neighborhood approximation. Thus, the hashing method that diverges the least from the NA$_{EUC}$ graph preserves the true neighborhood to the best possible extent. The NA graph is evaluated over the target database and the results for all comparative methods are reported in Fig. 2.4. For fair validation, we keep the size of the code-block fixed at 256 bits for this experiment. This evaluation is performed for all the baselines and comparative methods. The `treeDepth` for the baselines BL1-4 and HF was fixed at 4, thus leading to `numTrees` of 64.

2.4.6 Hashing retrieval performance vs. Code block size

We measure the G_{mean} and Kendall's κ statistic for all comparative methods as well as baselines by varying the code block size from 4 bytes to 64 bytes in geometric order of 2. We compare the performance for retrieval (both search and ranking) of the top 10 neighbored neurons using these methods for a heterogeneous test set of 800 randomly selected neurons (not included while training) and the results are tabulated in Table 2.5 and Table 2.6. This validation is performed to evaluate the improvement in similarity preserving aspect of the hash code with increasing code-size. It also serves to validate our hypothesis that introducing oblique splits with cluster validity and using whole tree-traversal path for encoding leads to more efficient hash codes over the baselines and comparative methods. Fig. 2.3 demonstrates the performance for 4 distinct neurons of differing morphologies with the closest neighbors retrieved using the proposed HF formulation and the ground truth (minimal normalized

Code Size	Comparative Methods					Baselines				Proposed
(in bytes)	LSH	SH	STH	PCA‡	NPE‡	BL1	BL2	BL3	BL4	
4	24.08	26.45	24.80	42.11	**48.20**	29.23	30.91	29.44	30.59	34.27
8	28.86	35.20	38.20	**57.15**	52.41	29.69	42.92	44.19	45.92	49.41
16	41.15	53.00	43.40	61.60	62.65	43.11	58.40	60.61	63.16	**69.51**
32	46.98	67.40	47.60	64.91	67.75	59.37	72.70	76.05	81.00	**83.13**
64	57.51	81.60	47.20	67.72	70.80	74.53	84.27	87.31	83.48	$\boxed{\mathbf{92.72}}$

‡ - These are non-hashing comparative methods (dimensionality reduction) using floating-point representation (1 float = 4 bytes).

Note: The best performance for a fixed code size is shown in **boldface**. and the best result amongst all the comparative methods is $\boxed{\text{frameboxed}}$.

Tab. 2.5. *Retrieval performance using G_{mean} with varying code size*: We observe that the retrieval performance for most of the methods increases substantially with increasing code-size, as the Hamming distance can better approximate the true neighborhood distance. Further, for code-size ≥ 16 bytes, the performance of unsupervised Hashing forests is observed to be superior to the comparative methods and baselines. Reprint from [42], with permission of Springer.

Euclidean distance). The HF was trained with $\gamma = 1.0$, `numTrees` = 64 and `treeDepth` = 4 and through visual evaluation, we observe close morphological similarity amongst the ground-truth neurons and its retrieved neighbors.

The time for retrieval is an important evaluation metric for retrieval using hashing. To compare and contrast the retrieval time against exhaustive pairwise distance computation, we report the time for training and testing for the comparative methods and baselines in Table 2.7. The training time includes the time to train the hash functions and extract the hashing table on the training data of 30466 neurons. The testing time includes the time for generation of the test hash codes and comparing it against the *a priori* extracted hash table using forward / inverse coding schemes for 800 test neurons. These algorithms were implemented on a general purpose 64-bit CPU with 16GB RAM memory and 2.7GHz Intel(R) Core(TM) i7-4600U processor. In case of retrieval with a mobile application, the actual retrieval time additionally depends on the data transfer speed and the hardware configuration of the mobile device.

2.4.7 Incremental training with database evolution

As the database evolves with addition of new data, the current form of the hashing function can be directly employed for populating the hash table with the new incoming data (method M1). Alternatively, the hashing functions can be retrained on the extended dataset with the additional new data (method M2). If the current code-size cannot sufficiently handle the added heterogeneity as the database evolves, the hashing functions can be augmented with further hash functions trained independently only on the additional new dataset (method M3) and appending the newly generated hash codes to the existing hash table. In case of HF, such a code augmentation translates to training additional independent hashing trees on the additional dataset and concatenating these to the tree ensemble of the existing HF.

Code Size	Comparative Methods					Baselines				Proposed
(in bytes)	LSH	SH	STH	PCA‡	NPE‡	BL1	BL2	BL3	BL4	
4	0.2266	0.2450	0.2280	0.3986	**0.4713**	0.2879	0.2949	0.2998	0.2913	0.3250
8	0.2875	0.3569	0.3307	**0.5628**	0.5135	0.2970	0.4015	0.4113	0.4270	0.4554
16	0.3970	0.4729	0.3742	0.6010	0.6218	0.3869	0.5355	0.5354	0.5787	**0.6505**
32	0.4344	0.7329	0.4049	0.6351	0.6703	0.5492	0.6867	0.7338	0.7847	**0.8149**
64	0.5382	0.7867	0.4200	0.6645	0.7081	0.7035	0.8230	0.8476	0.8108	$\boxed{\mathbf{0.9274}}$

Tab. 2.6. *Retrieval performance (Kendall's rank correlation coefficient κ) vs. Code block size.* : We observe that the retrieval performance for most of the methods increases substantially with increasing code-size, as the Hamming distance can better approximate the true neighborhood distance. Further, for code-size ≥ 16 bytes, the performance of unsupervised Hashing forests is observed to be superior to the comparative methods and baselines. ‡ - These are non-hashing comparative methods (dimensionality reduction) using floating-point representation (1 float = 4 bytes). **Note:** The best performance for a fixed code size is shown in **boldface** and the best result amongst all the comparative methods is $\boxed{\text{frameboxed}}$. Reprint from [42], with permission of Springer.

Alternatively, the hash functions can be retrained on extended dataset for a larger code-size (method M4). Comparing the alternative methods to handle database evolution, M2 and M4 are computationally expensive in comparison to M1 and M3. It must be noted that in scenarios where database evolution involves addition of new input features, the proposed and the comparative hashing methods can potentially be extended to multi-view formulations such as proposed in [118].

To evaluate the performance of different hashing functions as the database evolves, we create a test scenario wherein the initial hash functions are trained on 19886 neurons curated from 86 archives. The database evolution is modeled by addition of 5048 new neurons from 16 additional data archives to the initial dataset. The new incoming dataset is divided into non-overlapping training and testing datasets of 4548 and 500 neurons respectively. M1 is trained for a code-size of 32 bytes, M2 is retrained for the same code size as M1, M3 augments M1 with an additional 8 bytes making the code-size 40 and M4 is retrained for a code size of 40 bytes. The retrieval performance evaluated using G_{mean} score for top-10 neighbor retrieval for each of the proposed and comparative methods is tabulated in Table 3.4. To analyze the time overheads incurred during each of the four methods M1-M4, we also report the training time and the testing time (using inverse coding) in Table 2.8.

2.5 Discussion

In the previous section, we designed experiments to validate our hashing forest performance and perform comparative analysis with reference to other large-scale generalized hashing methods and baselines. We further discuss in detail the observations and inferences we draw from them in the following section.

	Exhaustive	**Code Size**	Comparative Methods					**Baselines**				**Proposed**
		(in bytes)	LSH	SH	STH	PCA‡	NPE‡	BL1	BL2	BL3	BL4	
Training (in s)	-	4	0.046	0.34	160.9	0.124	289.3	1.83	1.77	1.78	5.85	5.65
		8	0.081	0.49	320.8			3.23	3.41	3.43	11.52	12.06
		16	0.166	1.23	684.4			6.48	6.41	6.27	25.73	22.39
		32	0.270	2.07	1385.6			15.44	13.7	13.9	45.12	48.56
		64	0.606	3.612	1953.6			29.09	26.7	27.8	93.15	97.31
Testing(in s) Forward Coding	91.654	4	2.642	2.652	2.718	5.29	5.18	3.282	5.648	5.654	3.282	5.646
		8	4.213	4.228	4.357	8.46	8.77	5.036	8.705	8.705	5.035	8.708
		16	7.104	7.139	7.329	9.51	9.49	7.838	15.868	15.869	7.836	15.868
		32	13.226	13.808	13.808	14.86	14.95	14.255	31.406	31.401	14.256	31.407
		64	24.742	24.909	25.676	29.81	31.99	33.119	67.741	67.739	33.019	67.718
Testing(in s) Inverse Coding	-	4	0.328	0.338	0.414	-	-	0.416	0.709	0.715	0.416	0.707
		8	0.543	0.558	0.687			0.660	1.129	1.129	0.659	1.132
		16	0.951	0.986	1.176			1.063	2.131	2.132	1.061	2.131
		32	1.581	1.653	2.163			1.728	3.771	3.766	1.729	3.772
		64	3.142	3.309	4.076			4.222	8.619	8.617	4.222	8.616

‡ - These are non-hashing comparative methods (dimensionality reduction) using floating point representation (1 float = 4 bytes).

Tab. 2.7. *Time analysis of Hashing performance*: We observe that the time for training hashing forests is highly competitive in comparison to the comparative methods. In terms of testing time, the use of inverse-coding significantly improves the retrieval time across all the comparative methods and baselines. Reprint from [42], with permission of Springer.

2.5.1 Neighborhood Approximation

The NA graph evaluates how well a code word generated by particular hashing method is able to approximate the neighborhood around a query item with respect to neighborhood defined using normalized Euclidean distance. From Fig. 2.4, we observe a divergent trend (with reference to the ground-truth NA$_{EUC}$ graph) in the NA graphs of all methods as the neighbor index increases. For a fixed code size, the HF and other forest based baselines BL1-4 approximate neighborhood better than the comparative LSH, SH, and STH methods. This supports the hypothesis that effective utilization of the tree-structure along with ensemble nature of these methods improves data-driven parsing of the input space. Comparing neighborhood approximation of dimensionality reduction driven retrieval methods, we observe that NPE exhibits better NA over PCA as it effectively preserves local neighborhoods during embedding. In comparison to hashing methods, NPE demonstrates performance superior to STH and LSH and is comparable to SH.

Comparing BL2 with BL1, and proposed HF with BL4, we infer that using tree-traversal path encoding over leaf node encoding leads to better neighborhood approximation. This can be associated to the fact that complete decision path allows for a partial neighborhood contribution in calculation of inter-item similarity. This effect is illustrated in Fig. 2.5, where we consider two distinct items n_i and n_j which share nodes R, S_1 and S_4 during tree traversal. However, they reach different leaf nodes L_3 and L_4 respectively. The similarity metric between n_i and n_j defined with tree-traversal path-encoding is $\mathcal{S}(n_i, n_j) = 2/3$, as they shared $2/3^{\text{rds}}$ of the traversal path. In contrast, with leaf node encoding, $\mathcal{S}(n_i, n_j) = 0$, as they reach distinct

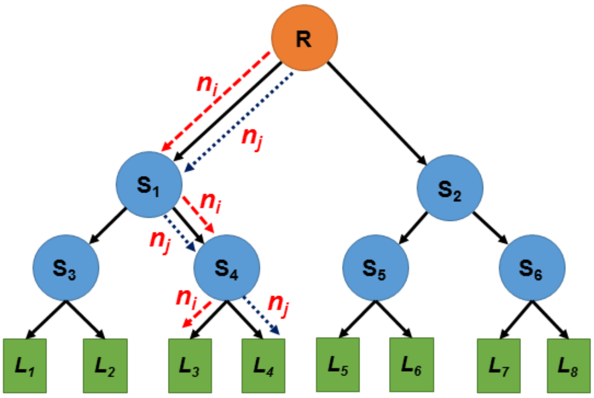


Fig. 2.5. *Illustration of partial neighborhood effect due to tree-traversal encoding*: Given two items n_i and n_j entering at root node **R** and traversing through the nodes $\mathbf{S}_1$ and $\mathbf{S}_4$ to be sorted into leaf nodes L_3 and L_4 respectively. Under leaf-node encoding, the similarity between the items (sim(n_i, n_j)), as approximated by such a hashing tree is 0 as they do not share the same leaf node. Under tree-traversal encoding, due to the partial sharing of the tree-traversal path (until $\mathbf{S}_4$, the similarity is sim$(n_i, n_j) = 0.66$. Reprint from [42], with permission of Springer.

leaf nodes. This partial neighborhood helps improving the neighborhood approximation of the hash codes. Finally, comparing the baselines BL2 and BL3 to proposed HF, we observe that the neighborhood approximation is improved when oblique splits (in HF over BL2) and cluster validity (in HF over BL3) are employed.

2.5.2 Hashing retrieval performance vs. Code size

Comparative Methods

We quantitatively evaluated the performance of the proposed method for different lengths of hash codes. It is clearly seen from Table 2.5 and Table 2.6 that the proposed HF performance improves as code length increases, and achieves better results consistently in comparison to other hashing based methods in searching and ranking relevant neurons. It must be noted that we chose larger code sizes over conventional code sizes (> 16 bytes), as it was observed that precision-recall performances for HF and comparative methods for smaller code sizes were not sufficient enough for the application at hand. In comparison, the dimensionality reduction based retrieval methods (PCA, NPE) exhibit superior retrieval performance for smaller code sizes (4-8 bytes) over hashing based methods. However, the retrieval performance for PCA and NPE does not improve significantly with increasing dimensionality of embedding with the inclusion of projections corresponding to lower eigen values.

By looking at the results obtained from comparative methods, we observe that the SH performs consistently better than the LSH and the STH. Though the LSH's performance steadily increases with increasing code-size, the improvement is considerably slower implying that LSH needs much higher code sizes for achieving comparable performance to the proposed HF or the SH, which will significantly increase the computational cost, resulting in delayed retrieval (corroborates observations reported by Yu and Yuan [212]). In case of the STH, as code-size increases, the eigenvectors corresponding to higher eigenvalues are utilized in defining the hashing function. This is not desirable because eigenvalues are often very noisy.

Baselines

As established in the discussion of NA Graph, tree-traversal path based encoding with its partial neighborhood effect demonstrates considerable improvement in retrieval performance. For code size of 64 bytes, we observe from Table 2.5, an overall increase of 9.74 % between BL2 and BL1 (84.27% from 74.53%) and 9.24% between the proposed HF and BL3 (92.72% from 83.48%). This trend is consistent in the ranking performance as the Kendall's κ statistic improvises by 0.1195 between BL2 and BL1 (0.8230 from 0.7035) and by 0.1166 between the proposed HF and its leaf-encoding baseline BL4. These observations further corroborate the hypothesis that partial neighborhood effect is desirable for effective retrieval of true neighbors. We also report considerable improvement of 8.45 % from 84.27% to 92.72% for the 64 byte code size, over our previous HF formulation [130] (BL2). This trend is consistently observed across all the other smaller code sizes too. These observations demonstrate the superiority of the proposed HF formulation over the baselines and validates our hypothesis that oblique splits with cluster validity improves code efficacy. The improved performance of BL3 in contrast to BL2 is due to the use of oblique splits. This can be attributed to the following aspects:

- Oblique splits can separate distributions that lie between the coordinate axes with a single multivariate split, which might have required deep nested axis-aligned splits otherwise;

- The learnt hashing trees are less biased to the geometrical constraints of the base learner if oblique splits are used (also observed by Menze et al. [129]).

Further, inclusion of cluster validity during training, ensured that the neighborhoods, resulting from clustering of similar neurons in the input space (as observed by Polavaram et al. [149]), are preserved during the generation of hashing forest splits. This has resulted in improved retrieval performance of the proposed HF in comparison to the nearest baseline BL3 (oblique splits without cluster validity).

Time Analysis

We profiled the training and the testing time for retrieval of the comparative methods for varying code-lengths for 10 trials with setting identical to Table 2.6 and tabulated the average observed time for training and testing in Table 2.7. In the comparison of the training times, we observe that all the methods except STH and NPE exhibit training time of under 100 seconds. The high training time of STH and NPE is attributed to the computationally expensive eigenvalue decomposition step (order complexity of $\mathcal{O}(M^3)$). Additionally in STH, the hash functions are independently trained binary support vector machine classifiers that are computationally expensive to train for large datasets (order complexity of $\mathcal{O}(SMN)$). During retrieval, we observe that employing inverse coding for hashing methods reduces the time for comparison and ranking significantly in comparison to forward coding and is significantly lower than exhaustive pairwise distance computation. Comparing to the baselines, we observe that BL1 and BL4 exhibit lower retrieval time in comparison to BL2, BL3 and the proposed method due to the difference in the encoding schemes employed for comparison (leaf node for BL1, BL4 and tree-path encoding for BL2, BL3 and the proposed method). Compared to other hashing methods, the proposed method with inverse coding has a higher retrieval time for the

	Comparative Method	M1 **Original** 32 bytes$^{\#}$	M2 **Retrained** 32 bytes	M3 **Augmented** 40 bytes$^{\$}$	M4 **Retrained** 40 bytes
Performance	LSH	40.23	42.93	46.13	47.10
	SH	45.32	46.18	47.22	51.40
	STH	41.44	42.80	43.15	44.54
	BL2‡	47.87	49.20	49.77	53.43
	Proposed	**68.20**	**70.03**	**71.47**	**73.37**
Training Time (in s)	LSH	0.098 (0.281)	0.265	0.119 (0.078 + 0.041)	0.292
	SH	0.030 (1.263)	1.697	0.192 (0.074 + 0.118)	2.38
	STH	3.085 (900.6)	1177.8	67.36 (51.32 + 16.04)	1662.7
	BL2‡	0.379 (7.94)	10.69	1.944 (0.58 + 1.364)	17.8
	Proposed	0.318 (30.107)	41.27	3.603 (2.171 + 1.432)	67.98
Testing Time (in s)	LSH	1.012	0.961	1.176	1.305
	SH	1.157	1.215	1.413	1.243
	STH	1.319	1.467	2.055	2.230
	BL2‡	2.681	2.582	2.972	3.156
	Proposed	2.474	2.756	3.282	3.151

‡ - Prior art method [130]

$^{\#}$ - $\tau_1^{M1}(\tau_2^{M1})$ - τ_1^{M1} is the time required to infer hash-codes the new incoming dataset using existing hash functions (τ_2^{M1} is the time required for training the existing hash functions, however it is not deemed as a part of the training time for M1).

$^{\$}$ - Total training time for M3 is $\tau^{M3} = (\tau_1^{M3} + \tau_2^{M3})$ where τ_1^{M3} is the time to train the augmented hash codes on the incoming dataset and τ_2^{M3} is the time required to repopulate the existing dataset through the augmented hash functions.

Tab. 2.8. *Time-analysis of training and testing for Hashing Forests and Comparative Methods under scenarios of incremental learning*: Specifically, for the proposed hashing forests, the performance increases upon retraining to to a larger code size (M4) and also upon augmentation (M3) to existing code sizes. Reprint from [42], with permission of Springer.

same code size, but is significantly lower than pairwise comparison used in dimensionality reduction methods.

2.5.3 Incremental training with database evolution

With addition of new unseen data to the database, we evaluate variants of hashing methods (retraining *v.s.* augmentation) that have been proposed in Section 2.4 and report their retrieval performance (G_{mean}) in Table 3.4. From an overall perspective, we conclude that augmenting hashing functions with additional bits (M3) performs comparably to retraining (M4) and is superior to retrieving with the the original hash function (M1). Further, the proposed HF demonstrates significantly higher retrieval performance over the comparative hashing methods (LSH, STH, SH and BL2) which is highly desirable as the database continually evolves.

From Table 2.8, we observe that time overhead for training/augmenting the hash codes for M1 and M3 are relatively lower in comparison to retraining based methods (M2 and M4). These observations are concurrent with the expected trends as M1 involves no additional learning of the hash functions and M3 learns the augmentation hash function on a relatively smaller incoming dataset for a smaller code size. In comparison, M2 and M4 involves retraining the entire hash functions on the extended dataset (existing dataset + incoming dataset). In terms of the testing time, we observe that M1 and M3 are comparable to M2 and M4 respectively, as the time complexity for inverse coding is linear in terms of code-size and independent of the search database size.

3 Metric Hashing Forests

> *A year spent in artificial intelligence is enough to make one believe in God.*
>
> — **Alan Perlis**
> (First Recipient of the Turing Award in 1966)

3.1 Overview and Publications

This chapter presents the contributions of this thesis concerning learning and validation of metric Hashing Forests (mHF), which are a supervised variant of random forests tailored for the task of nearest neighbor retrieval through hashing. mHF is achieved by training independent hashing trees that parse and encode the feature space such that local class neighborhoods are preserved and encoded with similar compact binary codes. At the level of each internal node, locality preserving projections are employed to project data to a latent subspace, where separability between dissimilar points is enhanced. Following which, we define an oblique split that maximally preserves this separability and facilitates defining local neighborhoods of similar points. By incorporating the inverse-lookup search scheme within the mHF, we can then effectively mitigate pairwise neuron similarity comparisons, which allows for scalability to massive databases with little additional time overhead. Exhaustive experimental validations on 22,265 neurons curated from over 120 different archives demonstrate the superior efficacy of mHF in terms of its retrieval performance and precision of classification in contrast to state-of-the-art hashing and metric learning based methods. We conclude that the proposed method can be utilized effectively for similarity-preserving retrieval and categorization in large neuron databases.

This chapter is organized as follows: in Sec. 3.2 we present the premise for introducing supervision into the training of hashing forests and how metric learning can be leveraged for the same. Particularly, the aspects related to training and testing mHF are detailed in Sec. 3.3, wherein we present our novel optimization objectives that enforce code-consistency within hashing by evaluating class separability and neighborhood quality locally at each node (in Sec. 3.3.1). We also discuss the training of a metric Hashing tree in Sec. 3.3.2 and how the hash tables are subsequently generated and leveraged for retrieval in Sec. 3.3.3. Following this, in Sec. 3.4, we present the results of evaluating mHFs for the task of large-scale semantics preserving retrieval on neuron image databases. We particularly compare and contrast against multiple ablative baselines (in Sec. 3.4.2), against non-hashing based methods (mostly metric learning and dimensionality reduction approaches in Sec. 3.4.3) and against state-of-the art supervised hashing based comparative methods (in Sec. 3.4.4). We conclude Sec. 3.4 with Sec. 3.4.6, where we present a detailed analysis of time and memory costs associated with

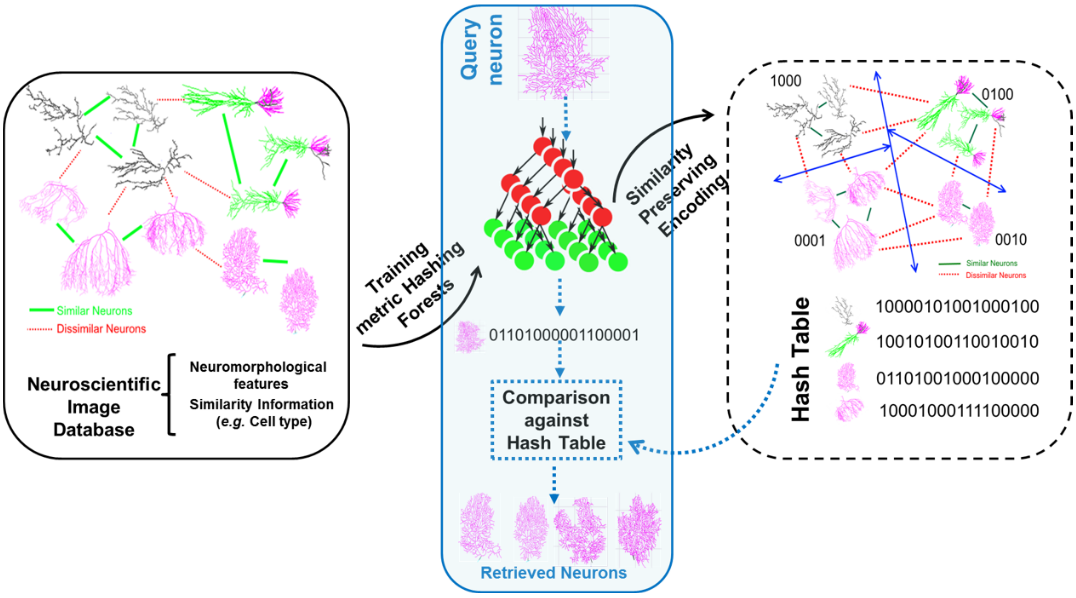


Fig. 3.1. *Schematic illustrating the training and testing of metric Hashing Forests towards the task of neuron image retrieval*: Given a database of neurons (defined with the neuromorphological space), individual trees are trained to partition the feature space with metric learning employed at each split node to choose an optimal split function. The tree traversal path is used to generate the hash codes, which are aggregated into a hash table representation. During testing, the query neuron traverses the learnt mHF and the generated hash code is compared against the hash table to rank and retrieve similar neurons which share morphological similarities and preserves desired semantics. Reprint from [40], with permission ©Elsevier.

the training and testing of the comparative methods. In the subsequent section (Sec. 3.5), we present detailed discussions on the comparison with the baselines, hashing and non-hashing based comparative methods with a detailed analysis of the retrieval performance in terms of accuracy as well as ease of retrieval.

Substantial parts of this chapter have already been published in the following article and are quoted verbatim:

[40] **Conjeti, Sailesh**, Amin Katouzian, Anees Kazi, Sepideh Mesbah, David Beymer, Tanveer F. Syeda-Mahmood, and Nassir Navab. "Metric hashing forests." *Medical image analysis 34 (2016): 13-29.*

Copyright Statement. ©2016 Elsevier B.V.

3.2 Motivation

In this chapter, we extend hashing with forests into the domain of classification by image retrieval and propose a supervised variant of HF called metric Hashing Forests (mHF). The extension of HF into the mHF is done by incorporating local metric learning and principled node optimization in order to maximize class separability. The main advantage is that the hashing functions generate codewords in which similar samples will have similar encodings. Figure 3.1 provides an overview of how the proposed mHF are trained and leveraged for neuron retrieval and classification. In the context of Fig. 3.1, the primary objective of mHF is to enable meaningful retrieval of neurons similar to the query and we use supervisory information to learn hash bits that are both discriminative and consistent. The task of classification of a new neuron into one of the considered cell-types should be seen as an additional outcome of effective retrieval.

As pointed out by Weiss et al. [201], an ideal hashing function has to be easy to compute for a novel input, should be compact, and preserve similarity. Our contributions in the context of the aforementioned requirements are as follows:

- **Out-of-sample Extension**: For a novel input, the code is generated by traversing the mHF trees. Each internal node of the tree carries a binary decision function, which determines the direction of propagation to children nodes until a terminal node is reached. As the original ambient feature space may not be optimal for class similarity-preserving retrieval, we introduce node-level metric learning through locality preserving projections (LPPs). These are linear approximations to Laplacian eigenmap embeddings that are defined on the entire ambient space, thus guaranteeing seamless extension to novel inputs.

- **Compactness**: At each internal node, the mHT splits the training dataset into two equally sized children subsets (by partitioning at the median), which ensures maximal code efficiency. Towards this end, we introduce the concept of class separability for choosing the node split function in a principled fashion. The cost function encourages splits that preserve the class similarity constraints and penalizes the loss of neighborhood quality. While growing hashing trees, the feature space is obliquely parsed to separate class distributions that lie between feature axes and the constrained subspaces generated are encoded with compact binary codewords (shown in Figure 3.6).

- **Similarity Preserving**: By local metric learning, we transform the data from the original ambient feature space to a latent space, where maximal separability between dissimilar data points is achieved (shown in Figure 3.4). Additionally, the partitioning function is defined on the latent space in such a way that class separability becomes maximal. Such a partitioning of the feature space is useful for preserving local similar class neighborhoods and encoding samples within neighborhoods of similar binary codewords.

As the main focus of this work is inclusion of metric learning within previously developed framework of hashing forests [130], we briefly overview the state-of-the-art in metric learning based techniques.

The key concept behind metric learning is to learn a distance function defined on the feature space that obeys class-similarity (supervised) or geometrically imposed constraints (unsupervised/manifold learning). In the context of similarity-preserving retrieval, the given supervisory class information is cast into pairwise similarity/dissimilarity constraints and the distance metric is learned under these constraints. Such a metric is often parameterized as matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$ that can be used to define distance between two samples $\mathbf{x}_i$ and $\mathbf{x}_j$ as:

$$d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \, \mathbf{M} \, (\mathbf{x}_i - \mathbf{x}_j) \tag{3.1}$$

If $\mathbf{M}$ is symmetric and positive semi-definite, it can be factorized as $\mathbf{M} = \mathbf{L}^T\mathbf{L}$, where $\mathbf{L} \in \mathbb{R}^{e \times d}$ and $e \geq rank(\mathbf{M})$. Using such a decomposition of $\mathbf{M}$, the distance function $d_\mathbf{M}$ can be equivalently written as:

$$d_\mathbf{M}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{L}^T\mathbf{L}\,(\mathbf{x}_i - \mathbf{x}_j) \tag{3.2}$$

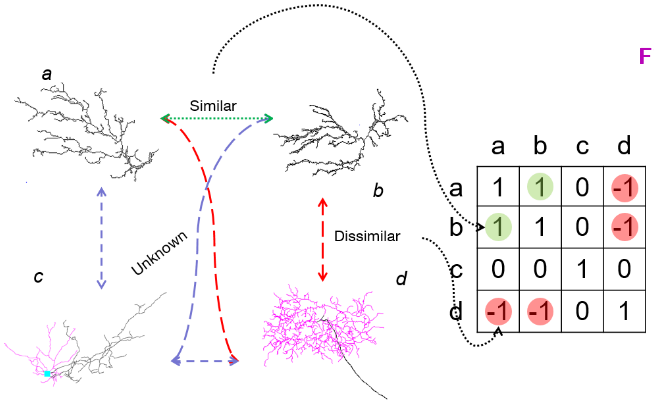
This can be interpreted as the Euclidean distance between points $\mathbf{Lx}_i$ and $\mathbf{Lx}_j$ defined on a new latent space.

Generalized metric learning methods can be broadly categorized into *supervised* and *unsupervised* approaches. Supervised techniques attempt to learn a projection such that similar samples are kept closer while simultaneously maximizing the distance between dissimilar samples. In the particular context of retrieval, this is highly desirable as the approximate kNN neighborhood would maximally correspond to the true class neighborhood. Some popular supervised metric learning methods that are driven by nearest neighbors include neighborhood component analysis (NCA) by Goldberger et al. [65], largest margin nearest neighbors (LMNN) by Weinberger and Saul [199], supervised neighborhood preserving embedding (sNPE) by He et al. [76], locality sensitive discriminant analysis (LSDA) by Cai et al. [21], locality preserving projections (LPP) by He and Niyogi [75] to name a few[1]. In this chapter, we consider sNPE, LSDA, and LPP as representative supervised metric learning methods in our comparative analysis of retrieval performance. With regards to baselines, for evaluating the overall improvement induced by considering supervised information in retrieval performance, we also include unsupervised metric learning methods based on principle component analysis (PCA) and Mahalanobis distance for comparison. The underlying hypothesis is that the learned distance metric $\mathbf{M}$ (or $\mathbf{L}$) is more task relevant and the approximate nearest neighbors defined on the projected space are closer to the true class neighbors in comparison to the ones defined on the ambient feature space. For a detailed exposition on metric learning, interested readers are directed to the surveys by Kulis [105] and Bellet et al. [15].

3.3 Methods

The rationale behind the proposed mHF is to parse the feature space and encode samples into binary codes (called hash codes), such that the Hamming distance defined between the hash codes of similar data pairs is minimized and simultaneously maximized on dissimilar pairs. In brief, the ultimate goal in designing the mHF retrieval algorithm is to ensure that samples that share similar class and characteristics to that of the query are fetched in a time-efficient manner.

[1] For sNPE, there exists an unsupervised metric learning counterpart (uNPE) which is also considered in this work.



	a	b	c	d
a	1	1	0	-1
b	1	1	0	-1
c	0	0	1	0
d	-1	-1	0	1

Fig. 3.2. *Schematic illustrating encoding of class similarity/dissimilarity in matrix* $\mathcal{S}$: Neurons a and b are similar (*i.e.* $\mathcal{S}(a,b) = \mathcal{S}(b,a) = 1$). They are dissimilar to Neuron d (*i.e.* $\mathcal{S}(a,d) = \mathcal{S}(b,d) = -1$). Class information of Neuron c is unknown (*i.e.* $\mathcal{S}(a,c) = \mathcal{S}(b,c) = 0$) Reprint from [40], with permission ©Elsevier.

3.3.1 Class Separability and Neighborhood Quality

Given a data set $\mathcal{X} = \left\{\mathbf{x}_i \in \mathbb{R}^d\right\}_{i=1}^N$, an $N \times N$ dimensional similarity matrix $\mathcal{S}$, like in Figure 3.2, is defined as follows:

$$\mathcal{S} = \begin{bmatrix} s_{11} & & \cdots & & s_{1n} \\ & \ddots & & & \\ \vdots & & s_{ij} & & \vdots \\ & & & \ddots & \\ s_{n1} & & \cdots & & s_{nn} \end{bmatrix} \tag{3.3}$$

$$\text{where } s_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_i, \mathbf{x}_j \text{ are similar} \\ \text{-1} & \text{if } \mathbf{x}_i, \mathbf{x}_j \text{ are dissimilar} \\ 0 & \text{o.w.} \end{cases}$$

The objective of a similarity-preserving hash function h is to map $\mathbb{R}^d \mapsto \{1,0\}^1$ such that similar pairs are assigned the same binary values and dissimilar pairs are assigned to different ones. Such a hashing function improves the local neighborhood of the samples as the distance metric (say standardized Euclidean distance) is now defined on samples that share the same hash bit than in the original sample distribution.

In the context of similarity preserving retrieval, we leverage this concept to evaluate the class separability induced by hashing functions. This is akin to the concept of node purity in traditional random forests [19, 46]. Prior to delving into how we evaluate the class separability, we introduce the neighborhood quality function $\mathcal{Q}$ in the context of k nearest neighbor retrieval. For a dataset $\mathcal{X}$ with similarity matrix $\mathcal{S}$, we evaluate the local neighborhoods around the samples and define the a neighborhood quality function $\mathcal{Q}$ as follows:

$$\mathcal{Q}(\mathcal{X}, \mathcal{S}, M) = \frac{1}{nk} \sum_{i=1}^{n} \sum_{t=1}^{k} s_{it} . p(\mathbf{x}_i, \mathbf{x}_t, M) \tag{3.4}$$

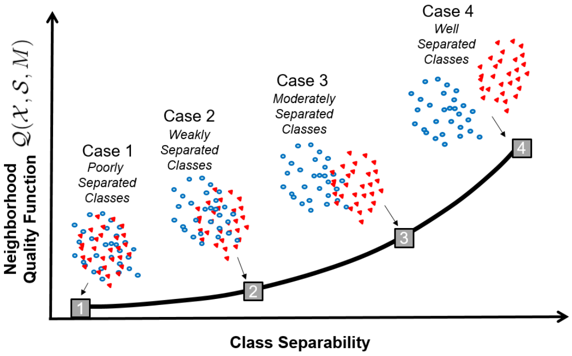


Fig. 3.3. *Schematic the neighborhood quality function* $\mathcal{Q}$: With improving class separability (*i.e.* local neighborhoods are of the same class), the neighborhood quality function $\mathcal{Q}$ increases and is hence used as a measure to rank candidate split functions. Reprint from [40], with permission ©Elsevier.

where $\mathbf{x}_t$ is the t^{th} neighbor of $\mathbf{x}_i$ and $\mathbf{x}_i, \mathbf{x}_t \in \mathcal{X}$. In the above quality function, the proximity between two samples $\mathbf{x}_i$ and $\mathbf{x}_t$ is evaluated as:

$$p(\mathbf{x}_i, \mathbf{x}_t, M) = e^{-\frac{d(\mathbf{x}_i, \mathbf{x}_t)}{\sigma_i \sigma_t}} \tag{3.5}$$

$$\text{where } d(\mathbf{x}_i, \mathbf{x}_t) = (\mathbf{x}_i - \mathbf{x}_t)^T M (\mathbf{x}_i - \mathbf{x}_t)$$

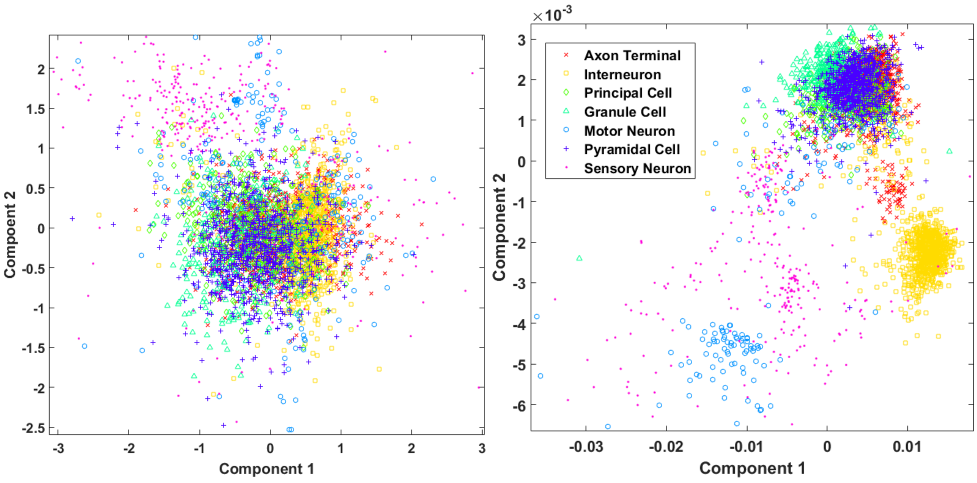
where σ_i and σ_t are local scaling factors. As defined in Zelnik-Manor and Perona [217], $\sigma_i = d(\mathbf{x}_i, \mathbf{x}_J)$, where $\mathbf{x}_J$ is the J^{th} neighbor of $\mathbf{x}_i$. The incorporation of local scaling allows for self-tuning of the point-to-point distances according to the local neighborhoods around the points $\mathbf{x}_i$ and $\mathbf{x}_t$. In addition to local scaling, we also use M, which is an appropriately defined distance metric on feature space $\mathcal{X}$ (*e.g.* $M = I$ for Euclidean distance, $M = C^{-1}$ for Mahalanobis distance, where C is the inverse of the covariance matrix). Alternatively, M can be learned from the data through metric learning approaches.

By looking at the quality function $\mathcal{Q}$, we observe that the pair of nearest neighbor points contribute positively if they are similar and penalize it if they are dissimilar. The behavior of $\mathcal{Q}$ is qualitatively illustrated in Fig. 3.3 for varying degrees of class-separability. Starting from poorly separated classes (Case 1) and progressively improving class-separability (towards Case 4: Well separated classes), we observe that the neighborhood quality function $\mathcal{Q}$ increases with better class separability. Thus, higher values of $\mathcal{Q}$ implies that the local class neighborhoods are well separated within a particular dataset and thus better guarantee of generalization for class similarity preserving k nearest neighbor based retrieval towards new unseen test data.

We further leverage $\mathcal{Q}$ to evaluate the improvement in class separability that a particular hash function h induces while dividing the training dataset $\mathcal{X}$ into two subsets $\mathcal{X}_1$ and $\mathcal{X}_0$ comprising of samples that are alloted binary values 1 and 0, respectively. The similarity matrix is partitioned into two components $\mathcal{S}_1$ and $\mathcal{S}_0$, which are defined on $\mathcal{X}_1$ and $\mathcal{X}_0$ respectively. The improved class separability $\Delta\mathcal{Q}(h)$ is evaluated as:

$$\Delta\mathcal{Q}(h, M) = \frac{|\mathcal{X}_0|}{|\mathcal{X}|}\mathcal{Q}(\mathcal{X}_0, \mathcal{S}_0, M) + \frac{|\mathcal{X}_1|}{|\mathcal{X}|}\mathcal{Q}(\mathcal{X}_1, \mathcal{S}_1, M) \tag{3.6}$$

where $|\mathcal{X}_1|$, $|\mathcal{X}_0|$, and $|\mathcal{X}|$ are the number of samples in $\mathcal{X}_1$, $\mathcal{X}_0$, and $\mathcal{X}$, respectively. Comparing two hashing functions h_1 and h_2 (for a fixed M), if $\Delta\mathcal{Q}(h_1, M) > \Delta\mathcal{Q}(h_2, M)$, then it implies that h_1 leads to subsets with better class separability and the retrieval in the neighborhood constrained by h_1 is superior to h_2. Also, if $\Delta\mathcal{Q}(h, M) > 0$, it implies better generalization of similarity-preserving retrieval in the search space constrained by h than in the original feature



(a) PCA (b) LPP

Fig. 3.4. *Effect of local metric learning on class separability*: The neurons of different cell types mapped into a 2-dimensional space. **Fig. 3.4a:** Representation using first two components of principal component analysis and **Fig. 3.4b:** Results of supervised locality preserving projections. Each color and marker type corresponds to a neuron cell type. Under the LPP transformation, we observe that the cell types are better separated in comparison to the PCA space. Reprint from [40], with permission ©Elsevier.

space. In this chapter, we use $\Delta\mathcal{Q}$ for evaluating split functions defined at the internal nodes of the hashing trees (discussed in Section 3.3.2).

3.3.2 Training metric Hashing Tree (mHT)

The goal of mHF is to concatenate piecewise binary hash codes generated using several decorrelated hashing trees to get a composite hash code for a particular sample. Each metric hashing tree h^j is trained independently with a bagged dataset $\mathcal{X}^j$ generated by Monte Carlo sampling of the training dataset $\mathcal{X}$ (both training samples and features). Thus, the trees are trained to be maximally decorrelated which helps reducing the redundancy in the generated hash codes. The trees are grown by recursive partitioning of the feature set $\mathcal{X}_n$ that reaches a particular node n into two subsets $\mathcal{X}_{2n}$ and $\mathcal{X}_{2n+1}$. This node-level partitioning is governed by a split function ϕ_n that is generated to maximize the class separability between the two subsets (discussed in Section 3.3.1). At each node level, we learn a locality preserving projection (LPP) of the data that reaches the node using the algorithm proposed in [75]. Such a projection is learned to discover the underlying non-linear local manifold structure and to project it along directions that maximizes separability between dissimilar pairs and simultaneously cluster similar data pairs. This is demonstrated in Figure 3.4 for the neuroscientific image data and its constituent cell types. Defining the split function ϕ_n in the projected space is expected to induce better class separability between the two children subsets over splitting in the original feature space.

Locality Preserving Projection

Let $\mathcal{X}_n = \left\{\mathbf{x}_i \in \mathbb{R}^d\right\}_{i=1}^{m}$ denote a subset of the training data that reaches internal node n of the mHT. Let $\mathcal{S}_n$ denote the similarity matrix associated with it. In this context, we leverage the LPP to learn a linear transformation matrix A_n that maps the m points in $\mathcal{X}_n$ to a set of points in the latent subspace $\mathcal{Y}_n = \{\mathbf{y}_i \in \mathbb{R}^l\}_{i=1}^m$ $(l \leq d)$ such that $\mathbf{y}_i = A^T\mathbf{x}_i$. The LPP is an optimal linear approximation to the eigenfunctions of the Laplace-Beltrami operator on the manifold constituted by $\mathcal{X}_n$ and preserves the neighborhood structure of the data set while embedding.

In order to find the optimal linear embedding, a weighted undirected graph G with symmetric weights W is constructed on $\mathcal{X}$, such that edges connect similar points together. The task of learning a projection using LPP is to find a mapping that minimizes the distance between connected points in the projected space, and it is mathematically formulated as:

$$\min_A \sum_{ij} (\mathbf{y}_i - \mathbf{y}_j)^2 W_{ij} \tag{3.7}$$

where $\mathbf{y}_i = A^T\mathbf{x}_i$. Using standard spectral graph theory, [75] cast this problem as $\min_A A^T\mathcal{X}_n L \mathcal{X}_n^T A$. Here, L is the Graph Laplacian matrix of W, which is estimated as $L = D - W$, where D is a diagonal matrix such that $D_{ii} = \sum_j W_{ij}$. To avoid the trivial solution of $A = 0$, they imposed an additional constraint that $A^T\mathcal{X}_n D \mathcal{X}_n^T A = 1$. Thus, the estimation of optimal transformation matrix A_n reduces to the following minimization problem:

$$A_n = \arg\min_A A^T\mathcal{X}_n L \mathcal{X}_n^T A \text{ s.t. } A^T\mathcal{X}_n D \mathcal{X}_n^T A = 1 \tag{3.8}$$

The optimal transformation A_n that minimizes this objective function is obtained by the minimum eigenvalue solution to the generalized eigenvalue problem $\mathcal{X}_n L \mathcal{X}_n^T A = \lambda \mathcal{X}_n D \mathcal{X}_n^T A$. We chose the eigenvectors corresponding to the smallest l non-zero eigenvalues to constitute A_n. With this learned transformation matrix A_n, we project $\mathcal{X}_n$ to $\mathcal{Y}_n$, which is subsequently used to define the split function ϕ_n. The advantage of LPP in the context of retrieval is two-fold as follows:

- LPP discovers and preserves the local class neighborhood structure while embedding.
- It is simple, linear, fast, and more importantly defined everywhere in the ambient feature space, thus, enabling a natural out-of-sample extension to new unseen test data.

Split Function

Each split node n in the mHT is governed by a partitioning function ϕ_n, which divides the training dataset $\mathcal{X}_n$ that reaches it into two children subsets $\mathcal{X}_{2n}$ and $\mathcal{X}_{2n+1}$. The split functions are defined on the learned projective space $\mathcal{Y}_n$ (using A_n learned from LPP on $\mathcal{X}_n$) and parameterized as oblique hyperplanes. We use oblique hyperplanes over traditional axis aligned split functions as they are better suited to separate distributions that lie between the coordinate axes, which might have otherwise required deep nested axis-aligned splits [129]. Particularly in the context of encoding with hashing trees, the structure of the tree determines the number of bits required to encode the feature space spanned by it (code size is directly proportional to tree depth) and deeper trees with axis aligned splits require longer codes in order to be as effective as shallower trees with oblique split.

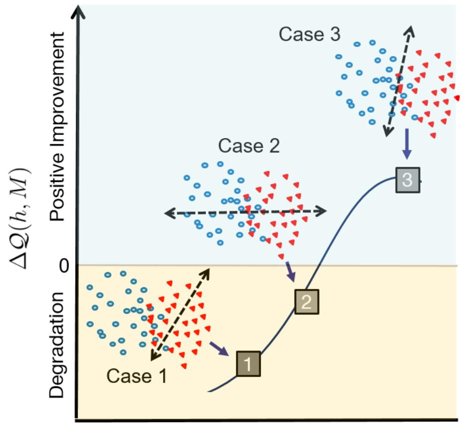


Fig. 3.5. *Schematic illustrating the randomized node optimization*: Given multiple candidate node-partitioning functions, the optimal choice is made by evaluating the degree to which such a function partitions the training data into balanced homogeneous children nodes. Reprint from [40], with permission ©Elsevier.

The hyperplane ϕ_n is parametrized by the parameter set θ_n, which comprises of coefficients of individual features α_n along with an intercept scalar α_n^0. The oblique split $\phi_n(\mathbf{y}, \theta_n)$ is defined as $\phi(\mathbf{y}, \theta_n) = \alpha_n^T \mathbf{y} + \alpha_n^0$. Using this oblique split, the data set $\mathcal{X}_n$ is split into left and right subsets $\mathcal{X}_{2n}$ and $\mathcal{X}_{2n+1}$ by corresponding split functions as follows:

$$\mathcal{X}_{2n} = \{\mathbf{x}|\mathbf{x} \in \mathcal{X}_n \wedge \phi_n(A_n^T\mathbf{x}, \theta_c) \leq 0\} \tag{3.9}$$

$$\mathcal{X}_{2n+1} = \{\mathbf{x}|\mathbf{x} \in \mathcal{X}_n \wedge \phi_n(A_n^T\mathbf{x}, \theta_c) > 0\} \tag{3.10}$$

An ideal splitting function at each node induces maximum class separability between the two children subsets ($\mathcal{X}_{2n}$ and $\mathcal{X}_{2n+1}$), so that dissimilar data points are assigned to different tree branches and are hence alloted different encoding codewords (and simultaneously assign similar data points to the same tree branch). As the task of k-NN retrieval is highly non-linear, no closed form solution exists that generates an optimal split and therefore we resort to greedy node optimization to generate ϕ_n (akin to traditional random forests [46]). We generate a family of candidate splits ($\mathcal{F}_n$), where each split (say $\theta_c \in \mathcal{F}_n$) parametrized by randomly generated coefficient values (say α_c). The coefficients were individually generated at random from a normal distribution with zero mean and unit standard deviation and normalized such that $\sqrt{\sum |\alpha_c|.^2} = 1$. This is equivalent to generating the coefficients at random from a parameter hypersphere of radius 1 centered at the origin (*i.e.* $\sqrt{\sum |\alpha_c|.^2} = 1$).

For maximum code efficiency while encoding, it is necessary that the trees grow in a balanced fashion, implying that $|\mathcal{X}_{2n}| = |\mathcal{X}_{2n+1}|$. Thus, the intercept α_c^0 is generated as the median value of $\alpha_c^T \mathbf{y}$. From the candidate splits $\mathcal{F}_n$, the split function that induces the maximal improvement in neighborhood quality (evaluated using Eq. (3.6)) is assigned to the

$$\begin{aligned}\theta_n =& \text{argmax}_{\theta_c \in \mathcal{F}_n} \Delta \mathcal{Q}(\theta_c, A_n^T A_n) \\ =& \text{argmax}_{\theta_c \in \mathcal{F}_n} \begin{pmatrix} \frac{|\mathcal{X}_{2n}|}{|\mathcal{X}_n|} \mathcal{Q}(\mathcal{X}_{2n}, \mathcal{S}_{2n}, A_n^T A_n) \\ + \\ \frac{|\mathcal{X}_{2n+1}|}{|\mathcal{X}_n|} \mathcal{Q}(\mathcal{X}_{2n+1}, \mathcal{S}_{2n+1}, A_n^T A_n) \end{pmatrix}\end{aligned} \tag{3.11}$$

This is schematically illustrated in Figure 3.5, where the candidate split that creates maximal class separability between the two ensuing children subsets is chosen as optimal. In Case 1, the candidate split spans entirely over a particular class resulting in a degradation of the neighborhood quality as a significant number of similar pairs are assigned to different child nodes resulting in inconsistent encoding. In Case 2, the candidate split sub-optimally divides the dataset as dissimilar pairs are assigned to the same child-node resulting in a hash function that is not discriminative. Finally, in Case 3, the candidate split induces maximal class-separability amongst the resulting children nodes and would thereby be preferred as a split function over the candidate splits in Cases 1 and 2.

The tree is further grown through such a recursive splitting of the training dataset until the maximum defined tree-depth (treeDepth) is reached. The parsing of the feature-space by an oblique tree is shown in Figure 3.6. We create an ensemble of such independently grown trees $\{h^1, \cdots, h^j, \cdots, h^{\texttt{numTrees}}\}$ to form the hashing forest $\mathcal{H}$. Such a forest of `numTrees` binary trees with maximum depth of `treeDepth`, requires `numTrees` × `treeDepth` bits to encode each sample.

3.3.3 Hash Table Generation

Given a trained tree (h^j) of the hashing forest $\mathcal{H}$, each data sample $\mathbf{x}_i$ in the training database, is passed through the hashing forest until it reaches the leaf node. For a tree h^j of depth `treeDepth`, the leaf nodes are assigned breadth-first order indices (say n_j), which are associated with binary bit b_n^j in the codeword $C_j(\mathbf{x}_i)$. If the data sample reaches a particular leaf node n^j in the j^{th} tree, the bit b_n^j is set to 1, otherwise to 0. This leads to a $2^{\texttt{treeDepth}}$ bit sparse codeword $C_j(\mathbf{x}_i)$ for the encoded sample $\mathbf{x}_i$ through tree h^j. Figure 3.6 depicts the encoding of the feature space parsed by a hashing tree of `treeDepth` = 3. It must however be noted that only `treeDepth` bits are required to generate the codeword as there are only $2^{\texttt{treeDepth}}$ possible traversal paths, each leading to a unique leaf node. We repeat the same process for every other tree in the forest to generate the sparse code block $C_{\mathcal{H}}(\mathbf{x}_i) = [C_1(\mathbf{x}_i), \cdots, C_{\texttt{numTrees}}(\mathbf{x}_i)]$ of size $S = (\texttt{numTrees} \times 2^{\texttt{treeDepth}})$ for each data sample. For faster retrieval, we pre-compute the code blocks for all N samples in retrieval/training database $\mathcal{X}$ and generate a hash table $C_{\mathcal{H}}(\mathcal{X})$ of size $N \times S$. This is stored using $(N * (\texttt{numTrees} * \texttt{treeDepth}))$ bits along with traversal paths saved in a $(2^{\texttt{treeDepth}} * 2^{\texttt{treeDepth}})$ binary look-up table.

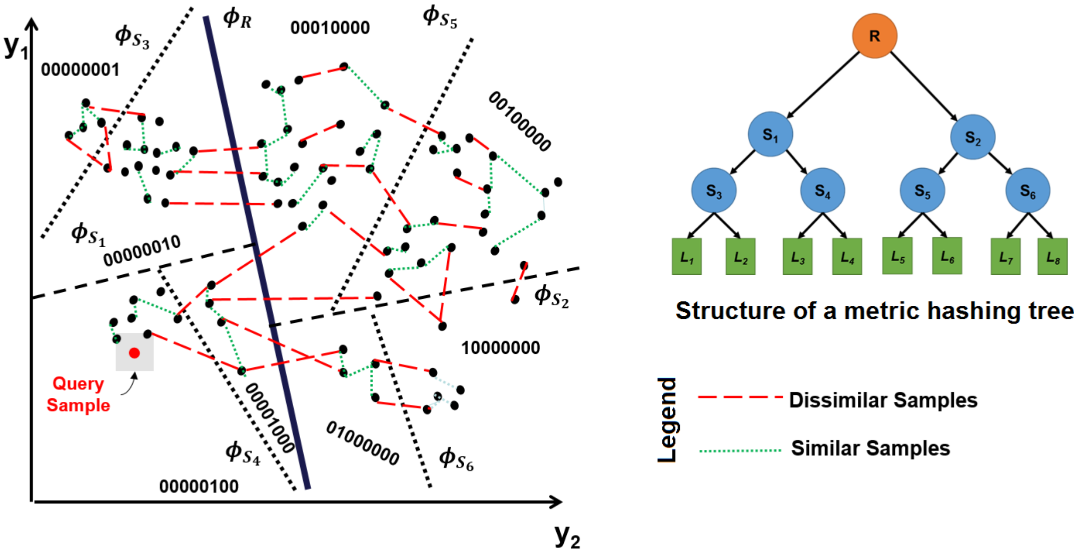


Fig. 3.6. *Schematic illustrating parsing and encoding of the feature space by a metric hashing tree*: Starting from the root-node R, the feature space is recursively split by partitioning functions ($\phi_{S_{1-6}}$), which are chosen to induce the maximal improvement in class-neighborhood quality ($\Delta\mathcal{Q}$). A query sample (shown in red) traverses down the tree and is alloted the sparse codeword corresponding to the leaf node that it reaches. Reprint from [40], with permission ©Elsevier.

Baselines	Local Metric	Class Separability Criterion	Split Type
Baseline 1 (BL1)	Eye	×	Univariate in original feature space
Baseline 2 (BL2)	Eye	×	Oblique in original feature space
Baseline 3 (BL3)	Eye	✓	
Baseline 4 (BL4)	PCA	×	Oblique in metric space space Oblique in original feature space
Baseline 5 (BL5)	PCA	✓	
Baseline 6 (BL6)	LPP	×	
Proposed (mHF)	LPP	✓	

Tab. 3.2. *metric Hashing Forest Baselines*: To evaluate the individual contributions within mHF, six baselines (BL1 - BL6) are set ablatively on choices of the local metric used (Eye / PCA / LPP), if class-separability criterion was employed and the nature of the node partitioning function (univariate / oblique). Reprint from [40], with permission ©Elsevier.

3.3.4 Baselines

The main contributions of the mHF formulation presented in this chapter over our previously proposed formulation [130] are: (a) inclusion of local metric learning and (b) improvised node-partitioning function optimization with the introduction of class separability criterion. To validate the overall improvement due to local metric learning, we consider two baseline transformations: I (identity matrix) and PCA basis that are learned in an unsupervised fashion from the data reaching each split node. Additionally, we validate the performance improvement by introducing the class separability criterion through contrasting against purely random balanced oblique splits at each split node (no node optimization). A combination of these methods form six baselines that are tabulated in Table 3.2. Comparisons to these baselines would support our proposed hypothesis that training hashing forests with local metric learning and class separability optimization criterion leads to more meaningful transformation and parsing of the feature space, thus resulting in higher code efficiency and discriminative retrieval performance.

3.3.5 Semantics-preserving Neuron Image Retrieval

We used 22265 3D reconstructions of neurons extracted from about 120 different archives, which curated from multiple laboratories and are publicly available on `http://neuromorpho.org` [9]. The neurons were each classified into seven major cell types (with database compositional percentages), namely, axon-terminal neurons (4.82%), inter-neurons (53.76%), principal cells (4.29%), granule cells (2.67%), motor neurons (1.38%), pyramidal cells (31.36%) and sensory neurons (1.71%). This labeling is motivated by the differences in the morphology between these major cell types. It must be noted that Neuromorpho.org (accessed on December 2015) curated around 34.5K neurons. The ontological scheme followed by Neuromorpho.org [227] involves either structural or functional classification of neurons. Functional classification is mainly based on the type of neuron-transmitter used by the neuron. As this study focuses on anatomical classification, we excluded around 12.3K neurons from the database that were classified functionally. Therefore in total about 22.2K neurons that met this inclusion criterion are used for further validations within this study.

We employed the Lmeasure toolbox to extract 3D neuromorphological features, which characterize different aspects of neuron structure and topology [44, 166]. In studies on morphological diversity and discovery of biologically relevant clusters [211], it is desirable to retrieve neurons that belong to the same neuronal cell type, which share similar morphological attributes [145]. This acts as a major motivation factor towards incorporating the requirement of class-similarity preservation into our retrieval framework. In this chapter, we use the class information to generate the similarity matrix $\mathcal{S}$ (Eq. 3.3), assuming that pairs of samples are similar if they share the same label and dissimilar otherwise. As pointed out in Bellet et al. [15], this approach of deriving the constraints prior to learning the metric is widely accepted and quite often never challenged. A more refined (but computationally expensive) approach towards generation of $\mathcal{S}$ was proposed by Wang et al. [194], where they discover constraints and learn the metric using an alternating optimization framework. We direct interested readers towards Appendix A for a more detailed exposition on aspects of neuron image retrieval and the neuromorphological space.

Parameters	Settings	Applicable to
Depth of the tree(`treeDepth`)	4	All baselines and mHF
Code Size	$8, 16, \cdots, 96$	
Number of trees (`numTrees`)	$2, 4, \cdots, 24$ (Code Size / `treeDepth`)	
Bagging Ratio (`bagRatio`)	1/`numTrees`	
Number of bagged features (`numFeat`)	$\lceil\sqrt{d}\rceil$	
Number of Candidate Splits (`numSplit`)	50	BL3, BL5‡ and mHF‡
	1	BL1, BL2, BL4‡, BL6‡
Number of variables in split (`featSplit`)	1	BL1
	`numFeat`	BL2, BL3, BL4‡, BL5‡, BL6‡ and mHF‡
Number of nearest neighbors (k)	1	All (BL3, BL5 and mHF in `GenerateSplit`)
Local scaling factor (J in Eq. 3.5)	5	BL2 - 6 and mHF

Tab. 3.3. *Parameter settings for the baselines and proposed method*: The parameters for proposed mHF and the associated baselines (BL1 - BL6) are chosen such that the resultant models are comparable in model complexity with sufficient learnable parameters for the problem at hand. Here, ‡ implies the parameter is defined on the metric learned space. Reprint from [40], with permission ©Elsevier.

3.3.6 Configuration Settings

The parameter configurations for the baselines (BL1 - BL6) and the proposed mHF method are tabulated in Table 3.3. It must be noted that in training the baselines and mHF, the parameter choice of k nearest neighbors is as same as the one used in retrieval. In practice, the local scaling factor is empirically fixed at $J = 5$ and such an intermediate value (not too small or large) is deemed acceptable. The algorithms were implemented using MATLAB R2015b on a general purpose 64-bit CPU with 16GB RAM memory and 2.7GHz Intel(R) Core(TM) i7-4600U processor.

3.4 Experiments and Results

3.4.1 Evaluation Metrics

Successful hashing in neuroscientific databases depends on the efficacy of the codeword to compactly parse and represent the true class neighborhood around the neurons in the neuromorphological space. Let $\mathcal{N}_{\mathcal{H}}(n_q)$ represents the set of neurons retrieved from the database $\mathcal{X}$ through hashing that fall within the k^{th} nearest neuron neighborhood of query neuron n_q *i.e.* $\mathcal{N}_{k,\mathcal{H}}(n_q) = \{n_i | d_{\mathcal{H}}(\mathbf{x}_i, \mathbf{x}_q) \leq d_{\mathcal{H}}(\mathbf{x}_k, \mathbf{x}_q), \mathbf{x}_i \in \mathcal{X}, n_i \mapsto \mathbf{x}_i\}$, where $\mathbf{x}_k$

corresponds to the feature vector of the k^{th} nearest neuron to n_q. Let $c_q \in \mathcal{C}$ represent the class label of neuron n_q. Let $\mathcal{X}_t$ refer to the test dataset. We quantify the retrieval performance using following metrics:

- **Retrieval Precision** (RP): It is defined as the ratio of the total number of similar neurons amongst the retrieved neighbors to the total number of retrieved neighbors for a particular value of k-NN. RP is evaluated as:

$$\text{RP}_{\mathcal{H}}^{k} = \frac{\sum_{\forall n_q \in \mathcal{X}_t} \left|\mathcal{N}_{k,\mathcal{H}}^{c_q}(n_q)\right|}{\sum_{\forall n_q \in \mathcal{X}_t} \left|\mathcal{N}_{k,\mathcal{H}}(n_q)\right|} \tag{3.12}$$

  where $\mathcal{N}_{k,\mathcal{H}}^{c_q}(n_q) = \{n_i | n_i \in \mathcal{N}_{k,\mathcal{H}}(n_q) \ \& \ c_i = c_q\}$ refers to the subset of neurons within the neighborhood of n_q that share the same class as that of n_q.

- **Classification Accuracy** (CA): For classification through retrieval, we assign the predicted class $c_{k,\mathcal{H}}^{\text{pred}}(n_q)$ for a query neuron n_q using maximum a posteriori class evaluated from the top k nearest neighbors retrieved through hashing with method $\mathcal{H}$. Let $\mathcal{C}$ represent the set of all possible classes of neurons in the database $\mathcal{X}$ and $\mathbf{x}_q$ the features of neuron n_q.

$$c_{k,\mathcal{H}}^{\text{pred}}(n_q) = \underset{c \in C}{argmax}\ p(c|\mathbf{x}_q, k, \mathcal{H}) \tag{3.13}$$

$$\text{where } p(c|\mathbf{x}_q, k, \mathcal{H}) = \frac{\left|\mathcal{N}_{k,\mathcal{H}}^{c}(n_q)\right|}{\left|\mathcal{N}_{k,\mathcal{H}}(n_q)\right|}$$

  The CA for a particular hashing method is evaluated as the ratio of the number of correctly classified test samples to the total number of samples in the testing dataset and is estimated as:

$$\text{CA}_{\mathcal{H}}^{k} = \frac{\sum_{\forall n_q \in \mathcal{X}_t} \left|c_{k,\mathcal{H}}^{\text{pred}}(n_q) == c^{\text{true}}(n_q)\right|}{|\mathcal{X}_t|} \tag{3.14}$$

In the subsequent experiments, $k = 1$ nearest neighbors are used for retrieval and classification as it does not require cross-validating parameters. Unless specified, to make each evaluation metrics statistically meaningful, we calculate them as an average of 10 experimental runs, which differ from one another due to the randomization involved in defining the training and testing datasets and in training the hashing forests (BL 1 - 6 and mHF).

3.4.2 Validations against baselines

In this section, we validate the incorporation of local metric learning (Eq. 3.8) and node optimization with neighborhood quality improvement (Eq. 3.6, 3.11) by contrasting retrieval performance of the proposed mHF against baselines listed in Table 3.2. We vary the code-sizes from 16 bits to 96 bits in increments of 8 bits and evaluate the overall performance of baseline and mHF (configuration settings presented in Table 3.3) using RP (Eq. 3.12) and CA (Eq. 3.14) metrics. Following a 5-folded cross-validation strategy, for each fold 80% of the total dataset was reserved as training and the rest non-overlapping 20% is used as an evaluation set. We

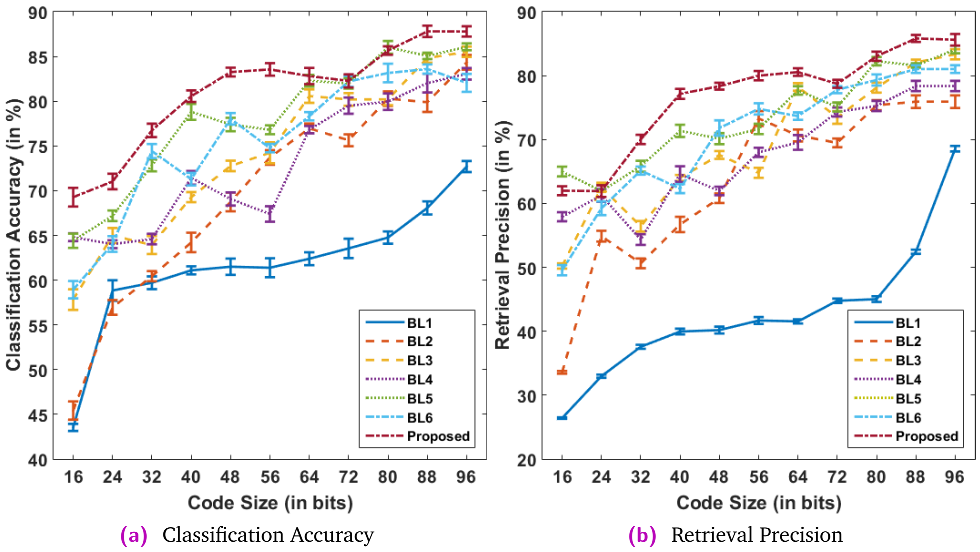


Fig. 3.7. *Evaluation of the proposed mHF vs. other hashing forest baselines varying code-size*: **Fig. 3.7a:** Overall accuracy of classification by retrieval and **Fig. 3.7b:** Precision of retrieval through hashing. With improved performance of the proposed method across varying code sizes, we observe that the contributions of node-wise metric learning (contrasting with BL2-3), choice of metric learning (LPP *vs.* PCA (BL-5)) and the use of class-separability criterion for optimizing (*vs.* BL6) are significantly contributing towards improved hashing performance. In particular, contrasting with BL-1(unsupervised hashing forests), we observe a significant margin with all the other learning based methods. Reprint from [40], with permission ©Elsevier.

report the average performance of CA and RP metrics with the 95% confidence interval error bars in Fig. 3.7a and Fig. 3.7b, respectively. We also perform the two sample Kolmogrov-Simrnov test to assess the statistical significance of the margins and it is deemed significant if p-value$< 10^{-3}$.

3.4.3 Comparative analysis against non-hashing methods

Hashing based methods generate compact binary representations of feature vectors, which facilitate scalability to massive databases and high dimensional feature spaces. In this section, we perform a comparative analysis between popular non-hashing methods employed for retrieval against the proposed mHF technique. These include l_p- norm based methods (Euclidean (p = 2); Chebyshev (p = ∞), and Minkowski distance (p = 1)); unsupervised metric learning methods (Mahalanobis, Principal Component Analysis PCA, unsupervised Neighborhood Preserving Embedding uNPE [76]), and supervised metric learning methods (supervised Neighborhood Preserving Embedding sNPE [76], Locality Sensitive Discriminant Analysis LSDA [21] and Locality Preserving Projections LPP [75]). For comparison, we present the results of mHF with code-size of 96 bits (`treeDepth` = 4 and `numTrees` = 24). In Figure 3.8, we illustrate the average and the 95% confidence intervals of the overall classification accuracy for $5-$folded cross-validation using the same data splits in Section 3.4.2. We also refer back to the observations in Figure 3.7a for comparative analysis.

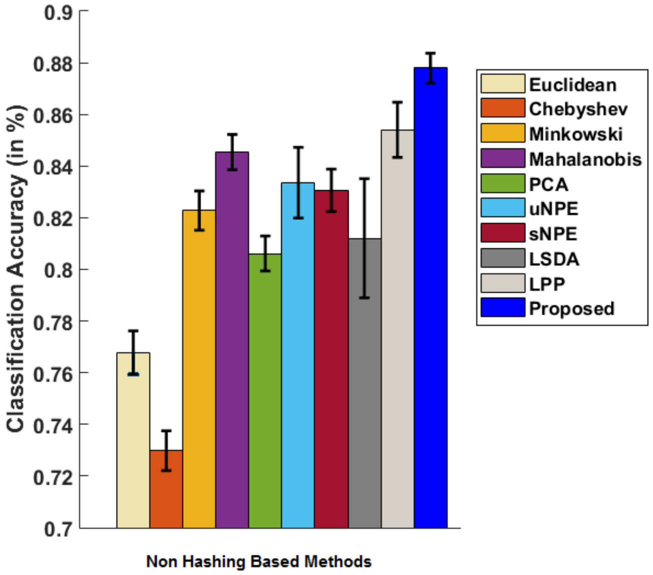


Fig. 3.8. *Comparison of the proposed mHF's classification accuracy vs. state-of-the-art non-hashing based retrieval methods*: These include l_p-norm based methods (Euclidean (p = 2); Chebyshev (p = ∞) and Minkowski distance (p = 1)); unsupervised metric learning methods (Mahalanobis, PCA, uNPE), and supervised metric learning methods (sNPE, LSDA and LPP). In terms of retrieval performance, LPP is the closest to mHF . The supervised metric learning methods are observed to be superior to their unsupervised counterparts and significantly better than performing retrieval in the native input space. Reprint from [40], with permission ©Elsevier.

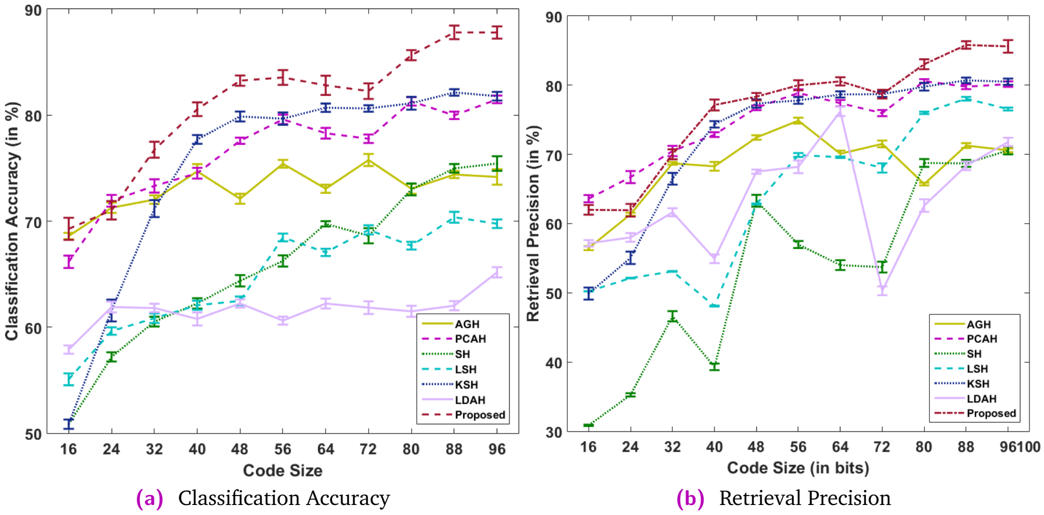
3.4.4 Comparative analysis against hashing methods

In this section, we evaluate the performance of mHF in comparison to state-of-the-art generalized hashing methods that are suited for large-scale retrieval. These include data independent methods (*e.g.* Locality Sensitive Hashing LSH [64, 180] *etc.*), data-driven unsupervised methods (*e.g.* Spectral Hashing SH [201], Anchor Graph Hashing AGH [119], Principal Component Analysis-based variant of LSH (PCAH) *etc.*) and data-driven supervised methods (*e.g.* Linear Discriminant Analysis based variant of LSH (LDAH), Kernel Supervised Hashing KSH [120] *etc.*). We quantify the retrieval performance (using CA and RP), varying the code-size from 16 to 96 bits in increments of 16 bits. Similar to the earlier validations, Figure 3.9a and 4.5b, we illustrate the average and the 95% confidence intervals of CA and RP, respectively, for $5-$folded cross-validation using the same data splits employed in Section 3.4.2.

3.4.5 Evaluation of class-wise retrieval performance

We evaluate the performance of mHF towards retrieval of the individual classes, which in this particular application are the seven distinct neuron types based on neuromorphology: axon-terminal, interneuron, principal cell, granule cell, motor neuron, pyramidal cell and sensory neuron. The mHF is compared against the following methods:

- BL2: the oblique forest variant of the prior-art method proposed in Mesbah et al. [130]



(a) Classification Accuracy (b) Retrieval Precision

Fig. 3.9. *Evaluation of the proposed mHF vs. hashing based comparative methods varying code-size*: **Fig. 3.9a:** Overall accuracy of classification by retrieval and **Fig. 3.9b:** Precision of retrieval through hashing. We observe an increasing trend in performance of the hashing methods as the code size increases (for mHF, the classification accuracy improves from 68% with a code size of 16 to 88% for a code size of 96 bits). mHF is observed to be consistently better than the state-of-the art for code sizes beyond 32 bits followed by KSH demonstrating a similar trend. The performance of hashing methods such as KSH, AGH and PCAH saturates without significant improvement for code-sizes beyond 64 bits. Reprint from [40], with permission ©Elsevier.

- Minkowski metric - the closest competitive non-hashing technique
- LPP - the closest competitive metric learning method
- State-of-the-art unsupervised hashing algorithms (LSH, AGH, PCAH, SH) and
- Supervised hashing methods (KSH, LDAH)

Adopting the *one vs. rest* binarization strategy, the class-specific performance is assessed with F_1-score, which is a commonly used as evaluation measure in the domain of information retrieval. The F_1-score is evaluated as the harmonic mean of the precision and recall and is evaluated at the precision-recall break-even point on a precision - recall curve (PR curve). Similar to the earlier validations, we illustrate the average and the 95% confidence intervals of neuron-type specific F_1-score in Figure 3.10, for $5-$folded cross validation setting using the same data splits employed in Section 3.4.2. Additionally, we also demonstrate the performance for retrieval of 14 distinct neurons (2 from each class) from the test set along with the top 5 approximate neighbors retrieved using the proposed mHF algorithm (96 bit hash code with `numTrees` = 24 and `treeDepth` = 4.) in Figure 3.11.

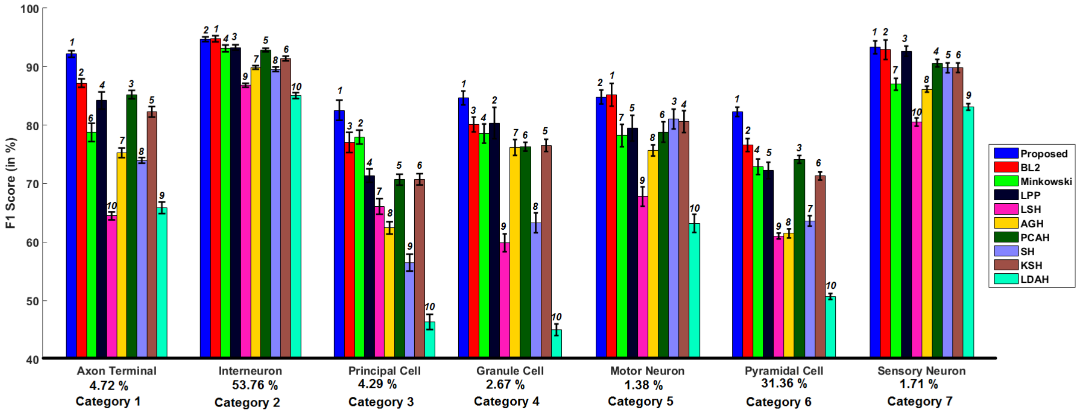


Fig. 3.10. *Class-specific evaluation of retrieval using* F_1*-score*: Contrasting class-wise retrieval performance of metric hashing forests *vs.* selected comparative methods, we osberve that mHF consistently demonstrates an F_1-score over 80% across all the neuron categories and demonstrates highest mean performance in six of the seven categories. The closest method amongst metric learning methods to mHF is using locality preserving projections (LPP) and kernel supervised hashing (KSH) is the closest amongst hashing methods. Reprint from [40], with permission ©Elsevier.

3.4.6 Analysis of time and memory costs for training and testing of the comparative methods

The time efficiency is an important performance measure of information retrieval methods and is especially important for validating the efficacy of hashing based techniques. Methods requiring exhaustive pair-wise computation of similarity measures across the database have poor scalability to massive databases as the average time complexity of these methods is $\mathcal{O}(Nd)$ where it directly depends on size of the target database (N) and the dimensionality of the feature space (d). These methods additionally incur a memory storage overhead of the size of the entire database, thus requiring a memory footprint of $8Nd$ bytes (assuming double-precision floating-point representation). Employing a hashing method, we can generate compact representations of the features, leading to significant minimization of memory storage overhead. For n independent hash functions, each generating k bits, the incurred memory storage costs is $Nnk/8$ bytes (and typically $nk/8 << 8d$). As discussed in Section 2.3.4, we consider two different strategies for hashing codeword comparison: (1) Forward Search and (2) Inverse Lookup. Despite reduced computational expense due to binary `xor` operation (between two nk binary codewords) in comparison to distance computation (between two d-dimensional floating-point feature vectors), forward search still incurs a time complexity of $\mathcal{O}(Nnk)$, which is linear to the database size and hence limits its scalability. With inverse lookup scheme, we achieve a time complexity of $\mathcal{O}(nk)$, which is independent of the database size. This argument favoring code-comparison in hashing with inverse lookup protocol extends seamlessly to mHF and to the comparative hashing techniques as well as the baselines considered in this work.

To evaluate the time and memory costs, we record the training time (time required to train the hashing functions (τ_1); generate the hashing table of the target database (τ_2)) and testing time (time required for code generation of the query items (τ_3); time required for code-comparison and sorting through forward search (τ_4^f) and inverse lookup (τ_4^i)) using the code-profiler

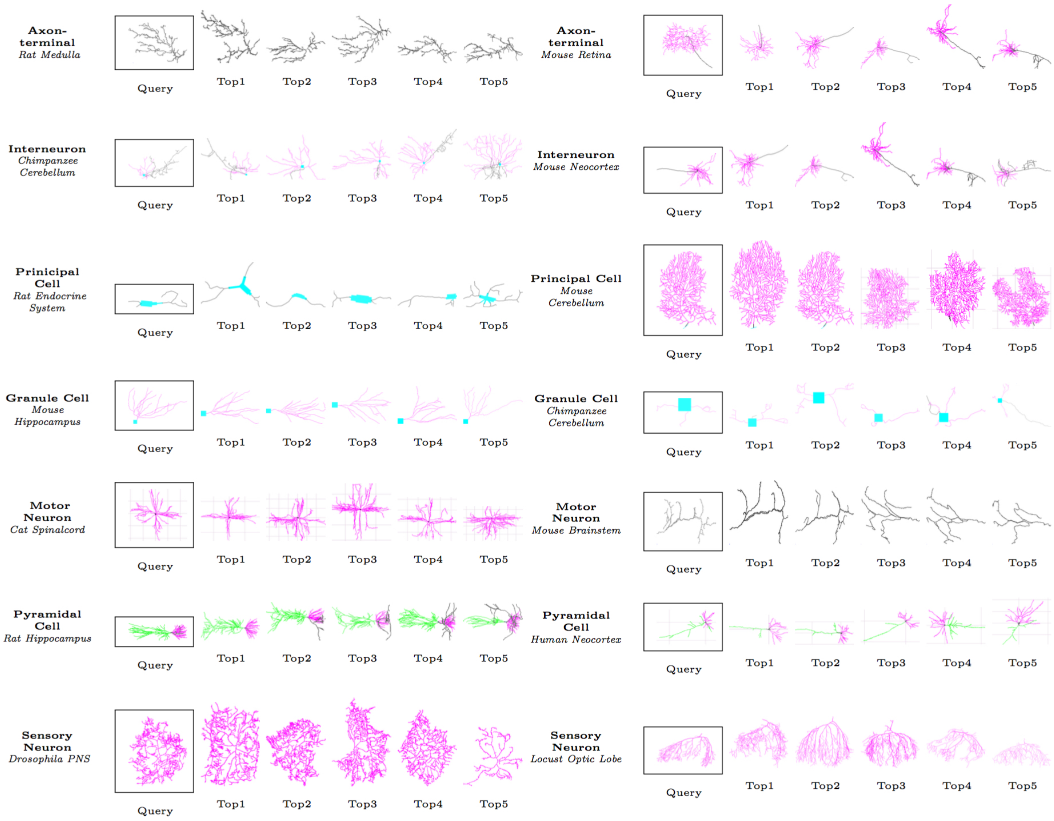


Fig. 3.11. *Illustrative examples neurons fetched upon query based neuron image retrieval using mHF*: We present the retrieval results for 14 different unseen query neurons (belonging to 7 different cell types) acquired from different species and brain regions. From a superficial visual inspection, we observe that all the retrieved neighbors consistently share morphological similarities with respect to the query especially. Reprint from [40], with permission ©Elsevier.

within MATLAB 2015b. We also report the memory cost incurred at testing time when a particular method is deployed for retrieval. This includes the memory overhead of the hashing functions and the database (for non-hashing based methods) or storing the hash table (for hashing based methods). The average time and memory costs are tabulated in Table 3.4 for all the baselines and the comparative non-hashing as well as hashing based methods (code-size of 96 bits) for a $5-$folded cross-validation setting using data-splits identical to the ones used in the earlier experiments.

3.5 Discussion

3.5.1 Validations against baselines

In this validation, we compare the performance of mHF against incrementally designed baselines to critically analyze the major contributing elements namely, inclusion of supervisory information, local metric learning and the importance of neighborhood quality function in choosing optimal splits. By looking at Fig. 3.7, we observe that all hashing forests demonstrate an overall monotonically improving trend as the code-size increases. This is consistent with

Category	Comparative Methods	Training Time (in s)			Testing Time (in s)					Query Time per sample		Memory
		τ_1	τ_2	$\tau_1 + \tau_2$	τ_3	Forward τ_4^f	Inverse τ_4^i	Forward $\tau_3 + \tau_4^f$	Inverse $\tau_3 + \tau_4^i$	Forward τ_q^f **(in ms)**	Inverse τ_q^i**(in ms)**	(in MB)
Exhaustive Search	Euclidean	-	-	-	-	53.27	-	53.27	-	11.95	-	5.441
	Chebyshev					52.04		52.04		11.69		
	Minkowski					47.51		47.51		10.67		
Metric Learning	Mahalanobis	0.002		0.002		52.27		52.27		11.74		
	PCA	0.129		0.129		46.88		46.88		10.53		
	uNPE	3.914		3.914		50.11		50.11		11.25		
	sNPE	0.852		0.852		50.49		50.49		11.34		
	LSDA	0.641		0.641		49.83		49.83		11.19		
	LPP	0.210		0.210		46.93		46.93		10.54		
Hashing based comparative methods	AGH	0.573	0.112	0.685	0.029	1.348	0.342	1.378	0.372	0.319	0.084	0.279
	PCAH	0.195	0.011	0.206	0.008	1.550	0.332	1.558	0.340	0.350	0.076	0.249
	SH	0.194	0.094	0.288	0.083	1.618	0.317	1.701	0.400	0.382	0.089	0.250
	LSH	0.020	0.006	0.026	0.006	1.724	0.372	1.730	0.378	0.388	0.084	0.238
	KSH	943.4	0.018	943.4	0.018	1.538	0.342	1.556	0.360	0.349	0.081	0.314
	LDAH	0.010	0.007	0.018	0.006	1.502	0.348	1.508	0.354	0.339	0.079	0.217
Baselines	BL1	1.375	0.147	1.522	0.061	1.254	0.321	1.315	0.382	0.295	0.086	0.316
	BL2	1.405	0.170	1.575	0.059	1.267	0.312	1.324	0.369	0.297	0.083	**0.325**
	Bl3	24.86	0.352	25.23	0.086	1.308	0.331	1.393	0.417	0.313	0.094	
	BL4	9.510	0.277	9.786	0.089	1.333	0.342	1.421	0.431	0.319	0.097	
	BL5	27.06	0.289	27.35	0.089	1.337	0.332	1.425	0.421	0.320	0.095	
	BL6	11.01	0.382	11.39	0.086	1.333	0.311	1.420	0.400	0.319	0.089	
Proposed		**27.98**	**0.335**	**28.32**	**0.082**	**1.349**	**0.346**	**1.430**	**0.427**	**0.321**	**0.096**	

Notation: τ_1 - Hashing function training time; τ_2 - Hash table generation time; τ_3 - Query hash code generation time; τ_4^f - Comparison time with forward search; τ_4^i - Comparison time with inverse lookup; τ_q^f - Query time *per* sample with forward search and τ_q^i - Query time *per* sample with inverse lookup.
Note: Comparative analysis is presented for a scenario of $k = 5$ cross-validation (*i.e.* 80% of the data (17,812 neurons) is employed in training and the rest 20% (4,453 neurons) for testing.)

Tab. 3.4. *Analysis of Training and Retrieval Time and Memory Requirement*: We observe that the training time for mHF (28 seconds) is highly competitive and significantly faster than KSH (943 seconds) which is the closest state-of-the art supervised hashing method in terms of retrieval performance. We consistently observe that the testing time of hashing methods employing inverse-lookup (0.42 seconds for mHF) is significantly lower than methods employing exhaustive pairwise distance computations such as in metric learning 47 seconds for LPP). In terms of memory requirements, the hashing models incur a significantly low memory cost in contrast to metric learning methods. Reprint from [40], with permission ©Elsevier.

the expected behavior of an ensemble learning method as increasing strength of the ensemble leads to improved generalization performance until convergence. Additionally, the proposed mHF demonstrates a statistically significant improvement over the baselines for code-sizes in the range of 8-64 bits (at 56 bits, we observe over 8% (for CA) and over 7% (for RP) improvement of mHF in comparison to the closest baselines BL5 and BL6, respectively). Beyond 72 bits, the mHF is consistently better, however, the margin on improvement is reduced to 4% (for both CA and RP) at the maximal 96 bits. Beyond 96 bits, the feature space is heavily parsed and the methods demonstrate asympotic convergence to the upper-bound of the generalization performance. For time-efficient hashing and reduced storage complexity, it is highly desirable to maintain smaller code-sizes. As mHF discovers and encodes the true class neighborhood better than the other baselines, we infer that it is well suited for the task of similarity-preserving retrieval and subsequent classification. Next, we perform a comprehensive evaluation and contrast each of the baselines against mHF and present our observations and inferences as follows:

Effect of choice of split function ϕ

The BL1 is our prior-art unsupervised hashing forests technique [130], where it performed superior Euclidean neighborhood approximation at the cost of longer code-sizes (over 512 bits - not shown in Fig. 3.7) for comparable retrieval performance (CA: 81.4% $\pm 2.1\%$), which is achievable through proposed mHF at code-size of 56 bits (CA: 83.5% $\pm 2.2\%$). As the architecture of the tree is directly related to the code-size, univariate splits often require deep nested trees to separate distributions that are obliquely aligned between the feature axes [129]. Given a fixed `treeDepth`, the parsing induced by BL1 is significantly weaker than that induced by oblique decision boundaries employed in mHF (and other baselines). The baseline BL2 alleviates the aforementioned limitation of univariate splits in BL1 by introducing oblique splits at each split node. In comparison to BL1, BL2 demonstrates a very significant improvement (beyond code-size of 32 bits) thus reaffirming our earlier inference. Contrasting against mHF, for code-sizes up to 88 bits, the large difference in performance is probably attributed to the lack of inclusion of task-specific information in training of hashing forests.

Effect of supervision

The baseline BL3 is a supervised variant of BL2, where supervisory information is incorporated by optimizing the node-level class separability with the neighborhood quality function (Eq. 3.6, 3.11). In comparison to BL2, the margin of improvement is statistically significant till code-size of 80 bits. In comparison to mHF, the performance margin of BL3 is comparatively lower than BL2, validating our hypothesis of optimizing class separability as the parsed subspaces are more class-consistent in comparison to random parsing used by BL2.

Effect of local metric learning

The baselines BL4 - 6 have been defined to validate the proposal of local metric learning for defining splits on a learned latent subspace. The BL4 utilizes PCA transformation (unsupervised) at each node level to project the data along the directions of maximal variance. Contrasting against BL2 (equivalent to Eye transformation at each node level), the improvement margin is statistically significant at smaller code-sizes (up to 32 bits). Beyond 72 bits, BL3 and BL4 show convergent trends with no significant margin and incorporating PCA at each node level is deemed not necessary as oblique splits can effectively induce the same amount of subspace parsing. Comparing with mHF, a large performance margin is seen across all code-sizes, which can be associated to the same limitations inherited within purely unsupervised methods as discussed earlier for BL2.

Following on similar lines of BL3, we modify BL4 by incorporating the neighborhood quality function to obtain BL5. As evident in Fig. 3.7, the BL5 is the closest to mHF when contrasting the performance. The BL5 also demonstrates a statistically significant improvement margin over BL4, thus strongly supporting the efficacy of the proposed node optimization strategy for reasons similar to the ones discussed in BL3. We define the baseline BL6 to evaluate the hypothesis that local supervised metric learning helps discover subspaces with better class separability, which are highly desirable for similarity preserving retrieval. By contrasting against BL4, we observe that incorporating supervisory information, significantly improves the retrieval performance across all code-sizes. In comparison to BL5 and mHF, the performance of BL6 is marginally lower than BL5 (and similarly BL5 is lower than mHF) suggesting that mHF,which uses local metric learning augmented with splits preserving class separability, results in superior performance than the baselines.

3.5.2 Comparative analysis against non-hashing methods

Within this validation, we test our hypothesis that mHF generates hash codes that are both compact and discriminative by comparing against non-hashing based methods (dimensionality reduction) that can be alternatively used for retrieval. By looking at Figure 3.8, we observe that the mHF (CA: 87.8%$\pm 0.5\%$ at 96 bits) performs better than the closest comparative non-hashing method LPP, involving exhaustive search with supervised distance metric learning (CA: 85.4%$\pm 1.1\%$). This validates our hypothesis that retrieval in the hamming space defined by mHF preserves the desired class similarity among the approximate nearest neighbors. Amongst the l_p- norm based methods, Minkowski metric (CA: 82.2%$\pm 0.7\%$) is observed to perform the best. Comparing l_p- norm based methods to unsupervised metric learning methods (Mahalanobis, PCA and uNPE), we observe improved retrieval performance probably due to the incorporation of distribution geometry (in Mahalanobis and PCA) and local neighborhood information (in uNPE) in generating the distance metric used for defining the pairwise distances for comparison. We further included supervisory similarity information in sNPE, LSDA and LPP to learn the distance metric. The LPP improved the retrieval performance by a margin of 1.2% over Mahalanobis metric, suggesting incorporating available task relevant supervisory information is desirable for achieving better performance during retrieval.

3.5.3 Comparative analysis against hashing methods

Under Section 2.2 and Section 3.2, we highlighted the major theoretical advantages of using mHF over other hashing based approaches. Towards, this end we discuss the detailed observations and inferences of the experiment on comparing and contrasting against other hashing methods re-highlighting these advantages within the context of hashing performance on the target application. The observations are organized under three categories of code-sizes as follows:

Short Code-Sizes $16 - 32$ bits

For shorter code-sizes, we observe that the mHF's CA and RP performances are comparable to both PCAH and AGH. The performance margins amongst these methods are not statistically significant (except at 24 bits, where RP of PCAH $>$ mHF). Comparing the supervised KSH to the other methods, the performance is significantly lower. The KSH uses the separable nature of code inner products (and its equivalence to hamming distance) to greedily learn hashing functions with a sequential update scheme to minimize the difference between the ground-truth similarity matrix and the one generated using hash codes [120]. The lower performance of KSH is probably due to high residual difference between the core-inner product similarity and the ground-truth similarity matrix $\mathcal{S}$ at small code sizes.

Intermediate Code-Sizes $32 - 72$ bits

Beyond 32 bits, we observe a clear statistically significant margin of improvement of the proposed mHF over the comparative methods. The KSH exhibits the second best performance, implying that sufficiently large code-sizes along with incorporated supervisory information drives it towards more meaningful parsing and encoding of the feature space. For KSH, error between the similarity matrix inferred through hashing and the ground-truth is convergent as the code-size increases beyond 48 bits and the margin of improvement with additional bits in the codeword is not statistically significant. For LDAH, the RP is significantly improved from code-sizes of 40 to 64 bits, however, the CA metric gets saturated. This is possibly due to the shrinking of the approximate nearest neighborhood size as the code-size increases while the true class neighbors contribute little within that neighborhood. The comparatively lower performance of LDAH is mainly because of the over simplistic nature of it underlying linear separability assumption amongst different classes, which is not extendable to the highly heterogeneous neuromorphological feature space.

Longer Code-Sizes $80 - 96$ bits

For longer code-sizes, the performance of mHF, KSH, PCAH, and LSH are observed to converge asymptotically around 96 bits. The mHF converges to a significantly higher value over the closest hashing method (KSH) by a margin of over 6.1% and 5.1% in CA and RP, respectively. This trend of convergence is expected as the nearest approximate neighborhood defined by hashing tends to converge to the desired true class neighborhood for supervised methods (or geometric true neighborhood for the unsupervised methods) as code-size increases.

3.5.4 Evaluation of class-wise retrieval performance

Firstly, through visual evaluation of Figure 3.11, we observe close 3D morphological and class similarity amongst the neurons and their retrieved neighbors, invariant to their orientation, which implies that the mHF fetches true class-specific neighbors to a query neuron during retrieval. Secondly, in Figure 3.10, we observe that the proposed mHF is consistently ranked top by a statistically significant margin in Categories 1,3,4 and 6 and very competitive to the top performing methods for 2,5 and 7 categories. It is worth mentioning that the target classification by retrieval task has an imbalanced training dataset and the mHF is able to handle it in a satisfactory fashion. The hierarchical nature of the parsing function allows for exploration of the localized constrained feature subspaces, where linear metric learning is more meaningful in inducing class separability. Comparing mHF and baseline BL2 [130] to non-hashing exhaustive search (Minkowski and LPP), we observe that in a majority of categories, the LPP fares better than the retrieval in the original feature space (Categories 1,2,3,4 and 7). The inclusion of supervisory information in defining the LPP transformation renders the neighborhoods on the resultant metric space to be more class consistent in comparison to the original feature space. Comparing against the state-of-the-art hashing based methods, we observe that KSH exhibits the most consistent performance reaffirming the need for supervision in the learning of the hash functions. The comparatively lower performance of SH and LDAH indicate that the underlying assumptions of SH (uniform distribution of eigenfunctions) and LDAH (linear separability amongst classes) are not justified for a challenging application like neuroscientific image retrieval.

3.5.5 Analysis of time and memory costs for training and testing of the comparative methods

From Table 3.4, we quantify the time and the memory requirements of mHF in comparison to the comparative methods and report our inferences as follows[2]:

Training time

Due to high computational costs of node optimization and learning of optimal LPP at each split node, the mHF incurs a considerably high training time in comparison to global metric learning based and hashing based methods (except KSH). The KSH, which is the closest to mHF in terms of retrieval performance has a very high training time (over $30 \times \tau_1$ reported for mHF), because of underlying sequential optimization. It must be noted that the training time for mHF can be considerably lowered if parallelized training of the independent mHTs is used.

Testing time

Retrieval time is an important indicator of hashing efficiency during deployment. Hashing tries to circumvent exhaustive pairwise comparisions across the database using hash table generation and inverse lookup (discussed in Section 2.3.4). From Table 3.4, for forward search, we observe that the query time per sample is reduced for mHF by over 32 times in contrast to forward PCA Metric Learning based retrieval (fastest non-hashing based comparative method). This is owing to the reduced computational expense of binary `xor` operation used in forward search of hashing in comparison to distance calculation in non-hashing methods. Further, employing inverse-lookup, the retrieval time is significantly decreased(by over 3.3 times for mHF). This improved trend is consistently observed among inverse-lookup in all the hashing-based methods.

Memory

For scalability to large databases, the retrieval method must have minimalistic memory requirements. In comparison to non-hashing methods, which demand for an access to the full feature matrix, access to the hash table suffices for retrieval using hashing based methods. From Table 3.4, we observe a 16 times compression in memory requirement for mHF (96 bits) in contrast to non-hashing methods. In comparison to other hashing-based methods, the memory requirement for mHF is fractionally higher.

[2] It must be noted that the time reported in Table 3.4 does not include the time for generating features from 3D digital reconstructions of neurons. On average, we observed that the L-measure tool [166] required 0.1005 second per neuron for feature extraction. This is a constant overhead that is applicable to all retrieval methods.

4 Hashing with Residual Networks

> *We chose it because we deal with huge amounts of data. Besides, it sounds really cool.*
>
> — **Larry Page**
> Co-founder of Google Inc.

4.1 Overview and Publications

This chapter presents the contributions of this thesis concerning learning end-to-end hash codes using deep learning. We discuss a novel deeply learnt convolutional neural network architecture for supervised hashing of medical images through residual learning, coined as Deep Residual Hashing (DRH). First, DRH offers maximal separability of classes in hashing space while preserving semantic similarities in local embedding neighborhoods. Second, a new optimization formulation comprising of complementary loss terms and regularizations that suit hashing objectives is introduced. Extensive validations on a large-scale public Chest X-ray images with co-morbidities demonstrates improved performance and computational benefits of the proposed algorithm for fast and scalable retrieval.

The chapter is organized as follows: in Sec. 4.2 we present the underlying clinical motivation for investigating deep learning for hashing, particularly for the task of retrieval within Chest X-ray databases. The methodology behind training such a deep hashing model is presented in detail in Sec. 4.3, wherein we discuss the architecture of such a network in Sec. 4.3.1, the formulation of the supervised retrieval loss function in Sec. 4.3.2, hashing related losses and regularizations employed while model learning in Sec. 4.3.3 and finally conclude the section with aspects of model learning in Sec. 4.3.4. Following the exposition of the methodology, the contributions of this chapter are evaluated on a large-scale Chest-X ray image database with co-occurring disease manifestations and the results of which are presented in Sec. 4.4. Particularly in Sec. 4.4.2 and Sec. 4.4.3, we compare and contrast with multiple ablative baselines and incremental variants of DRH and also evaluate against state-of-the art deep learning based hashing methods. In Sec. 4.5, we present an in-depth discussion on different aspects of leveraging deep learning for hashing, focusing on the trainability of such networks in Sec. 4.5.1, the representability of the learnt hash codes in Sec. 4.5.2, on the compactness of the learnt hash codes in Sec. 4.5.3, on how well the semantics are preserved in local neighborhoods in the learnt encoding space in Sec. 4.5.4 and finally close the section on how joint optimization improved hash code quality in Sec. 4.5.5.

Substantial parts of this chapter have already been published in the following article and are quoted verbatim:

[39] **Conjeti, Sailesh**, Abhijit Guha Roy, Amin Katouzian, and Nassir Navab. "Hashing with residual networks for image retrieval." *In International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 541-549. Springer, Cham, 2017.*

Copyright Statement. ©Springer International Publishing AG 2017.

4.2 Motivation

Content-based image retrieval (CBIR) aims at effectively indexing and mining large image databases such that given an unseen query image we can effectively retrieve images that are similar in content. The search is driven in entirety by the contents of the image and does not rely explicitly on keywords describing the image. With the deluge in medical imaging data, there is a need to develop CBIR systems that are both fast and efficient. For example, a radiologist examining a chest x-ray (CXR) image which displays manifestations of a particular cardiopulmonary disease should be able to retrieve images with similar abnormalities. This could facilitate and objectify the reading of the medical image and help make better diagnosis [10]. However, in practice, it is often infeasible to exhaustively compute similarity scores between the query image and each image within the database. A radiologist can better understand the disease in an image if presented with its context *e.g.,* where the disease is, the degree of severity, organ affected *etc.* Furthermore, with large collection of medical images being curated within hospitals, images can be retrieved based on their context, example with textual queries such as " *retrieve images with cardiopulmonary diseases in the upper right lobe of the lung*" or purely image-based by *querying with an image*. A preview of retrieved images using the proposed image-based retrieval in the setting of cardiopulmonary chest X-rays is demonstrated in Fig. 4.1.

With the latter goal, we propose an end-to-end one-stage deep residual hashing (DRH) network to directly generate hash codes from input images. Specifically, the DRH model consists of a sub-network with multiple residual convolutional blocks for learning discriminative image representations followed by a fully-connected hashing layer to generate compact binary embeddings. Through extensive validation, we demonstrate that DRH learns discriminative hash codes in an end-to-end fashion and demonstrates high retrieval quality on standard chest X-ray image databases. The ultimate objective of learning similarity preserving hashing functions is to generate embeddings in a latent Hamming space such that the class-separability is preserved while embedding and local neighborhoods are well defined and semantically relevant. This can be visualized in 2D by generating the t - Stochastic Neighborhood Embedding (t-SNE) [124] of unseen test data post learning like shown in Fig. 4.2. Starting from Fig. 4.2(a) which is generated by a purely unsupervised setting we aim at moving towards Fig. 4.2(d) which is closer to an ideal embedding. In fact, Fig. 4.2 represents the results of our proposed DRH approach in comparison to other methods and baselines.

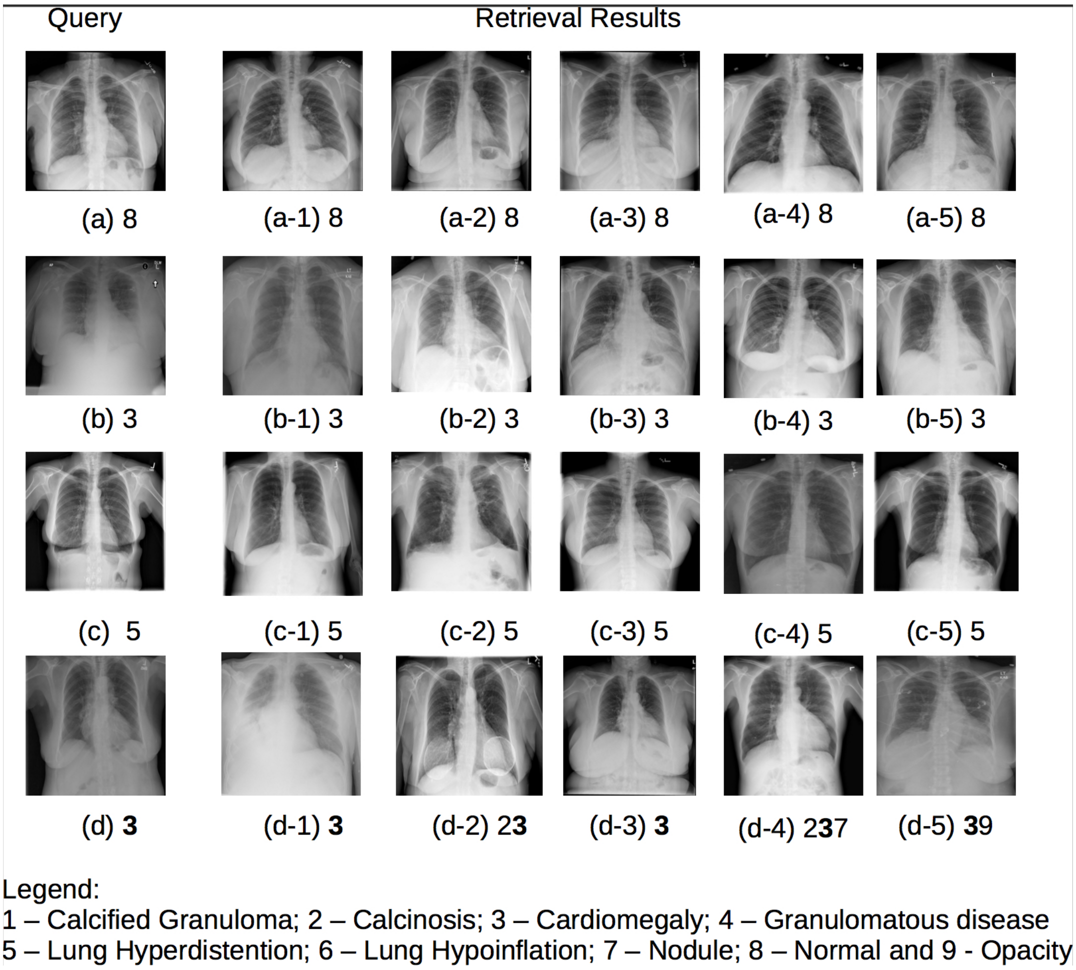


Fig. 4.1. *Preview of retrieved results for unseen query images* using Deep Residual Hashing network with 64-bit encoding and 34-layer depth. Particularly note that the retrieval preserves class-similarity and is insensitive to intensity differences due to windowing and varying anatomies. Reprint from [39], with permission of Springer.

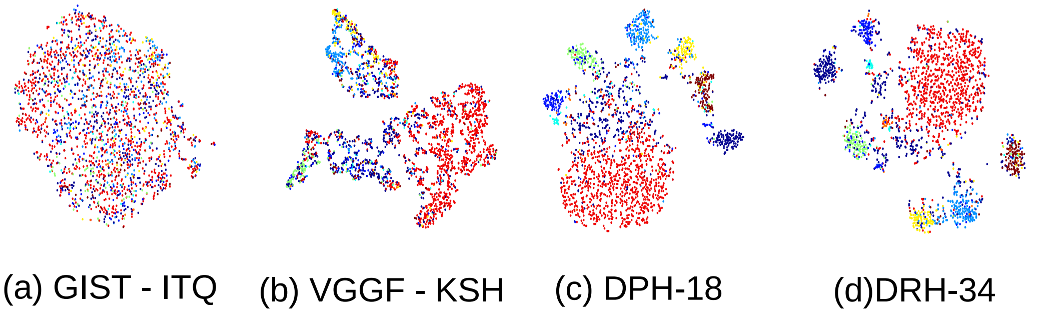


Fig. 4.2. *tSNE embeddings of the hash codes generated by the proposed and comparative methods*: GIST - ITQ uses hand-crafted features followed by iterative quantization for hashing. VGGF - KSH is a two-stage hashing method that fine-tunes VGGF network for CXR images and subsequently hashes them with KSH. Deep PlainNet Hashing (DPH-18) is a 18-layer convolutional neural network trained end-to-end for hashing. Deep Residual Hashing (DRH- 34) is the proposed 34-layer residual network for simultaneous representation learning and hashing. Color indicates different classes. The figure needs to be viewed in color. Reprint from [39], with permission of Springer.

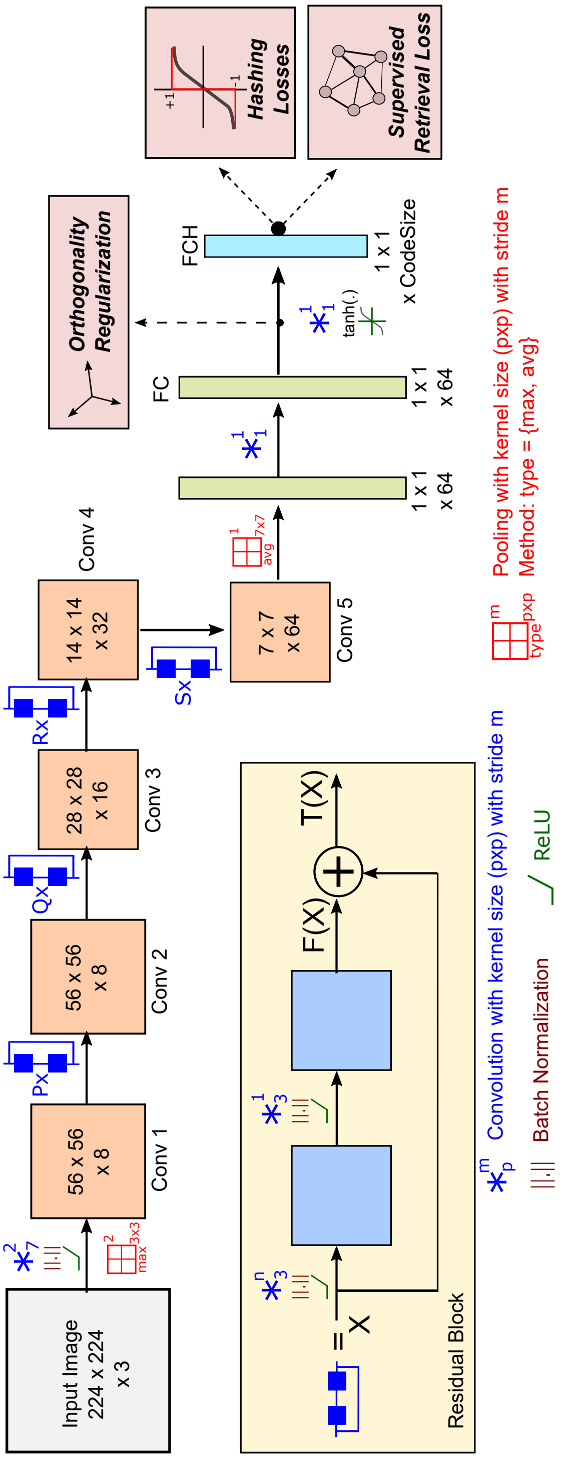


Fig. 4.3. *Network architecture for deep residual hashing (DRH) with a hash layer*: For a 18 - layer network, the number stacked residual blocks are: P = 2, Q = 2, R = 2 and S = 2. Likewise, for a 34 - layer network, P = 3, Q = 4, R = 6 and S = 3. The inset image on the left corner is a schematic illustration of a residual block. Reprint from [39], with permission of Springer.

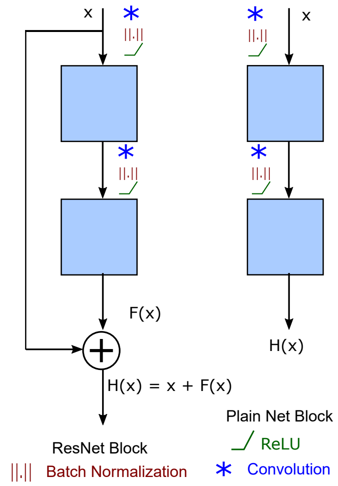


Fig. 4.4. *Residual vs. Plain Convolutional Block*: Residual connections are well suited for training very deep networks without loss of representational power and helps mitigate vanishing gradients due to skip connections. With increasing depth, CNNs with PlainNet blocks are prone to limited trainability as the gradients can potentially vanish during back-propagation.

4.3 Methods

Unlike classification which aims at assigning a semantic class to a test image, retrieval aims at fetching examples from the training set that share semantic similarities with the query image. Features representations tailored for the retrieval task map input images into a latent feature space such that dissimilar instances are mapped apart while simultaneously mapping similar instances closer into a compact local neighborhood. Typically, Euclidean distance in such a space between the instances mimics their semantic similarity.

An ideal hashing method should generate codes that are compact, similarity preserving and easy to compute representations (typically, binary in nature), which can be leveraged for accurate search and fast retrieval [180]. The desired similarity preserving aspect of the hashing function implies that *semantically similar images are encoded with similar hash codes*. Mathematically, hashing aims at learning a mapping $\mathcal{H} : \mathcal{I} \rightarrow \{-1, 1\}^K$, such that an input image $\mathcal{I}$ can be encoded into a K bit binary code $\mathcal{H}(\mathcal{I})$. In hashing for image retrieval, we typically define a similarity matrix $\mathcal{S} = \{s_{ij}\}$, where $s_{ij} = 1$ implies images $\mathcal{I}_i$ and $\mathcal{I}_j$ are similar and $s_{ij} = 0$ indicates they are dissimilar. Similarity preserving hashing aims at learning an encoding function $\mathcal{H}$ such that the similarity matrix $\mathcal{S}$ is maximally-preserved in the binary hamming space.

4.3.1 Architecture for deep residual hashing

We start with a deep convolutional neural network architecture inspired in part by the seminal ResNet architecture proposed for image classification by He *et al.* [73]. As shown in Fig. 4.3, the proposed architecture consists of the a convolutional layer (Conv 1) followed by a sequence of residual blocks (Conv 2-5) and terminating in a final fully connected hashing (FCH) layer for hash code-generation.

Residual connections have been demonstrated to facilitate training very deep networks without overfitting, which is highly desirable for an end-to-end learning scenario like the current one. The key idea behind residual learning arises from the empirically tested hypothesis that while adding a stack of learning layers to deep network it is often easier to optimise residual mapping than to optimise the original, unreferenced mapping [73]. Multiple non-linear layers in network can asymptotically approximate complicated functions, say mapping to $T(\mathbf{x})$ from $\mathbf{x}$. Residual learning drives such blocks to explicitly approximate a residual function $F(\mathbf{x}) = T(\mathbf{x}) - \mathbf{x}$ and then adds $\mathbf{x}$ to the residual $F(\mathbf{x})$ to approximate $T(\mathbf{x})$. The last operation is performed via a shortcut connection from the input and by performing element-wise addition. This is schematically illustrated in Fig. 4.4. Another major issue to training deep architectures is the problem of vanishing gradients during training (this is in part mitigated with the introduction of rectified linear units (ReLU), input batch normalization [87] and layer normalization). Residual connections offer additional support *via* a no-resistance path for the flow of gradients along the shortcut connections to reach the shallow learning layers. These unique advantages offered by the ResNet architecture motivates use to leverage it for the application at hand.

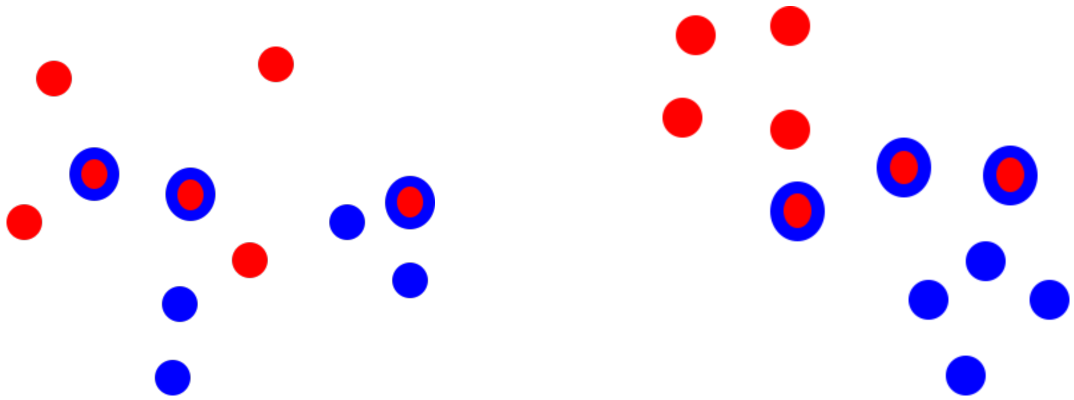
Let $\mathbf{h}_i$ represent the output of the final FCH layer (shown in Fig. 4.3) for an input image $\mathcal{I}_i$ generated by passing through a DRH network. We perform quantization of this output to obtain the binary codes as: $\mathbf{b}_i = \operatorname{sgn}(\mathbf{h}_i)$. Let $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_N] \in \{-1, 1\}^{K \times N}$ and $\mathbf{H} = [\mathbf{h}_1, \cdots, \mathbf{h}_N] \in \mathbb{R}^{K \times N}$ be the matrix representation of the K bit hash codes and the corresponding non-quantized outputs of the DRH network for a batch of N input images.

4.3.2 Supervised Retrieval Loss Function

Typically, end-to-end learnt deep networks are trained with respect of classification by minimising loss functions like L_2-norm, cross-entropy loss *etc.* [103] [26]. This does not seamlessly extend to the task of retrieval and might generate sub-optimal representations. In order to learn feature embeddings tailored for retrieval and specifically for the scenario at hand where the pairwise similarity matrix $\mathcal{S}$ should be preserved, we propose our supervised retrieval loss drawing inspiration from the neighbourhood component analysis [65]. To encourage the learnt embedding to be binary in nature, we squash the output of the residual layers to be within $[-1, 1]$ by passing it through a hyperbolic tangent (tanh) activation function. The final binary hash codes $(\mathbf{b}_i)$ are generated by quantizing the output of the tanh activation function (say, $\mathbf{h}_i$) as follows: $\mathbf{b}_i = \operatorname{sgn}(\mathbf{h}_i)$. Given N instances and the corresponding similarity matrix is defined as $\mathcal{S} = \{s_{ij}\}_{i,j=1}^{N} \in \{0, 1\}^{N \times N}$, the proposed supervised retrieval loss is formulated as:

$$J_S = 1 - \frac{1}{N} \sum_{i,j=1}^{N} p_{ij} s_{ij} \tag{4.1}$$

where p_{ij} is the probability that any two instances (i and j) can be potential neighbours. Inspired by kNN classification, where the decision of an unseen test sample is determined by



(a) Before Training (b) After Training

Fig. 4.5. *2D embedding of images in Hamming Space*: Schematic illustrating in 2D the embedding of samples in the Hamming space, within the batch. Colors indicate present disease manifestations, with multiple colors indicating co-occurring diseases. In Fig. 4.5a, prior to training, the data items are organized in random fashion within the encoding Hamming space only regularized by global image similarities. Post training, as shown in Fig. 4.5b, the embeddings ideally order such that local neighborhoods preserve the semantic similarities. Under such an encoding, the Hamming distance defined between hash codes approximates the true neighbor distances as defined by the training semantic labels. Particularly, data items with multiple classes would embed in features spaces *in between* the clusters of data items with respective individual labels.

the semantic context of its local neighbourhood in the embedding space, we define p_{ij} as a softmax function of the hamming distance (indicated as $\oplus$) between the hash codes of two instances and is derived as:

$$p_{ij} = \frac{e^{-(\mathbf{b}_i \oplus \mathbf{b}_j)}}{\sum_{l \neq i} e^{-(\mathbf{b}_i \oplus \mathbf{b}_l)}} \text{ where } \mathbf{b}_{(\cdot)} = \operatorname{sgn}\left(\mathbf{h}_{(\cdot)}\right) \tag{4.2}$$

As gradient based optimisation of J_s in a binary embedding space is infeasible due to its non-differentiable nature, we use a continuous domain relaxation and substitute non-quantized embeddings $\mathbf{h}_{(\cdot)}$ in place of hash code $\mathbf{b}_{(\cdot)}$ and Euclidean distance as as surrogate of Hamming distance between binary codes. This is derived as:

$$p_{ij} = \frac{{}^{-\|\mathbf{h}_i - \mathbf{h}_j\|^2}}{\sum_{i \neq l} e^{-\|\mathbf{h}_i - \mathbf{h}_l\|^2}} \tag{4.3}$$

It must be noted that such an continuous relaxation could potentially result in uncontrollable quantization error and large approximation errors in distance estimation. With continuous relaxation, Eq. (4.1) is now differentiable and continuous thus suited for backpropagation of gradients during training. In Fig. 4.5a and Fig. 4.5b, we illustrate the embedding of the data points before and after training with the supervised cost function.

4.3.3 Hashing related Loss Functions and Regularization

Generation of high quality hash codes requires us to control this quantization error and bridge the gap between the Hamming distance and its continuous surrogate. We jointly optimise for J_s and improve hash code generation by imposing additional loss functions as follows:

Quantization Loss

In the seminal work on iterative quantization (ITQ) for hashing [66], Gong and Lazebnik introduced the notion of quantization error $J_{Q-\mathrm{ITQ}}$ as

$$J_{Q-\mathrm{ITQ}} = \|\mathbf{h}_i - \mathrm{sgn}\left(\mathbf{h}_i\right)\|_2 \tag{4.4}$$

Optimizing for $J_{Q-\mathrm{ITQ}}$ required a computation intensive alternating optimization procedure and is not compatible with back propagation which is used to train deep neural nets (due to non-differentiable signum (sgn) function within the formulation). Towards this end, we use a modified point-wise quantization loss function proposed by Zhu *et al.* sans the sgn function as [224]:

$$J_{Q-\mathrm{Zhu}} = \||\mathbf{h}_i| - \mathbf{1}\|_1 \tag{4.5}$$

They establish that $J_{Q-\mathrm{Zhu}}$ is an upper bound over $J_{Q-\mathrm{ITQ}}$, therefore can be deemed as a reasonable loss function to control quantization error. For ease of back-propagation, we propose to use a differentiable smooth surrogate to L_1 norm:

$$|(\cdot)|_1 \approx \log\, \cosh\left(\cdot\right) \tag{4.6}$$

and derived the proposed quantization loss function as

$$J_Q = \sum_{i=1}^{N} \left(\log\, \cosh\left(|\mathbf{h}_i| - \mathbf{1}\right)\right) \tag{4.7}$$

With the incorporation of the quantization loss, we hypothesise that the final binarization step would incur significantly less quantization error and the loss of retrieval quality (also empirically validated in Section 4.4).

Bit Balance Loss

In addition to J_Q, we introduce an additional bit balance loss J_B to maximise the entropy of the learnt hash codes and in effect create balanced hash codes. Here, J_B is derived as:

$$J_B = -\frac{1}{2N}\mathrm{tr}\left(\mathbf{H}\mathbf{H}^T\right) \tag{4.8}$$

This loss aims at encouraging maximal information storage within each hash bit.

Orthogonality Regularisation

Inspired by ITQ [66], we also introduce a relaxed orthogonality regularisation constraint R_O on the convolutional weights (say, $\mathbf{W}_h$) connecting the output of the final residual block of

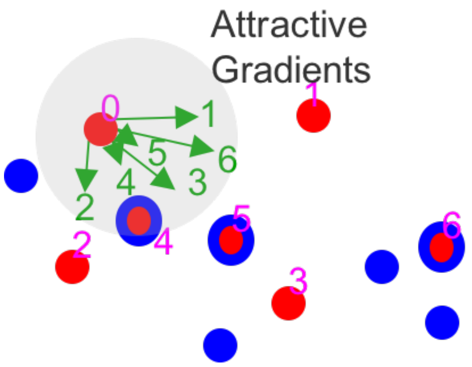


Fig. 4.6. *Attractive Gradients*: These gradients *pull* similar class samples to one-another. Unlike scenarios of training with pairwise similarities, here for each item in a training batch, every other item within the batch that shares similar class contributes with an associated attractive gradient.

the network to the hashing block. This weakly enforces that the generated codes are not correlated and each of the hash bits are independent. Here, R_O is formulated as:

$$R_O = \frac{1}{2} \left\| \mathbf{W}_h \mathbf{W}_h^T - \mathbf{I} \right\|_F^2 \tag{4.9}$$

Weight Decay Regularisation

The weight decay regularization term (R_W) is imposed to control the scale of the learnt weights and biases ($\mathbf{W}^{(\cdot)}$ and $b^{(\cdot)}$) and also helps prevents over-fitting and numerical instability in scenarios of learning with limited training examples. Here, R_W is derived as:

$$R_W = \frac{1}{2} \left(\left\| \mathbf{W}^{(\cdot)} \right\|_F^2 + \left\| b^{(\cdot)} \right\|_2^2 \right) \tag{4.10}$$

4.3.4 Model Learning

In this section, we detail on the training procedure for the proposed DRH network with respect to the supervised retrieval and hashing related loss functions. We learn a single-stage end-to-end deep network to generate hash codes directly given an input image. We formulate the optimization problem to learn the parameters of our network (say, $\Theta : \left\{ \mathbf{W}^{(\cdot)}, b^{(\cdot)} \right\}$):

$$\underset{\Theta:\left\{W^{(\cdot)},b^{(\cdot)}\right\}}{\text{argmin}} \quad J = J_S + \underbrace{\lambda_q J_Q + \lambda_b J_B}_{\text{Hashing Losses}} + \underbrace{\lambda_o R_O + \lambda_w R_W}_{\text{Regularisation}} \tag{4.11}$$

where λ_q, λ_b, λ_o and λ_w are four parameters to balance the effect of different contributing terms for the quantization, bit-balance, orthogonality regularization and weight decay respectively.

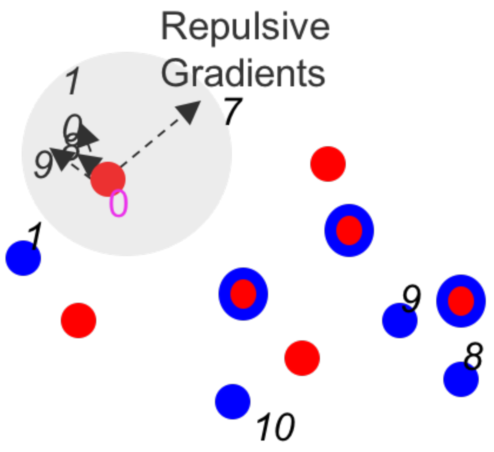


Fig. 4.7. *Repulsive Gradients*: These gradients *push* dissimilar class samples to one-another. Unlike scenarios of training with pairwise similarities, here for each item in a training batch, every other item within the batch that has a dissimilar class contributes with an associated repulsive gradient.

To solve this optimisation problem, we employ stochastic gradient descent to learn optimal network parameters. Differentiating J with respect to Θ and using chain rule, we derive:

$$\frac{\partial J}{\partial \Theta} = \frac{\partial J}{\partial \mathbf{H}} \frac{\partial \mathbf{H}}{\partial \Theta} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial J}{\partial \mathbf{h}_i} \frac{\partial \mathbf{h}_i}{\partial \Theta} \tag{4.12}$$

The second term $\partial \mathbf{h}_i / \partial \Theta$ is computed through gradient back-propagation. The first term $(\partial J / \partial \mathbf{h}_i)$ is the gradient of the composite loss function J with respect to the output hash codes of the DRH network.

We differentiate the continuous relaxation of the supervised retrieval loss function with respect to the hash code of a single example $(\mathbf{h}_i)$ as follows [65]:

$$\begin{aligned} \frac{\partial J_S}{\partial \mathbf{h}_i} = 2 &\left(\sum_{l: s_{li} > 0} p_{li} d_{li} - \sum_{l \neq i} \left(\sum_{q: s_{lq} > 0} p_{lq} \right) p_{li} d_{li} \right) \\ -2 &\left(\sum_{j: s_{ij} > 0} p_{ij} d_{ij} - \sum_{j: s_{ij} > 0} p_{ij} \left(\sum_{z \neq i} p_{iz} d_{iz} \right) \right) \end{aligned} \tag{4.13}$$

where $d_{ij} = \mathbf{h}_i - \mathbf{h}_j$. As shown in Fig. 4.6 and Fig. 4.7, the gradients at each data point (here Point 0) are computed with respect to every other point in the training batch. They are attractive in nature from points $(l : s_{li} > 0)$ and repulsive from points $(l : s_{li} = 0)$. Both the aforementioned gradients are added through gradient tensor addition (Fig. 4.8) and the network parameters are updated using gradients across the whole batch (Fig. 4.9).

The derivatives of hashing related loss functions (J_Q and J_B) are derived as:

$$\frac{\partial J_Q}{\partial \mathbf{h}_i} = \tanh\left(|\mathbf{h}_i| - \mathbf{1}\right) \operatorname{sgn}\left(\mathbf{h}_i\right) \tag{4.14}$$

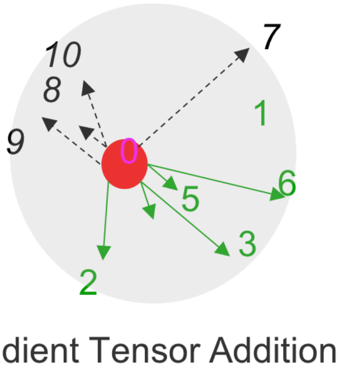


Fig. 4.8. *Gradient Tensor Addition*: The final gradient update at each sample within the batch is evaluated through a tensor addition of attractive and repulsive gradients.

and

$$\frac{\partial J_B}{\partial \mathbf{h}_i} = -\mathbf{h}_i \tag{4.15}$$

The regularisation function R_O acts on the convolutional weights corresponding to the hash layer ($\mathbf{W}_h$) and its derivative with respect to $\mathbf{W}_h$ is derived as follows:

$$\frac{\partial R_O}{\partial \mathbf{W}_h} = \mathbf{W}_h \left(\mathbf{W}_h \mathbf{W}_h^T - \mathbf{I} \right) \tag{4.16}$$

Similarly, the derivative of weight-decay regularisation R_w is derived as:

$$\frac{\partial R_w}{\partial \mathbf{W}^{(\cdot)}} = \mathbf{W}^{(\cdot)} \text{ and } \frac{\partial R_w}{\partial c^{(\cdot)}} = c^{(\cdot)} \tag{4.17}$$

Having computed the gradients of the individual components of the loss function with respect to the parameters of DRH, we apply gradient-based learning rule to update Θ. We use mini-batch stochastic gradient descent (SGD) with momentum. SGD incurs limited memory requirements and reduces the variance of parameter updates. The addition of the momentum term γ leads to stable convergence. The update rule for the weights of the hash layer is derived as:

$$\mathbf{W}_h^t = \mathbf{W}_h^{t-1} - \nu^t \text{ where } \nu^t = \gamma \nu^{t-1} + \eta \left(\frac{\partial J}{\partial \mathbf{W}_h^{t-1}} + \lambda_o \frac{\partial R_O}{\partial \mathbf{W}_h^{t-1}} + \lambda_w \frac{\partial R_W}{\partial \mathbf{W}_h^{t-1}} \right) \tag{4.18}$$

The convolutional weights and biases of the other layers are updated similarly. It must be noted that the learning rate η in Eq 4.18 is an important hyper-parameter. For faster learning, we initialize it the largest learning rate that stably decreases the objective function (typically, at 10^{-2} or 10^{-3}). Upon convergence at a particular setting of η, we scale the learning rate multiplicatively by a factor of 0.1 and resume training. This is repeated until convergence or reaching the maximum number of epochs.

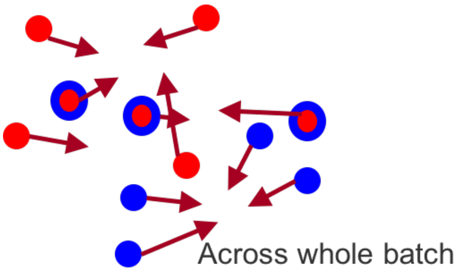


Fig. 4.9. *Gradients across a batch*: Schematic illustrating the direction and magnitude of gradients on all samples within a training batch. The similar items are incrementally pulled towards one-another, while the dissimilar items are pushed further apart.

4.4 Experiments and Results

4.4.1 Chest XRay Image Retrieval

We conducted empirical evaluations on the publicly available Indiana University Chest X-rays (CXR) dataset archived from their hospital's picture archival systems [48]. The fully-anonymized dataset is publicly available through the OpenI image collection system [229]. For this chapter, we use a subset of 2,599 frontal view CXR images that have matched radiology reports available for different patients. Following the label generation strategy published in [174] for this dataset, we extracted nine most frequently occurring unique patterns of Medical Subject Headings (MeSH) terms related to cardiopulmonary diseases from these expert-annotated radiology report [225] using the same labeling protocols under the Disease Label Mining section reported in [174].

The dataset was divided into non-overlapping subsets for training (80%) and testing (20%) with patient-level splits. The semantic similarity matrix $\mathcal{S}$ is constructed using the MeSH terms *i.e.* a pair of images are considered similar if they share atleast one MeSH term. These include (% the term is mentioned / % overlap with other terms): normal (44.3 %/ 0 %), opacity (11.8%/ 52.4%), calcified granuloma (8.2%/ 54.3%), calcinosis (8.2%/72.8%), cardiomegaly (8.8%/62.3%), catheters indwelling (2.4%/53,2%), bone-fractures (2.2%/ 48.3%), granulomatous disease (3.4%/ 77.5%), lung hyperdistention (6.5%/ 50%), lung hypoinflation (6.9% / 56.2%), nodule (3.2% / 67.5%), osteophyte (11.8% / 52.4%), scoliosis (4.54% / 55.9%), spine/degenerative (4.5% / 55.9%), spondylosis (2.2% / 42.1%) , surgical instruments (2.5%/ 53.8%) and thoracic vertebrae/degenerative (8.3% / 58.6%). The co-morbidities within the dataset are illustrated as a chord diagram in Fig. 4.10 [1] It must be noted that these MeSH term combinations are chosen carefully as they can be associated with unique image appearance patterns.

[1] The percentages reported in this chapter correspond to the subset of frontal CXR images used in this study, hence differ from the reportings of [174], where they report it over the complete dataset.

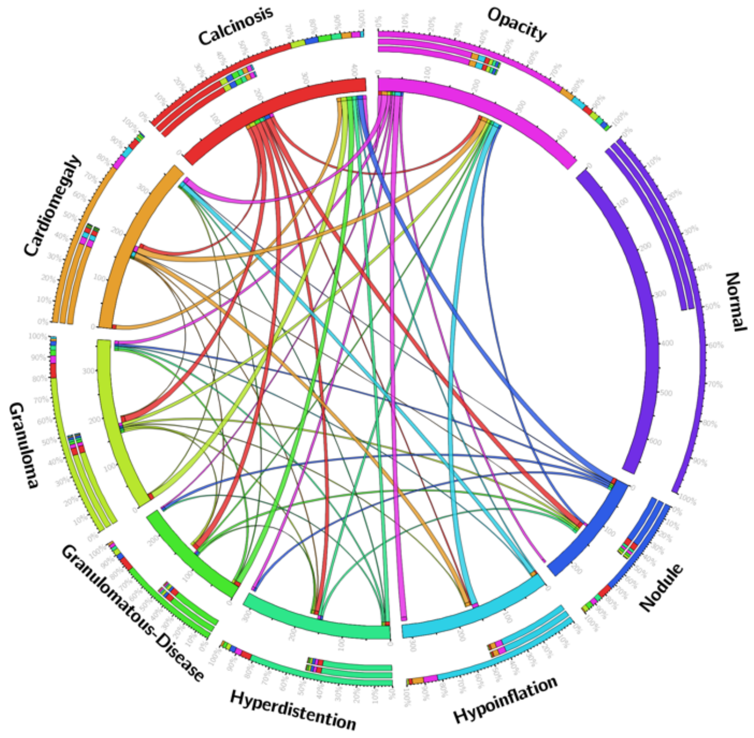


Fig. 4.10. *Schematic illustrating the co-morbidities within the OpenI dataset*: We observe that almost all diseases co-occur with all other pathologies, thus justifying the choice of including a multi-label formulation within this chapter.

4.4.2 Comparative Methods and Baselines

We evaluate and compare the retrieval performance of the proposed DRH network with eight state-of-the art methods including two unsupervised shallow-learning methods: LSH [180], ITQ [66]; two supervised shallow-learning methods: KSH [120] and MHF [40] and four deep learning based methods: AlexNet - KSH (A - KSH) [103] and VGGF - KSH (V - KSH) [26], SFLH [107] and DHN [224]. To justify the proposed formulation, we include simplified four variants of the proposed DRH network as baselines as follows:

- DPH (Deep Plain Net Hashing) by removing the residual connections;
- DRHNQ (Deep Residual Hashing without Quantization) by removing the hashing related losses and generating binary codes only through tanh activation;
- DRN - KSH by training a deep residual network with only the supervised retrieval loss and quantizing through KSH post training and
- DRH - NB which is a variant of DRH where continuous embeddings are used sans quantization, which may act as an upper bound on performance.

The contrastive aspects of aforementioned baseline variants of DRH is tabulated in Table 4.3.

layer name	output size	18-layer	34-layer
conv1	112×112	$[7 \times 7, 64] \times 1$, stride 2, BN, ReLU	
conv2_x	56×56	3×3 max pool, stride 2	
		$\begin{bmatrix} 3 \times 3, 8 \\ 3 \times 3, 8 \end{bmatrix} \times 2$, BN, ReLU	$\begin{bmatrix} 3 \times 3, 8 \\ 3 \times 3, 8 \end{bmatrix} \times 3$, BN, ReLU
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix} \times 2$, BN, ReLU	$\begin{bmatrix} 3 \times 3, 16 \\ 3 \times 3, 16 \end{bmatrix} \times 4$, BN, ReLU
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 2$, BN, ReLU	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 6$, BN, ReLU
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$, BN, ReLU	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$, BN, ReLU
fc	7×7 average pool, $[1 \times 1, 64] \times 1$		
hash	$[1 \times 1,$ code size $] \times 1$, tanh activation		
# parameters			

Tab. 4.2. *Architecture for 18- and 34-layer DRH Networks*: The main difference between the 18- and the 34-layer DRH networks are the network depth as seen from the multiplicity of the convolution blocks (conv3-5) within the architecture design. Here, Brackets ($[\cdot]$) indicate the size of convolutional filters within a particular residual block, with the number of blocks stacked. The image down-sampling is performed by blocks conv3_1, conv4_1 and conv5_1 with stride 2.

We used the standard metrics for evaluating retrieval quality as proposed by Lai *et al.* [107]: Mean Average Precision (MAP) and Precision - Recall Curves varying the code size (16, 32, 48 and 64 bits) (discussed in detail in Sec. 1.6.1). For fair comparison, all the methods were trained and tested on identical data folds. The retrieval performance of methods involving residual learning and baselines is evaluated for two variants varying the number of layers: $(\cdot) - 18$ and $(\cdot) - 34$. The configurational parameters of the network architectures is tabulated in Table 4.2.

For the shallow learning methods, we represent each image as a 512 dimensional GIST vector [142]. For the DRH and associated baselines, the input image is resized to 224×224 and normalized to a dynamic range of 0-1 using the pre-processing steps discussed in [174]. For A-KSH and V-KSH, the image normalization routines were identical to that reported in the original works [103] [26]. We implement all our deep learning networks (including DRH) on the open-source MatConvNet framework [190]. The hyper-parameters λ_q, λ_b and λ_0 were set at 0.05, 0.025 and 0.01 empirically. The momentum term γ was set at 0.9, the initial learning rate η at 10^{-2} and batchsize at 128. The training data was augmented on-the-fly extensively through jittering, rotation and intensity augmentation by matching histograms between images sharing similar co-morbidities. All the comparative deep learning methods were also trained with similar augmentation. Furthermore, for A - KSH and V - KSH variants, we pre-initialized the network parameters from the pre-trained models by removing the final probability layer [103] [26]. These network learnt a 4096-dimensional embedding by fine-tuning it with cross-entropy loss. The hashing was performed explicitly through KSH [120] upon convergence of the network.

Baselines	Hashing Method TS - Two-stage SS - Single-stage	Architecture	Quantization Loss
Deep Residual Network with Kernel Sensitive Hashing (DRN - KSH)	TS	Residual	×
Deep Plain Net Hashing (DPH)	SS	Plain	✓
Deep Residual Hashing without Quantization Loss (DRH-NQ)	SS	Residual	×
Deep Residual Hashing without binarization (DRH - NB)	SS	Residual	×
Deep Residual Hashing (Proposed - DRH)	SS	Residual	✓

Tab. 4.3. *Baseline variants of Deep Residual Hashing*: To evaluate the choice of residual architecture and quantify the loss of retrieval performance due to quantization errors, we contrast against baseline variants of DRH. The baselines differ in terms of how the hashing is performed (Two-stage: representation learning followed by independent hashing and Singe-stage: end-to-end hashing), architecture choice (residual *vs.* plain convolutions) and if quantization loss is used in the objective function.

4.4.3 Evaluation against Comparative Methods and Baselines

The results of the MAP of the Hamming ranking for varying code sizes of all the comparative methods are listed in Table 4.6. We report the precision-recall curves for the comparative methods at a code size of 64 bits in Fig. 4.11 and Fig. 4.12 for the shallow and deep learning based comparative methods respectively. To justify the proposed formulation for DRH, several variants of DRH (namely, DRN - KSH, DPH, DRH - NQ and DRH - NB) were investigated and compare their retrieval results are tabulated in Table 4.4. In addition to MAP, we also report the retrieval precision withing Hamming radius of 2 (P @ H2). The associated precision-recall curves are shown in Fig. 4.14. To investigate the contributions of the proposed loss terms, we performed extensive ablative testing by setting combinations of λ_q, λ_b and λ_o to zero individually. The weight decay term R_W is standard practice in training deep networks, hence not tested ablatively. The MAP and the PR curves for code size of 64 bits are shown in Table 4.5 and Fig. 4.13 respectively.

4.5 Discussion

4.5.1 Trainability of Deep Residual Hashing

The introduction of residual connections offers short-cut connections which act as zero-resistance paths for gradient flow thus effectively mitigating vanishing of gradients as network depth increases. This is strongly substantiated by comparing the performance of DRH - 34 to DRH - 18 *vs.* the plain net variants of the same depth DPH - 34 to DPH - 18 respectively. From Table 4.6, we infer that there is a strong improvement in MAP with increasing depth for DRH of about 9.3%. On the other hand, we observe a degradation of 2.2% MAP performance on increasing layer depth in DPH. The performance of DRH-18 is fractionally better than DPH - 18

Method	MAP		P @ H2	
	18-L	34-L	18-L	34-L
DRN - KSH	54.60	62.50	83.58	87.50
DPH	66.59	64.38	91.50	93.10
DRH - NQ	62.69	66.23	82.37	89.32
DRH - NB	69.21	77.45	95.81	95.64
DRH	67.44	**76.72**	95.56	94.59

Tab. 4.4. *Retrieval Performance of DRH and associated baselines*: We report the mean average precision (MAP) and precision at Hamming distance of 2 (P @ H2) of the Hamming ranking w.r.t. varying network depths for baseline variants of DRH at a fixed code size of 64 bits. Performing hashing end-to-end improves performance significantly over the two-stage baseline (DRN - KSH). Training with quantization loss attempts to improve the performance of DRH over without quantization loss (DRH-NQ) and effectively bridge the gap to training without binarization (DRH-NB) which is an upper-bound on performance. Reprint from [39], with permission of Springer.

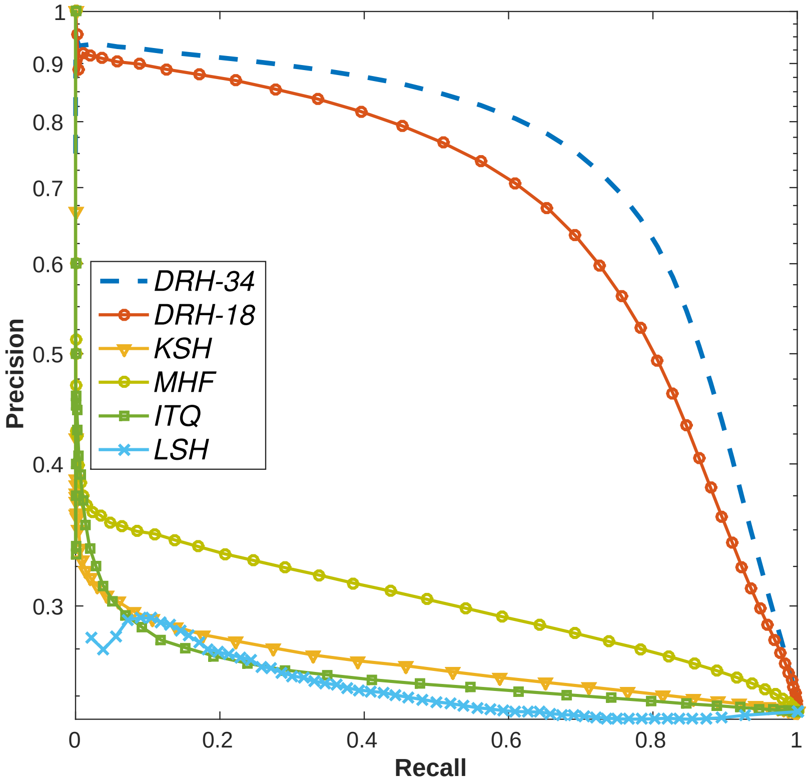


Fig. 4.11. *PR Curves at code size of 64 bits for the shallow comparative methods*: We observe a significant margin between the performance of DRH against shallow learning based comparative methods. This strongly suggests that due to complexity of the underlying semantics generic image features such as (GIST) used for the shallow learning methods fail to capture it sufficiently and motivates the need for end-to-end representation learning based methods (*i.e.* deep learning based methods). Reprint from [39], with permission of Springer.

indicating that DRH exhibits better generalizability and the degradation problem is addressed

Method	λ			MAP	
	λ_q	λ_b	λ_o	18-L	34-L
Ablative Testing DRH - ( )	○	○	○	48.97	55.07
	○	○	●	56.03	60.03
	○	●	○	52.51	55.80
	●	○	○	62.69	66.23
	○	●	●	53.43	62.66
	●	○	●	66.42	71.17
	●	●	○	64.39	72.46
NB	●	●	●	69.21	77.45
Proposed	●	●	●	67.44	**76.72**

Tab. 4.5. *Retrieval Performance of DRH under ablative testing*: We report the mean average precision (MAP) and precision at Hamming distance of 2 (P @ H2) of the Hamming ranking w.r.t. varying network depths for ablative variants of DRH at a fixed code size of 64 bits, selectively choosing if a particular hashing related loss or regularization is used within the optimization. The proposed combination of hashing related loss-terms is demonstrated to improve performance over individual contributions demonstrating that they independently attempt improve the hashing performance. The introduction of the quantization loss is observed to be the most significant contributor to performance gain within DRH. In comparison to the baseline *sans* binarization (NB), we only incur a marginal loss in retrieval performance. Reprint from [39], with permission of Springer.

well as we have significant MAP gains from increased depth. With the introduction of batch normalization and residual connections, we ensure that the signals during forward pass have non-zero variances and that the back propagated gradients exhibit healthy norms. Therefore, neither forward nor backward signals vanish within the network. This is substantiated by the differences in MAP observed in Table 4.6 between methods using BN (DRH, DPH and V-KSH) in comparison to A-KSH which does not use BN.

4.5.2 Representability of Deep Residual Hashing

Ideally, the latent embeddings in the Hamming space should be such that similar samples are mapped closer while simultaneously mapping dissimilar samples further apart. We plot the t-Stochastic Neighbourhood Embeddings (t-SNE) [124] of the hash codes for four comparative methods (GIST - ITQ, VGGF - KSH, DPH - 18 and DRH - 34) in Fig. 4.2 to visually assess the quality of the hash codes generated. Visually, we observe that hand-crafted GIST features with unsupervised hashing method ITQ fail to sufficiently induce semantic separability. In comparison, though VGGF-KSH improves significantly owing to network fine-tuning, better embedding results from DRH - 34 (DPH-18 is highly comparable to DRH-34). Additionally, the significant differences in MAP reported in Table 4.6 between these methods substantiates our hypothesis that in scenarios of limited training data it is better to train smaller models from scratch over fine-tuning to avoid over-fitting (DRH - 34 has 0.183M in comparison to VGGF with 138M parameters). Also the significant domain shift between natural images (ImageNet -

Method	MAP				Time (in ms)
	16 bits	32 bits	48 bits	64 bits	
LSH [180]	22.77	23.93	23.99	24.85	193.7‡
ITQ [66]	25.06	25.19	25.87	**26.23**	194.1‡
MHF [40]	23.62	27.02	30.78	**36.75**	212.3‡
KSH [120]	26.46	32.49	32.01	30.42	198.5‡
A - KSH [103]	35.95	37.28	36.64	39.31	28.28†
V - KSH [26]	47.92	50.64	53.62	52.61	40.45†
SFLH [107]	62.94	63.27	70.48	**73.37**	13.11†
DHN [224]	62.31	50.64	62.38	70.47	13.48†
DPH - 18	48.78	52.13	54.01	**66.59**	4.75†
DPH - 34	46.64	44.43	51.39	64.38	5.08†
DRH - 18	50.93	57.46	62.76	67.44	11.23†
DRH - 34	56.79	65.80	75.81	**76.72**	13.17 †

Tab. 4.6. *Mean Average Precision of Hamming ranking w.r.t. varying code sizes and time for retrieval*: We observe a significant margin between the performance of DRH against shallow learning based comparative methods. This strongly suggests that due to complexity of the underlying semantics generic image features such as (GIST) used for the shallow learning methods fail to capture it sufficiently and motivates the need for end-to-end representation learning based methods (*i.e.* deep learning based methods). Contrasting with deep learning methods, we observe that DRH-34 performs better than SLNH and DHN due to improved optimization. Increasing layer-depth helps significantly improve the performance (contrasting DRH - 18 *vs.* DRH-34). Further, an end-to-end hashing method is in general observed to be superior to two-stage methods that learn representations and hash functions independently (*i.e.* AlexNet+KSH and VGGF+KSH *vs.* rest). Here: † is computed on the GPU and ‡ is the CPU time. Reprint from [39], with permission of Springer.

VGGF) and CXR poses a significant challenge for generalizability of networks fine-tuned from networks trained for the ISLVRC ImageNet recognition challenge [26, 103].

4.5.3 Compactness of Hash Codes

Hashing aims at generating compact representations preserving the semantic relevance to the maximal extent. Varying the code sizes, we observe from Table 4.6 that the MAP performance of majority of the supervised hashing methods improves significantly. In particular for DRH - 34, we observe that the improvement in the performance from 48 bits to 64 bits is only fractional. The performance of DRH - 34 at 32 bits is highly comparable to DRH - 18 at 64 bits. This testifies that with increasing layer depth DRH learns more compact binary embeddings such that shorter codes can already result in good retrieval quality.

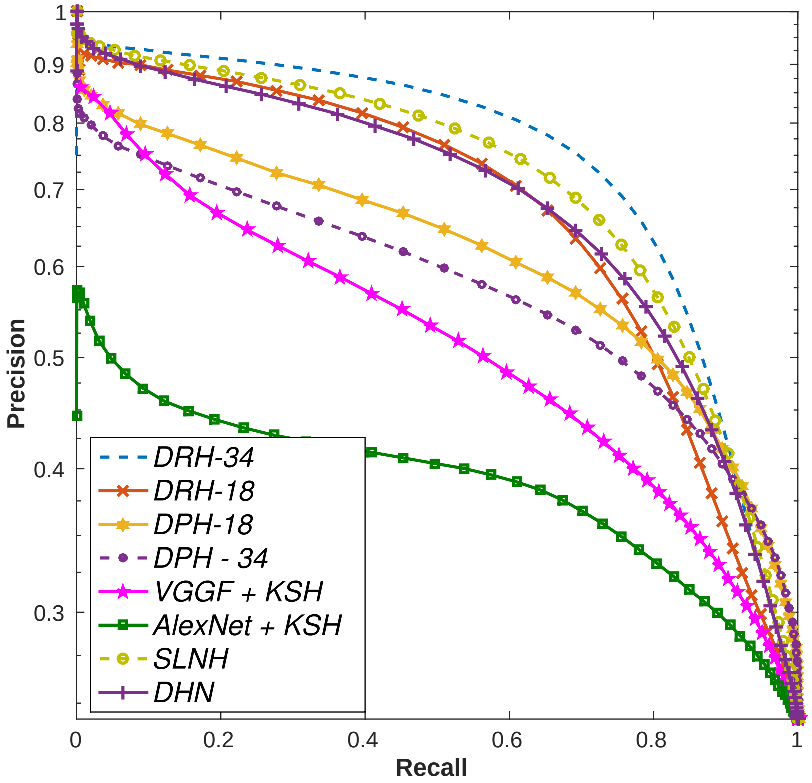


Fig. 4.12. *PR Curves at code size of 64 bits for the deep learning based comparative methods*: Contrasting with deep learning methods, we observe that DRH-34 performs better than SLNH and DHN due to improved optimization. Increasing layer-depth helps significantly improve the performance (contrasting DRH - 18 *vs.* DRH-34). Further, an end-to-end hashing method is in general observed to be superior to two-stage methods that learn representations and hash functions independently (*i.e.* AlexNet+KSH and VGGF+KSH *vs.* rest). Reprint from [39], with permission of Springer.

4.5.4 Semantic Similarity Preservation within Deep Residual Hashing

Visually assessing the t-SNE representation of GIST - ITQ (Fig. 4.2(a)) we can observe that it fails to sufficiently represent the underlying semantic relevance within the CXR images in the latent hamming space, which re-testifies the concerns over hand-crafted features that were raised in Section 1.4.2. VGGF - KSH (Fig. 4.2(b)) improves over GIST - ITQ substantially, however it fails to induce sufficient class-separability. Despite KSH considering pair-wise relationships while learning to hash, the feature representation generated by fine-tuned VGG-F is limited in representability as the cross-entropy loss is evaluated point-wise. Finally, the t-SNE embedding of DRH - 34 shown in Fig. 4.2 visually reaffirms that semantic relevance remains preserved upon embedding and the method generates clusters well separated within the hamming space. The high degree of variance associated with the t-SNE embedding of normal class (red in color) is conformal with the high population variability expected within that class.

Comparing DRH to shallow hashing methods, it is evident from significant gap in the PR curves (see Fig. 4.11), that GIST features fail to capture the semantic concepts despite introduction of supervised hashing (KSH and MHF). However, this is mitigated with end-to-end learning in

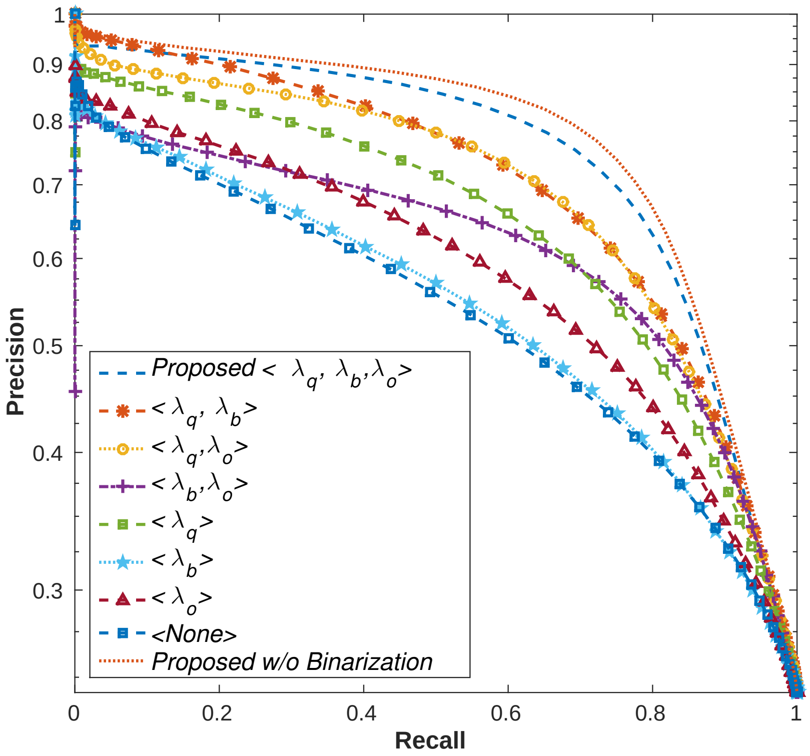


Fig. 4.13. *PR Curves at code size of 64 bits for ablative baseline variants of DRH*: The proposed combination of hashing related loss-terms is demonstrated to improve performance over individual contributions demonstrating that they independently attempt improve the hashing performance. The introduction of the quantization loss is observed to be the most significant contributor to performance gain within DRH. Reprint from [39], with permission of Springer.

deep hashing methods as shown in Fig. 4.12. Particularly, DRH outperforms state-of-the art methods (both SFLH and DHN) as well as A - KSH and V - KSH. This clearly demonstrates that simultaneous representation learning for hashing is preferred. Drawing comparisons from Table 4.6, we observe that at code-sizes larger than 16 bits, DRH-34 consistently outperforms SFLH and DHN (despite the network architecture for all the methods being residual in nature and of same depth). This singles out the proposed loss combinations to be better than triplet loss (SFLH) or pair-wise cross entropy (DHN), for the problem at hand.

Fig. 4.1 demonstrates the first five retrieval results sorted according to their Hamming rank for four randomly selected CXR images from the testing set. In particular, for Case (d), where we observe that the top neighbors (d 1-5) share at least one co-occurring pathology. For cases (a), (b) and (c), all the top five retrieved neighbors share the same class.

4.5.5 Joint Optimization for Improved Hash Code Quality

The main contribution of the work hinges on the hypothesis that performing an end-to-end learning of hash codes is better than a two stage learning process. Comparative validations, presented in Table 4.6 and Fig. 4.12 against the two-stage deep learning methods, as (A - KSH, V - KSH and baseline variant DRN - KSH) strongly support this hypothesis. In particular, we observe over 14.2% improvement in MAP comparing DRN - KSH (34 - L) to DRH - 34.

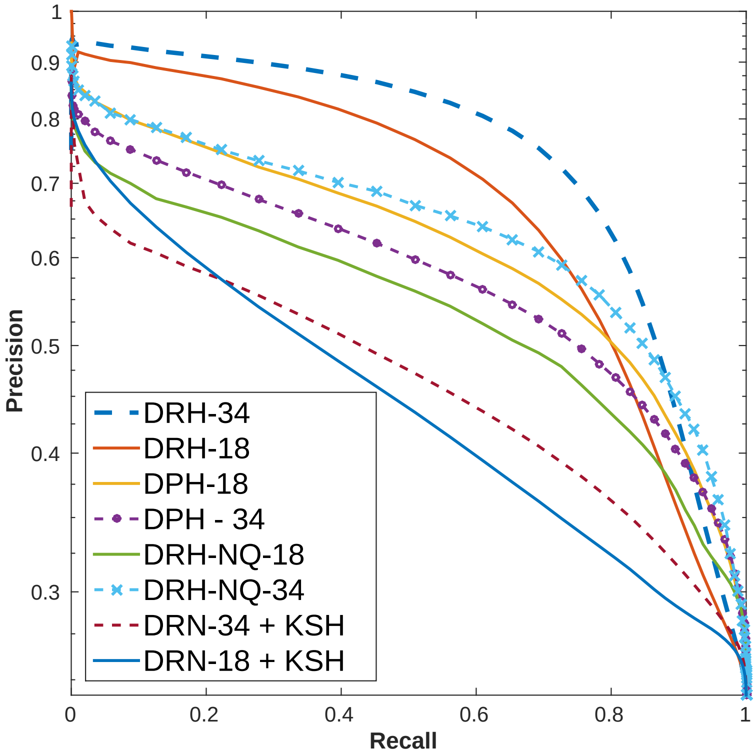


Fig. 4.14. *PR Curves at code size of 64 bits for baseline variants of DRH*: Performing hashing end-to-end improves performance significantly over the two-stage baseline (DRN - KSH). Training with quantization loss attempts to improve the performance of DRH over without quantization loss (DRH-NQ) and effectively bridge the gap to training without binarization (DRH-NB) which is an upper-bound on performance. Reprint from [39], with permission of Springer.

This difference in performance may be owed to a crucial disadvantage of DRN - KSH that the generated feature representation is not optimally compatible to binararization. We can also observe from Table 4.4 and Fig. 4.14 that, DRH - 18 and DRH - 34 incur very small average MAP decrease of 1.8% and 0.7% when binarizing hash codes against non-binarized continuous embeddings in DRH - NB- 18 and DRH - NB - 34 respectively. In contrast, DRH - NQ suffers from very large MAP decreases of 6.6% and 10.8% in comparison to DRH - B. These observations validate the need for the proposed quantization loss as it leads to nearly lossless binarization.

To investigate the contributions of the proposed loss terms, we performed extensive ablative testing by setting combinations of λ_q, λ_b and λ_o to zero individually and reported the results of MAP in Table 4.5 and the associated PR-curve for a code size of 64 bits in Fig. 4.13. The weight decay term R_W is standard practice in training deep networks, hence not tested ablatively. From, Table 4.5, we observe that amongst the hashing related losses, the quantization loss (J_Q) is of primordial importance (DRH - 34 without J_Q under-performs by 14% w.r.t. with J_Q). Orthogonalization (R_O) boosts the performance by 5% over the baseline sans any additional losses for DRH-34, implying that mutual independence of hash bits is improved. Interestingly, the bit balance loss (J_B) improves performance only marginally (0.7%) for DRH-34, but significantly for DRH-18 (3.5%). It must be noted that combinations with at least two losses improve consistently over individual baselines, substantiating effectiveness of the proposed loss function and the complementary nature of the individual losses.

5 Robust Multiple Instance Hashing

> *We are all agreed that your theory is crazy. The question that divides us is whether it is crazy enough to have a chance of being correct.*
>
> — **Neils Bohr**
> (Columbia University, 1958)

5.1 Overview and Publications

In this chapter, we introduce a multiple instance (MI) deep hashing technique, termed Robust Multiple Instance Hashing (RMIH), for learning discriminative hash codes with weak bag-level supervision suited for large-scale retrieval. We learn such hash codes by aggregating deeply learnt hierarchical representations across bag members through an MI pool layer. For better trainability and retrieval quality, we propose a two-pronged approach that includes robust optimization and training with an auxiliary single instance hashing arm which is down-regulated gradually. We pose retrieval for tumor assessment as an MI problem because tumors often coexist with benign masses and could exhibit complementary signatures when scanned from different anatomical views. Experimental validations demonstrate improved retrieval performance over the state-of-the-art methods for two large-scale breast carcinoma assessment datasets on mammography and histology.

The chapter is organized as follows: in Sec. 5.2 we present the clinical motivation behind retrieval with multiple instances and present state-of-the art methods that deal with multiple instance classification and retrieval. Having laid the premise for multiple instance retrieval, in Sec. 5.3 we present the methodology behind leveraging deep learning for the task of multiple instance hashing. Particularly, in Sec. 5.3.1 the architecture for RMIH is introduced and we discuss specific variations proposed for training and testing such a network and in Sec. 5.3.2 we mathematically formulate the optimization objective including the aspect of incorporating robustness for improved immunity against label noise. In the subsequent section Sec. 5.4, we present aspects of the target multiple instance retrieval databases (mammography and histology) and discuss various model settings and validations proposed to evaluate the contributions within this chapter. The results of such validations are presented and discussed in Sec. 5.5. In particular, we discuss the effects of training with auxiliary losses in Sec. 5.5.1, the effect of introducing robustness 5.5.2, of the quantization loss in improving hash code quality in Sec. 5.5.3 and compare and contrast the performance against state-of-the art MI hashing methods and MI variants of single instance hashing methods in Sec. 5.5.4.

Substantial parts of this chapter have already been published in the following article and are quoted verbatim:

[37] **Conjeti, Sailesh**, Magdalini Paschali, Amin Katouzian, and Nassir Navab. "Deep Multiple Instance Hashing for Scalable Medical Image Retrieval." *In International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 550-558. Springer, Cham, 2017.*

5.2 Motivation

Breast carcinoma is the second leading cause of cancer-related deaths among women. Early diagnosis is reported to significantly improve the five-year survival rate from 24% when the cancer is diagnosed at a distant stage to 99% if diagnosed at a localized stage [91]. A range of imaging modalities and biomarkers have been proposed and clinically adopted for breast cancer screening including non-invasive modalities such as mammography, elastography, theromography, MRI, CT, ultrasound *etc.* Upon identification of actionable tumors in such screening modalities, staging is performed through invasive modalities (biopsy) such as fine-needle aspiration cytology, histopathology *etc.* In this section, we introduce and discuss the motivation towards developing CBIR systems targeting two of the aforementioned modalities *viz.* mammography for screening and histopathology for staging and diagnosis.

Mammography is widely considered as the most effective and widely accepted method for breast cancer screening. Interestingly, the modality received a gold standard status for breast cancer detection by the American Cancer Society [182]. Disease specific manifestations as seen on mammography are majorly masses and micro calcifications. Amongst these, masses are observed to have a high variability in shape, size, texture, margin and are often co-occurring with surrounding tissue, which makes their detection and staging challenging. Interpretation of mammograms is also fraught with substantial inter- and intra-observer variability. Redondo et al. [158] reported the the observer variability, in a study across 21 expert radiologists interpreting 100 mammograms, as fair for the BI-RADS assessment (concordance score κ of 0.37). Upon collapsing categories into suspect normal/benign (Categories I and II) and actionable (III, 0, IV, V), the agreement moderately improved to κ =0.53 across all categories. The intra-observer agreement for BI-RADS assessment was observed to be moderate (κ = 0.3) and substantial on recall (κ = 0.66). In addition to these, a considerable portion of retrospectively visible masses are missed by radiologists and unnecessary biopsies are frequently conducted on normal tissues. Owing to the high clinical significance and the challenging nature of mammography mass detection, a number of computer aided diagnostics (CAD) approaches have been proposed [62, 89, 153].

Histology still stands as the *gold standard* for assessment of carcinoma in breast. Effective management of breast cancer necessitates need for early detection and treatment. The pre-cancerous stages of breast cancer are typically categorized into the lobular and ductal subtypes, with the majority of pre-invasive and invasive cancers belonging to the latter category [167]. Page et al. [144] in a study on atypical hyperplastic breast lesions, categorized intraductal lesions into three major classes namely, usual ductal hyperplasia (UDH), atypical ductal hyperplasia (ADH) and ductal carcinoma *in situ* (DCIS). ADH and DCIS are deemed as precursor lesions while UDH is not. Typically, patients diagnosed with UDH on core-biopsy are recommended for routine follow-up, whereas those with ADH and DCIS are subjected to excisional biopsy, which is invasive and uncomfortable and is associated with significant health-care costs [223]. Under such settings, developing an objective tool for comprehensive assessment of histology slides is needed, so that it offers more reliable and consistent analysis of histopathology images [56]. In this context, a reference based assessment, such as presenting prior cases with similar disease manifestations (termed Content Based Image Retrieval (CBIR)) could be used to circumvent discrepancies in cancer grading. With growing sizes of clinical databases, such a CBIR system ought to be both scalable and accurate. Towards this, hashing approaches for CBIR are being actively investigated for representing images as compact binary codes that can be used for fast and accurate retrieval [39, 120, 222].

Malignant carcinomas are often co-located with potentially benign looking manifestations and suspect normal tissues. In such cases, describing the whole image with a single label is often inadequate for objective machine learning and alternatively requires expert annotations delineating the exact location of the tumor or region of interest. This argument extends to screening modalities like mammograms, where multiple anatomical views are acquired. In such scenarios, the status of the tumor is best represented to a CBIR system by constituting a bag of all associated images, thus veritably becoming multiple instance (MI) in nature. This is illustrated in Fig. 5.1. With this as our premise we present, for the first time, a novel deep learning based MI hashing method, termed as Robust Multiple Instance Hashing (RMIH). Yang *et al.* were the first to extend hashing methods to MI learning scenarios with two variants: Instance Level MI Hashing (IMIH) and Bag Level MI Hashing (BMIH) [208]. However, these approaches are not end-to-end and are susceptible to semantic gap between features and associated concepts. Alternatively, deep hashing methods such as simultaneous feature learning and hashing (SFLH) [107], deep hashing networks (DHN) [224] and deep residual hashing (DRH) [39] to name a few, propose the learning of representations and hash codes in an end-to-end fashion, in effect bridging this semantic gap. It must be noted that all the above deep hashing works targeted single instance (SI) hashing scenarios and an extension to MI hashing was not investigated.

Earlier works on MI deep learning in computer vision include work by Wu *et al.* [203], where the concept of an MI pooling (MIPool) layer is introduced to aggregate representations for multi-label classification. Yan *et al.* leveraged MI deep learning for efficient body part recognition [207]. Unlike MI classification that potentially substitutes the decision of the clinician, retrieval aims at presenting them with richer contextual information similar to the case at hand to facilitate decision-making. RMIH effectively bridges the two concepts for CBIR systems by combining the representation learning strength of deep MI learning with the potential for scalability arising from hashing. Within CBIR for breast cancer, notable prior art includes work on mammogram image retrieval by Jiang *et al.* [92] and large-scale histology retrieval by Zhang *et al.* [222]. Both these works pose CBIR as an SI retrieval problem. Contrasting with [92] and [222], within RMIH we create a bag of images to represent a particular pathological case and generate a bag-level hash code, as shown in Fig. 5.2. Our contributions in this chapter include:

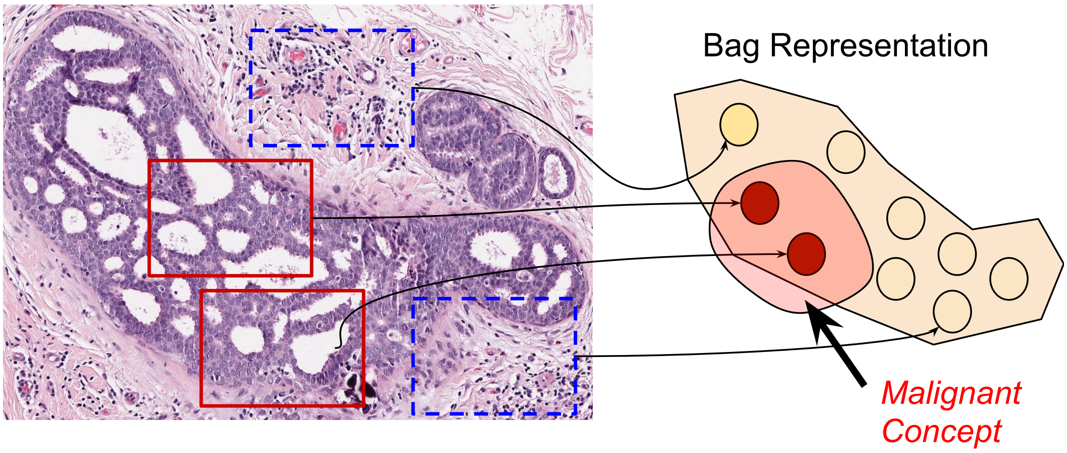


Fig. 5.1. *Schematic illustrating construction of image bags*: Here, an example of a large region of interest (∼ 6K × 4K) labeled as *malignant* is shown wherein a few patches overlapping with the actual tumor represent the underlying malignant concept while proximal patches are potentially benign or less discriminative connective / lipidic tissues. It must be noted that these are not individually identified and only a bag-level weak annotation is available for learning.

- introduction of a robust supervised retrieval loss for learning in presence of weak labels and potential outliers;
- propose the training of networks with an auxiliary SI arm with gradual loss trade-off for improved trainability; and
- incorporation of the MIPool layer to aggregate representations across variable number of instances within a bag, generating bag-level discriminative hash codes.

5.3 Methodology

Lets consider database $\mathcal{B} = \{B_1, \ldots, B_{N_B}\}$ with N_B bags. Each bag, B_i, with varying number (n_i) of instances (I_i) is denoted as $B_i = \{I_1, \ldots, I_{n_i}\}$. We aim at learning $\mathcal{H}$ that maps each bag to a K-d Hamming space $\mathcal{H} : \mathcal{B} \rightarrow \{-1, 1\}^K$, such that *bags with similar instances and labels are mapped to similar codes*. Fig. 5.2 presents an overview of how RMIH is used to generate hash codes for a bag of images. For supervised learning of $\mathcal{H}$, we define a bag-level pairwise similarity matrix $\mathcal{S}^{\text{MI}} = \{s_{ij}\}_{ij=1}^{N_B}$, such that $s_{ij} = 1$ if the bags are similar and zero otherwise. In applications, such as this one, where retrieval ground truth is unavailable we can use classification labels as a surrogate for generating $\mathcal{S}^{\text{MI}}$.

5.3.1 Architecture for RMIH

As shown in Fig. 5.4, the proposed RMIH framework consists of a deep CNN terminating in a fully connected layer (FCL). Its outputs $\{z_{ij}\}_{j=1}^{n_i}$ are fed into the MIPool layer to generate the aggregated representation $\hat{z}_i$ that is pooled ($\max_{\forall j} \{z_{ij}\}_{j=1}^{n_i}$, mean$(\cdot)$, *etc.*) across instances within the bag (as illustrated in Fig. 5.5). $\hat{z}_i$ is an embedding in the space of the bags and is the

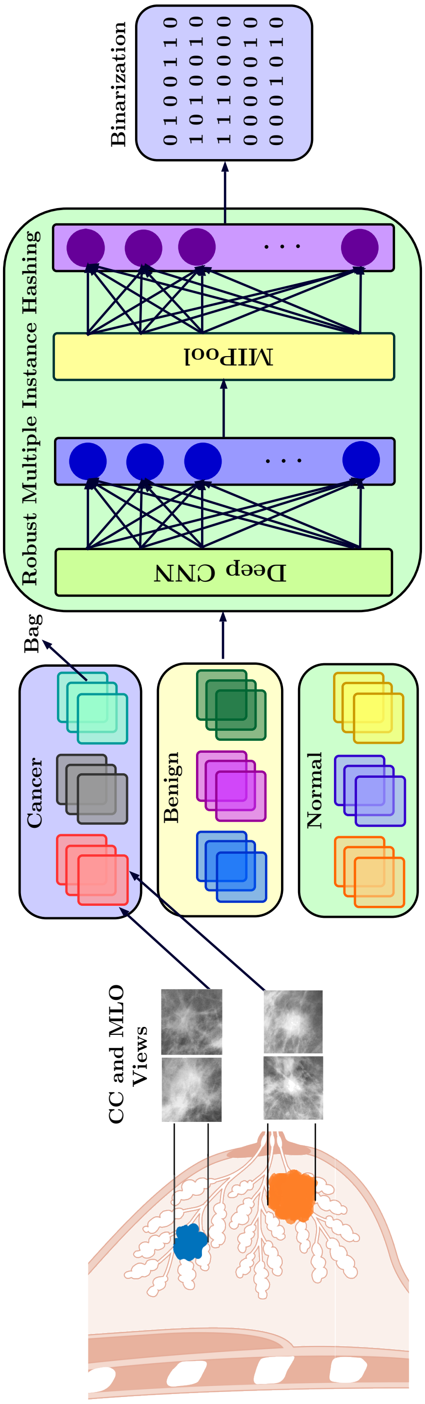


Fig. 5.2. *Overview of RMIH for end-to-end generation of bag-level hash codes*: We are given multiple images representing the same anatomy, thus constituting a *bag*. The images of the bag traverse through the deep CNN layers and the global features (*fully-connected*) at image-level are pooled across the bag to create a bag-level feature vector. This feature vector is further binarized using hashing layers thus resulting in a bag-level hash code. The breast anatomy image is attributed to Cancer Research UK/Wikimedia Commons. Reprint from [37], with permission of Springer.

input of a fully connected MI hashing layer. The output of this layer is squashed to $[-1, 1]$ by passing it through a $\tanh\{\cdot\}$ function to generate h_i^{MI}, which is quantized to produce bag-level hash codes as $\mathbf{b}_i^{\text{MI}} = \text{sgn}\ (\mathbf{h}_i^{\text{MI}})$. The deep CNN mentioned earlier could be a pretrained

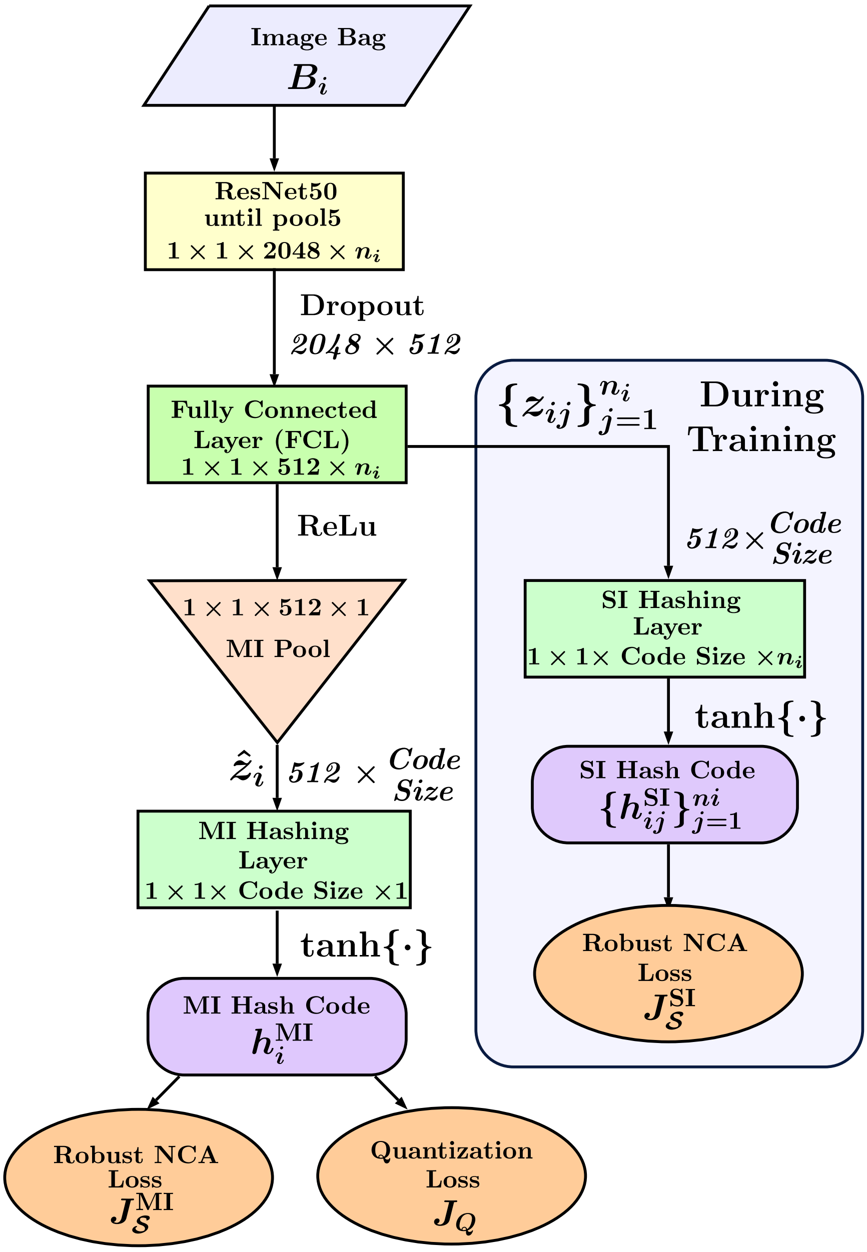


Fig. 5.3. *Architecture for Robust Multiple Instance Hashing for Training*: Given an image bag $B_i = \{I_j\}_{j=1}^{n_i}$ with n_i images, the individual images I_j independently traverse through the Deep CNN (with ResNet-50 [73] architecture and pre-training) until the pool5 layer, resulting in a $2048 \times n_i$ tensor representation for the bag. Following, they are reduced in dimensionality to $512 \times n_i$ *via* the fully connected layer (FCL) layer to generate $\{z_{ij}\}_{j=1}^{n_i}$. The network branches into two arms beyond this layer. Within the multiple instance arm, the image-level tensors are pooled into a 512-dimensional tensor $\hat{z}_i$, which is then fed to the MI hashing layer with a tangent hyperbolic activation function to generate the bag-level hash code h_i^{MI}. During training, gradients from the Robust NCA loss J_S^{MI} and the quantization loss J_Q are fed into this arm. To alleviate the gradient sparsification due to the MI Pool layer, the single instance hashing arm is fed with $\{z_{ij}\}_{j=1}^{n_i}$ from the FCL layer and hashed through the SI hashing layer to generate instance-wise hash codes $\left\{h_{ij}^{\mathrm{SI}}\right\}_{j=1}^{n_i}$. During training this arm is fed through gradients from the single-instance variant of the robust NCA loss J_S^{SI}. Reprint from [37], with permission of Springer.

network, such as VGGF [26], GoogleNet [187], ResNet50 (R50) [73] or an application specific network.

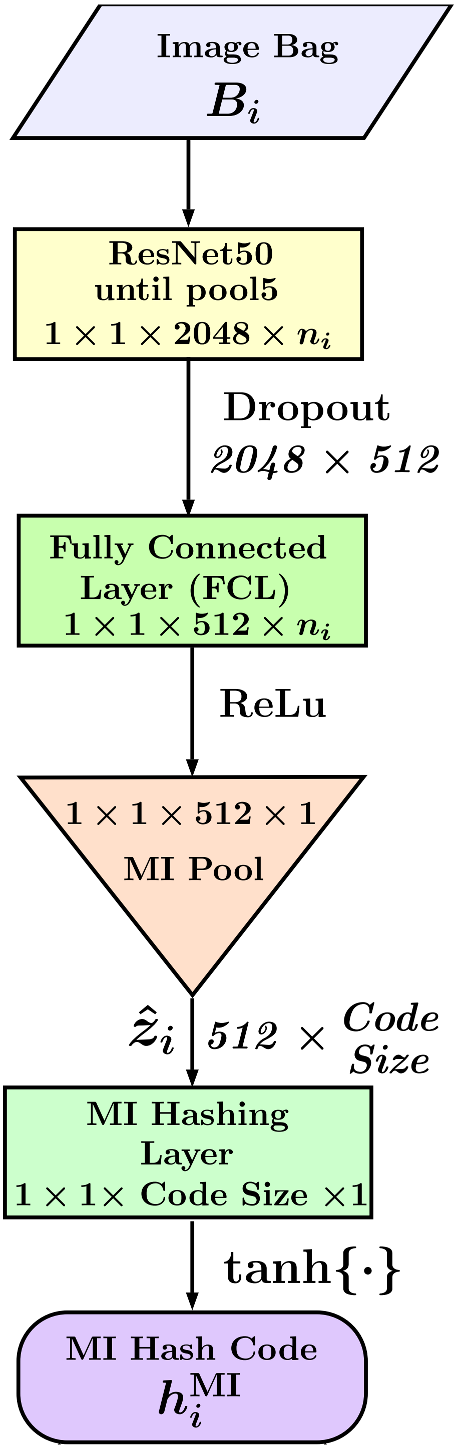


Fig. 5.4. *Architecture for Robust Multiple Instance Hashing for Testing*: Given an image bag $B_i = \{I_j\}_{j=1}^{n_i}$ with n_i images, the individual images I_j independently traverse through the Deep CNN (with ResNet-50 [73] architecture and pre-training) until the pool5 layer, resulting in a $2048 \times n_i$ tensor representation for the bag. Following, they are reduced in dimensionality to $512 \times n_i$ *via* the fully connected layer (FCL) layer to generate $\{z_{ij}\}_{j=1}^{n_i}$. The network branches into two arms beyond this layer. Within the multiple instance arm, the image-level tensors are pooled into a 512-dimensional tensor $\hat{z}_i$, which is then fed to the MI hashing layer with a tangent hyperbolic activation function to generate the bag-level hash code h_i^{MI}.

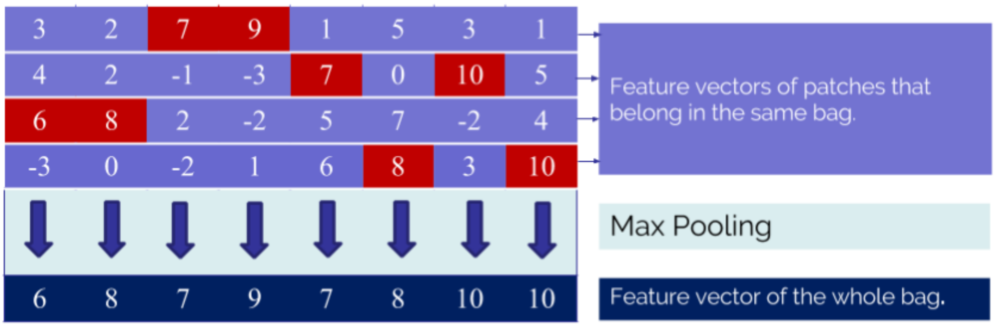


Fig. 5.5. *Schematic demonstrating the multiple instance pooling operation*: In this example, global instance-wise feature vectors extracted through a deep CNN are max-pooled along each feature to constitute a bag-level feature representation. Such a pooling operation is referred to as Multiple Instance pooling across this chapter.

During training of RMIH, we introduce an auxiliary SI hashing (aux-SI) arm, as shown in Fig. 5.3. It taps off at the FCL layer and feeds directly into a fully connected SI hashing layer with tanh$\{\cdot\}$ activation to generate instance level non-quantized hash codes, denoted as $\{h_{ij}^{\text{SI}}\}_{j=1}^{n_i}$. While training RMIH using backpropagation, the MIPool layer significantly sparsifies the gradients (analogous to using very high dropout while training CNNs), thus limiting the trainability of the preceding layers (illustratively shown in Fig. 5.6a). The SI hashing arm helps to potentially mitigate this by producing auxiliary instance level gradients which add to the gradients from the MI pool layer and *densify* them (shown in Fig. 5.6b).

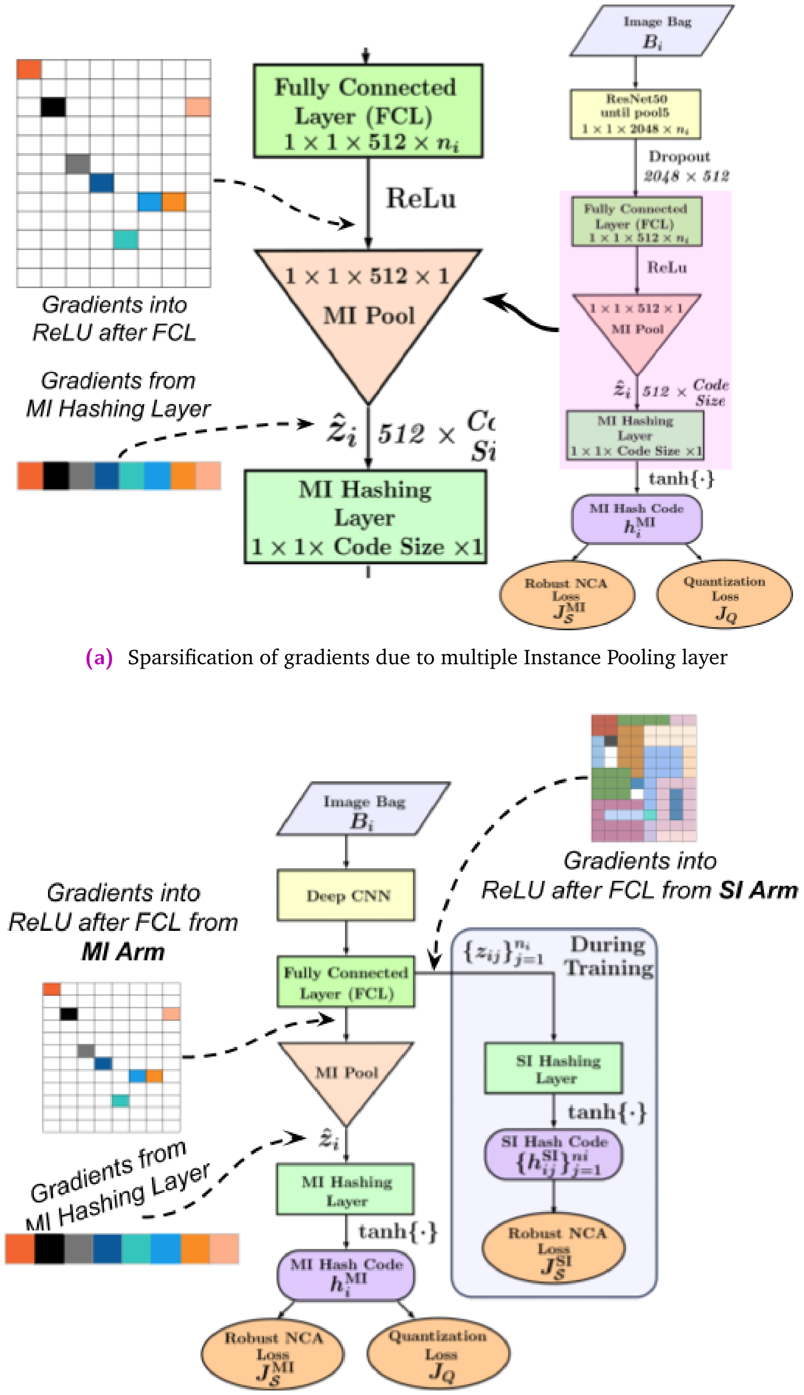
5.3.2 Model Learning and Robust Optimization

To learn similarity preserving hash codes, we propose a robust version of supervised retrieval loss based on neighborhood component analysis employed by [189]. The motivation to introduce robustness within the loss function is two-fold:

- robustness induces immunity to potentially noisy labels due to high inter-observer variability and limited reproducibility for the applications at hand [56];
- it can effectively counter ambiguous label assignment while training with the aux-SI hashing arm.

Given $\mathcal{S}^{\text{MI}}$, the robust supervised retrieval loss $J_{\mathcal{S}}^{\text{MI}}$ is defined as:

$$J_{\mathcal{S}}^{\text{MI}} = 1 - \frac{1}{N_B^2} \sum_{i,j=1}^{N_B} s_{ij} p_{ij} \tag{5.1}$$


Fully Connected Layer (FCL) $1 \times 1 \times 512 \times n_i$
ReLu
$1 \times 1 \times 512 \times 1$
MI Pool
Gradients into ReLU after FCL
Gradients from MI Hashing Layer
$\hat{z}_i$ 512 × Co Si
MI Hashing Layer $1 \times 1\times$ Code Size $\times 1$
Image Bag B_i
ResNet50 until pool5 $1 \times 1 \times 2048 \times n_i$
Dropout 2048×512
Fully Connected Layer (FCL) $1 \times 1 \times 512 \times n_i$
ReLu
$1 \times 1 \times 512 \times 1$
MI Pool
$\hat{z}_i$ 512 × Code Size
MI Hashing Layer $1 \times 1\times$ Code Size $\times 1$
$\tanh\{\cdot\}$
MI Hash Code h_i^{MI}
Robust NCA Loss J_S^{MI}
Quantization Loss J_Q

(a) Sparsification of gradients due to multiple Instance Pooling layer

Image Bag B_i
Deep CNN
Fully Connected Layer (FCL)
MI Pool
$\hat{z}_i$
MI Hashing Layer
$\tanh\{\cdot\}$
MI Hash Code h_i^{MI}
Robust NCA Loss J_S^{MI}
Quantization Loss J_Q
Gradients into ReLU after FCL from **MI Arm**
Gradients from MI Hashing Layer
Gradients into ReLU after FCL from **SI Arm**
$\{z_{ij}\}_{j=1}^{n_i}$
During Training
SI Hashing Layer
$\tanh\{\cdot\}$
SI Hash Code $\{h_{ij}^{\mathrm{SI}}\}_{j=1}^{ni}$
Robust NCA Loss J_S^{SI}

(b) Auxiliary gradients from the single instance arm

Fig. 5.6. *Schematic illustrating gradient sparsification and auxiliary gradients*: Due to the multiple instance pooling layer, the gradient arising from the hinter part of the network are sparsified during training (illustrated in Fig. 5.6a). This limits the trainability of really deep networks. To mitigate this and densify the gradients, an auxiliary single instance hashing arm is introduced within the architecture, which contributed auxiliary gradients in addition to ones from the multiple instance hashing arm (illustrated in Fig. 5.6b).

where p_{ij} is the probability that two bags (indexed as i and j) are neighbors. Given hash codes $\mathbf{h_i} = \{h_i^k\}_{k=1}^K$ and $\mathbf{h_j}$, we define a bit-wise residual operation r_{ij} as $r_{ij}^k = (h_i^k - h_j^k)$. We estimate p_{ij} as:

$$p_{ij} = \frac{e^{-\mathcal{L}_{\text{Huber}}(\mathbf{h_i},\mathbf{h_j})}}{\sum_{i\neq l}^{N_B} e^{-\mathcal{L}_{\text{Huber}}(\mathbf{h_i},\mathbf{h_l})}}, \text{ where } \mathcal{L}_{\text{Huber}}(\mathbf{h_i},\mathbf{h_j}) = \sum_{\forall k} \rho_k(r_{ij}^k) \qquad (5.2)$$

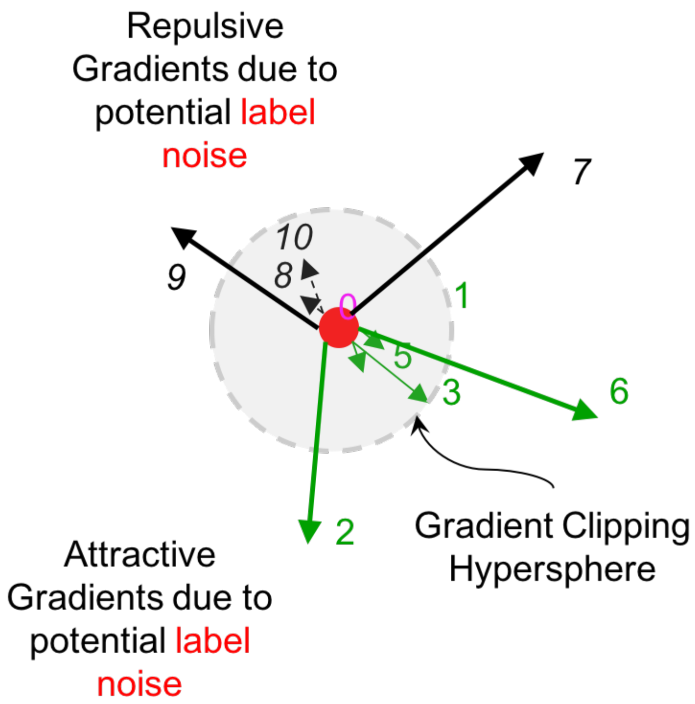

Repulsive Gradients due to potential label noise
7
10
9
8
0
1
5
3
6
2
Gradient Clipping Hypersphere
Attractive Gradients due to potential label noise

Fig. 5.7. *Noisy gradients during training of RMIH*: Both repulsive and attractive can arise due to potential label noise during the training of RMIH. The introduction of the single instance hashing arm for improved trainability has a counter-effect of introducing significant label noise, thus resulting in such potentially malicious gradients. To improve immunity towards it, the robustness is introduced in the objective function (Eq. 5.2 and Eq. 5.3), which during back-propagation of gradients clips potentially noisy gradients to an adaptively tuned gradient clipping hypersphere (parametrized by c_k.

The Huber norm's robustness operation ρ_k is defined as:

$$\rho_k(r_{ij}^k) = \begin{cases} \frac{1}{2}(r_{ij}^k)^2, & \text{if } \mid r_{ij}^k \mid \leqslant c_k \\ c_k \mid r_{ij}^k \mid -\frac{1}{2}c_k^2, & \text{if } \mid r_{ij}^k \mid > c_k \end{cases} \tag{5.3}$$

In Eq. (5.3), the tuning factor c_k is estimated inherently from the data and is set to c_k, estimated as:

$$c_k = 1.345 \times \sigma_k \tag{5.4}$$

The factor of 1.345 is chosen to provide approximately 95% asymptotic efficiency and σ_k is a robust measure of bit-wise variance of r_{ij}^k. Specifically, σ_k is estimated as 1.485 times the median absolute deviation of r_{ij}^k as empirically suggested in [86]. This robust formulation provides immunity to outliers during training by clipping their gradients (as shown in Fig. 5.7 and Fig. 5.8). For training with the aux-SI hashing arm, we employ a similar robust retrieval loss J_S^{SI} defined over single instances with bag-labels assigned to member instances.

$$J_S^{\text{SI}} = 1 - \frac{1}{N^2} \sum_{m,n=1}^{N} s_{mn} p_{mn} \tag{5.5}$$

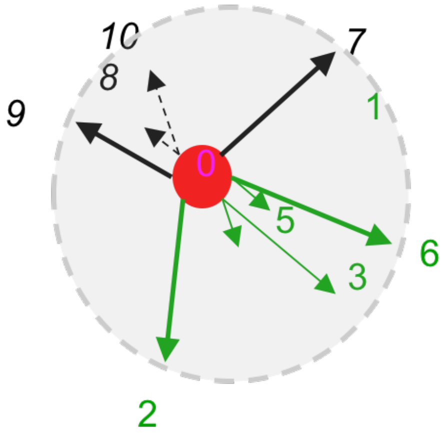


Fig. 5.8. *Noisy gradients are clipped during back-propagation in RMIH*: Due to the robustness within the optimization objective, potentially noisy gradients with magnitude beyond c_k are clipped to c_k during backpropagation and model-learning of RMIH (Eq. 5.11). The threshold c_k is adaptively computed from within the gradients of a batch. Clipping such *potentially noisy* gradients provides immunity against potential label noise and improper class assignment owing to the addition of the single instance auxiliary arm.

To minimize loss of retrieval quality due to quantization, we use a differentiable quantization loss J_Q proposed in [224] and is estimated as:

$$J_Q = \sum_{i=1}^{M} (\log \cosh(\mid \mathbf{h_i} \mid - \mathbf{1})) \tag{5.6}$$

This loss also counters the effect of using continuous relaxation in definition of p_{ij} over using Hamming distance. As a standard practice in deep learning, we also add an additional weight decay regularization term R_W, which is the Frobenius norm of the weights and biases, to regularize the cost function and avoid over-fitting.

The following composite loss is used to train RMIH:

$$J = \lambda^t_{\text{MI}} J^{\text{MI}}_S + \lambda^t_{\text{SI}} J^{\text{SI}}_S + \lambda_q J_Q + \lambda_w R_W \tag{5.7}$$

where λ^t_{MI}, λ^t_{SI}, λ_q and λ_w are hyper-parameters that control the contribution of each of the loss terms.

Specifically, λ^t_{MI} and λ^t_{SI} control the trade-off between the MI and SI hashing losses. The SI arm plays a significant role only in the early stages of training and can be traded off eventually to avoid sub-optimal MI hashing. For this we introduce a weight trade-off formulation that gradually down-regulates λ^t_{SI}, while simultaneously up-regulating λ^t_{MI}. Here, we use:

$$\lambda^t_{\text{SI}} = 1 - 0.5 \left(1 - \frac{t}{t_{\text{max}}}\right)^2 \tag{5.8}$$

$$\lambda^t_{\text{MI}} = 1 - \lambda^t_{\text{SI}} \tag{5.9}$$

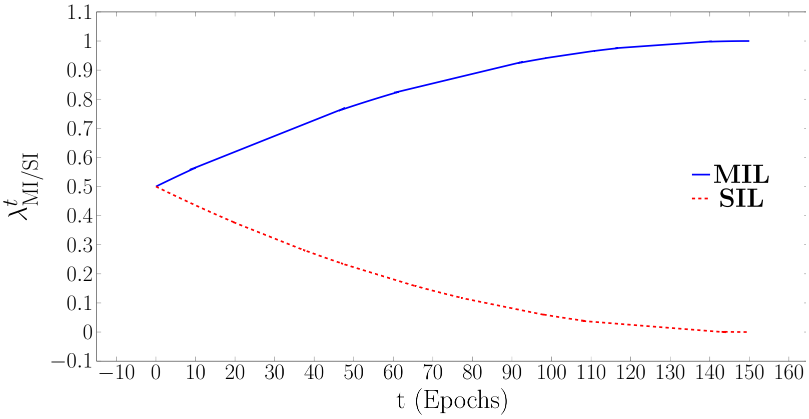


Fig. 5.9. *Weight trade-off between MI and SI arms for training RMIH*: The weighting factors λ^t_{MI} and λ^t_{SI} control the relative contributions of the MI and SI arms towards the overall cost function J (in Eq. 5.7). During training of RMIH, with epochs, we gradually down-regulates λ^t_{SI}, while simultaneously up-regulating λ^t_{MI}, such that the auxiliary gradients from the noisy SI arm help with initial convergence and is gradually traded-off for an improved multiple-instance hashing performance oblivious to how well the SI hashing performs. Reprint from [37], with permission of Springer.

where t is the current epoch and $t_{\max}$ is the maximum number of epochs (see Fig. 5.9). We train RMIH with mini-batch stochastic gradient descent (SGD) with momentum. Due to potential outliers that can occur at the beginning of training, we scale c_k up by a factor of 7 for $t = 1$ to allow a stable state to be reached. Specifically, the gradient of $J^{(\cdot)}_{\mathcal{S}}$ w.r.t. to $\mathbf{h}_i$ is derived as:

$$\begin{aligned}\frac{\partial J^{(\cdot)}_{\mathcal{S}}}{\partial \mathbf{h}_i} = &\left(\sum_{l:s_{li}>0} p_{li}\mathcal{L}'_{\mathcal{H}}(\mathbf{h}_l,\mathbf{h}_i) - \sum_{l\neq i}\left(\sum_{q:s_{lq}>0} p_{lq}\right) p_{li}\mathcal{L}'_{\mathcal{H}}(\mathbf{h}_l,\mathbf{h}_i)\right) \\ &- \left(\sum_{j:s_{ij}>0} p_{ij}\mathcal{L}'_{\mathcal{H}}(\mathbf{h}_i,\mathbf{h}_j) - \sum_{j:s_{ij}>0} p_{ij}\left(\sum_{z\neq i} p_{iz}\mathcal{L}'_{\mathcal{H}}(\mathbf{h}_i,\mathbf{h}_z)\right)\right)\end{aligned} \tag{5.10}$$

where $\mathcal{L}'_{\mathcal{H}}(\mathbf{h_i},\mathbf{h_j}) = \{\rho'_k(r^k_{ij})\}^k_{k=1}$. The derivative of the huber term $\rho_k{}'(r^k_{ij})$ can be computed as:

$$\rho'_k(r^k_{ij}) = \begin{cases} r^k_{ij}, & \text{if } \mid r^k_{ij} \mid \leqslant c_k \\ c_k\ \mathrm{sgn}(r^k_{ij}), & \text{if } \mid r^k_{ij} \mid > c_k \end{cases} \tag{5.11}$$

Regarding the quantization loss function, the derivative can be computed as follows:

$$\frac{\partial J_Q}{\partial \mathbf{h}_i} = \tanh\left(|\mathbf{h}_i| - \mathbf{1}\right)\mathrm{sgn}\left(\mathbf{h}_i\right) \tag{5.12}$$

Having computed these gradients, we use back propagation to compute the derivatives of the preceding layers.

5.4 Experiments

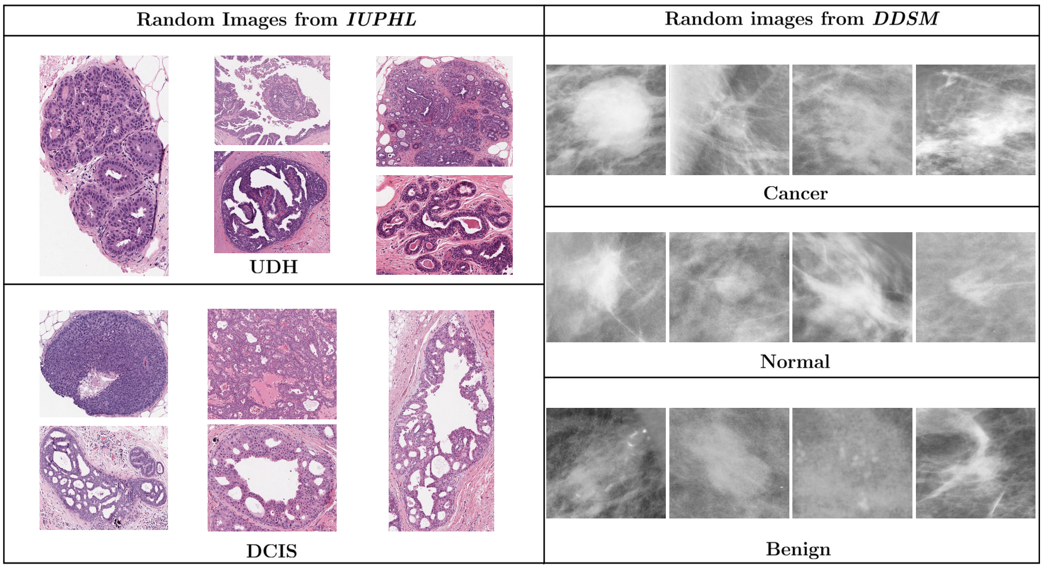


Fig. 5.10. *Example images of breast mammography and histology*: Select images from the histology database (*IPUHL*) and mammography database (*DDSM*) are shown to showcase the degree of anatomical variability within and across the classes. Within the *IUPHL* database, there exists significant differences in the size of the whole slide images and variations in staining patterns. Within the *DDSM* database, within a particular semantic class there exists differences with the scanning instruments used (especially X-ray windowing) and significant anatomical variability associated with lesion type, presence/absence of micro-calcification and extent of confounding fatty tissue in the background of the lesions.

5.4.1 Databases for Multiple Instance Retrieval

Clinical applicability of RMIH has been validated on two large scale datasets, namely, Digital Database for Screening Mammography (DDSM) [77, 92] and a retrospectively acquired histology dataset from the Indiana University Health Pathology Lab (*IUPHL*) [57, 222]. Fig. 5.10 illustrates select images from the two datasets to showcase anatomical variability within and across the constituent classes.

Mammography

The *DDSM* dataset comprises of 11,617 expert selected regions of interest (ROI) curated from 1861 patients. Multiple ROIs associated with a single breast from anatomical views constitute a bag (size: 1-12; median: 2), which has been annotated as normal, benign or malignant by expert radiologists. A bag labeled *malignant* could potentially contain multiple suspect normal and benign masses, which have not been individually identified.

Histology

The *IUPHL* dataset is a collection of 653 ROIs from histology slides from 40 patients (20 with precancerous ductal hyperplasia (UDH) and rest with ductal carcinoma *in situ* (DCIS)) with ROI level annotations done by expert histopathologists. Due to high variability in sizes of these ROIs (upto 9K × 8K pixels), we extract multiple patches (of size 1024×1024) and populate a ROI-level bag (size: 1-15; median: 8). As cellular and nuclei level characteristics are important to distinguishing DCIS from UDH, it is not recommended to rescale these images to standard input sizes used by CNNs (typically, 244 × 224 in [26, 73, 187]). It must be noted that patches from an ROI labeled as DCIS could contain UDH-like manifestations, hence associating the DCIS label individually to them could lead to potentially misleading results. From both the datasets, we use patient-level non-overlapping splits to constitute the training (80%) and testing (20%) sets.

5.4.2 Model Settings and Validations

To validate proposed contributions, namely robustness within NCA loss and trade-off from the aux-SI arm, we perform ablative testing with combinations of their baseline variants by fine-tuning multiple network architectures. Additionally, we compare RMIH against four state-of-the art methods: ITQ [66], KSH [120], SFLH [107] and DHN [224]. For a fair comparison, we use R50 for both SFLH and DHN, since as discussed later it performs the best. Since SFLH and DHN were originally proposed for SI hashing, we introduce additional MI variants by hashing through the MIPool layer. For ITQ and KSH, we further create two comparative settings as follows:

- Using IMIH [208] that learns instance-level hash codes followed by bag-level distance computation;
- Utilizing BMIH [208] using bag-level kernalized representations followed by binarization.

For IMIH and SI variants of SFLH, DHN and RMIH, given two bags B_p and B_q with SI hash codes, say $\mathcal{H}(B_q) = \{h_{q1}, \ldots, h_{qM}\}$ and $\mathcal{H}(B_p) = \{h_{p1}, \ldots, h_{pN}\}$, the bag-level distance is computed as:

$$d(B_p, B_q) = \frac{1}{M}\sum_{i=1}^{M}(\min_{\forall j} \text{ Hamming}(h_{pi}, h_{qj})). \tag{5.13}$$

All images were resized to 224×224 and training data were augmented to create equally balanced classes. λ_{MI}^t and λ_{SI}^t were set assuming t_{max} as 150 epoch; λ_q and λ_w were set at 0.05 and 0.001 respectively. The momentum term within SGD was set to 0.9 and batch size to 128 for *DDSM* and 32 for *IUPHL*. For efficient learning, we use an exponentially decaying learning rate initialized at 0.01. The RMIH framework was implemented in MatConvNet [190]. We use standard retrieval quality metrics: nearest neighbor classification accuracy (nnCA) and precision-recall (PR) curves to perform the aforementioned comparisons. The results (nnCA) from ablative testing and comparative methods are tabulated in Table 5.2 and Table 5.7 respectively. Within Table 5.7, methods were evaluated at two different code sizes (16 bits and 32 bits). We also present the PR curves of select bag-level methods (32 bits) in Fig. 5.11 and Fig. 5.12 for the mammography and histology datasets respectively. For ease of understanding, Table 5.2 has been broken down into Table 5.3, Table 5.4, Table 5.5 and Table 5.6 to discuss the effects of introducing an auxiliary SI arm during training, to contrast the contributions of introducing a loss trade-off, evaluate the effect of including robustness within the supervised loss function and understand the effect of quantization in the hash code generation respectively.

Method		**Variants**			***DDSM***			***IUPHL***		
		R	**T**	**P**	**VGGF**	**ResNet50**	**GoogleNet**	**VGGF**	**ResNet50**	**GoogLeNet**
Ablative Testing	A	○	○	○	62.66	70.92	68.96	80.21	79.69	82.29
	B	○	●	○	71.51	74.25	72.02	85.42	89.06	86.46
	C	●	○	○	68.96	73.27	72.72	84.90	88.02	85.42
	D	●	●	○	78.67	82.31	76.83	87.50	89.58	**89.06**
	E	○	○	●	68.65	72.76	71.70	83.85	85.42	82.29
	F	○	●	●	75.38	77.34	72.92	85.94	90.10	88.02
	G	●	○	●	70.65	76.63	70.02	83.33	85.94	86.46
	H	○	■	●	66.65	69.67	68.26	83.33	88.54	84.90
	I	●	■	●	67.05	76.59	72.84	84.38	89.58	85.42
RMIH		●	●	●	**81.21**	**85.68**	**78.67**	**91.67**	**95.83**	88.02
RMIH($\lambda_q = 0$)		●	●	●	75.34	79.88	73.06	87.50	89.58	88.51
RMIH NB		●	●	●	83.25	88.02	79.06	94.79	96.35	92.71
Legend	**R(Robustness)**	$\circ = L_2$, $\bullet = L_{\text{Huber}}$								
	T(Trade-off)	○ = Equal weights, ● = Decaying SIL weights, ■ = No SIL branch								
	P(Aggregation)	○ = Mean pooling, ● = Max pooling								

Tab. 5.2. *Ablative testing and comparative analysis with baselines of RMIH*: For a code size of 16 bits, we report the nearest neighbor classification accuracy estimated over unseen test data. We ablatively test RMIH against variants in terms of use of robustness within the optimization objective, if a trade-off was employed while training with the single instance arm and the nature of the pooling function to aggregate instance level features into bag level. We also contrast RMIH against baselines without the quantization loss and an upper bound baseline which does not employ any binarization. Reprint from [37], with permission of Springer.

5.5 Results and Discussion

In the following section we will showcase the superiority of RMIH through discussion of the results of ablative testing and the comparison of RMIH with the state of the art methods mentioned earlier. The testing objective of the ablative testing can be summarized as follows:

- *Effect of auxiliary SI hashing arm*: Compare with baselines sans SI hashing arm.
- *Contribution of robustness*: Contrast with baseline with classical NCA loss (L_2 norm).
- *Loss Trade-off*: Compare with baseline setting $\lambda^t_{\text{MI}} = \lambda^t_{\text{SI}} = 0.5$.
- *Quantization*: Contrast with baseline setting $\lambda_q = 0$.

5.5.1 Effect of auxiliary single instance arm

To justify using the aux-SI loss, we introduce a variant of RMIH without it (E in Table 5.3), which leads to a significant decline of 3% to 14% in contrast to RMIH. This could be potentially attributed to the prevention of the gradient sparsification caused by the MIPool layer. From Table 5.3, we observe a 3%-10% increase in performance, comparing cases with gradual decaying trade-off (B) against baseline setting ($\lambda^t_{\text{MI}} = \lambda^t_{\text{SI}} = 0.5$, A,C).

Method		Variants		*DDSM*			*IUPHL*		
		R	T	VGGF	R50	GN	VGGF	R50	GN
Ablative Testing	D	○	■	66.65	69.67	68.26	83.33	88.54	84.90
	E	●	■	67.05	76.59	72.84	84.38	89.58	85.42
RMIH		●	●	**81.21**	**85.68**	**78.67**	**91.67**	**95.83**	88.02
Legend	**R(Robustness)**	○ = L_2, ● = L_{Huber}							
	T(Trade-off)	○ = Equal weights, ● = Decaying SIL weights, ■ = No SIL branch							
	Networks	R50: ResNet50, GN: GoogleNet							

Tab. 5.3. **Effect of auxiliary Single Instance Arm**: We observe 3% - 14% improvement for RMIH over baseline without the SI hashing arm. Improved representation learning ability in presence of limited training data and sparsification of gradients through MIPool. Modified from [37], with permission of Springer.

Method		Variants		*DDSM*			*IUPHL*		
		R	T	VGGF	R50	GN	VGGF	R50	GN
Ablative Testing	A	○	○	68.65	72.76	71.70	83.85	85.42	82.29
	B	○	●	75.38	77.34	72.92	85.94	90.10	88.02
	C	●	○	70.65	76.63	70.02	83.33	85.94	86.46
RMIH		●	●	**81.21**	**85.68**	**78.67**	**91.67**	**95.83**	88.02
Legend	**R(Robustness)**	○ = L_2, ● = L_{Huber}							
	T(Trade-off)	○ = Equal weights, ● = Decaying SIL weights, ■ = No SIL branch							
	Networks	R50: ResNet50, GN: GoogleNet							

Tab. 5.4. **Effect of loss tradeoff**: We observe improvement in range of 3-10% improvement for RMIH over baseline without trade-off. Learning without trade-off potentially leads to sub-optimal MI hashing as SI loss might compete. Modified from [37], with permission of Springer.

5.5.2 Effect of Robustness

For robust-NCA, we compared against the original NCA formulation proposed in [189] (A,B,D in Table 5.5). Robustness helps handle potentially noisy MI labels, inconsistencies within a bag (like non-informative patches) and the ambiguity in assigning SI labels. Comparing the effect of robustness for baselines sans the SI hashing arm (D *vs.* E as shown in Table 5.5) we observe marginally positive improvement across the architectures and datasets, with a substantial 7% in ResNet50 for *DDSM*. Robustness contributes more with the addition of the aux-SI hash arm (proposed *vs.* E) with improved performance in the range of 4%-5% across all settings. This observation further validates our prior argument.

Method		**Variants**		*DDSM*			*IUPHL*		
		R	T	VGGF	R50	GN	VGGF	R50	GN
Ablative Testing	B	○	●	75.38	77.34	72.92	85.94	90.10	88.02
	D	○	■	66.65	69.67	68.26	83.33	88.54	84.90
	E	●	■	67.05	76.59	72.84	84.38	89.58	85.42
RMIH		●	●	**81.21**	**85.68**	**78.67**	**91.67**	**95.83**	88.02
Legend	**R(Robustness)**	○ = L_2, ● = L_{Huber}							
	T(Trade-off)	○ = Equal weights, ● = Decaying SIL weights, ■ = No SIL branch							
	Networks	R50: ResNet50, GN: GoogleNet							

Tab. 5.5. **Effect of robustness of loss function**: We observe improvement in range of 5-7% improvement for RMIH over baseline without robustness. Better learning in presence of label ambiguity and noise. Modified from [37], with permission of Springer.

Method		**Variants**		*DDSM*			*IUPHL*		
		R	T	VGGF	R50	GN	VGGF	R50	GN
RMIH		●	●	**81.21**	**85.68**	**78.67**	**91.67**	**95.83**	88.02
RMIH($\lambda_q = 0$)		●	●	75.34	79.88	73.06	87.50	89.58	88.51
RMIH NB		●	●	83.25	88.02	79.06	94.79	96.35	92.71
Legend	**R(Robustness)**	○ = L_2, ● = L_{Huber}							
	T(Trade-off)	○ = Equal weights, ● = Decaying SIL weights, ■ = No SIL branch							
	Networks	R50: ResNet50, GN: GoogleNet							

Tab. 5.6. **Effect of Quantization**: Learning without quantization loss leads to decrease of 3% to 5% in retrieval performance. Comparing with upper-bound (non-quantized embeddings), marginal decrease of 2% to 4%. Modified from [37], with permission of Springer.

5.5.3 Effect of Quantization

To assess the effect of quantization, we define two baselines: (1) setting $\lambda_q = 0$ and (2) using non-quantized hash codes for retrieval (RMIH - NB). The latter potentially acts as an upper bound for performance evaluation. From Table 5.6, we observe a consistent increase in performance by margins of 3%-5% if RMIH is learnt with an explicit quantization loss to limit the associated error. It must also be noted that comparing with RMIH - NB, there is only a marginal fall in performance (2%-4%), which is desired.

Within Table 5.2, we also evaluated two variants of the MIPool layer's aggregation functions, namely mean$(\cdot)$ and max$(\cdot)$. ($P = \bullet$ for max$(\cdot)$ and $P = \circ$ for mean$(\cdot)$). In all settings (A *vs.* E, B *vs.* F, C *vs.* G and D *vs.* proposed in Table 5.2), the choice of max pooling was observed to boost performance. except for GoogleNet in *IUPHL*) This can in part be justified by the the fact that max$(\cdot)$ is more selective, thus allowing more discriminative feature representation.

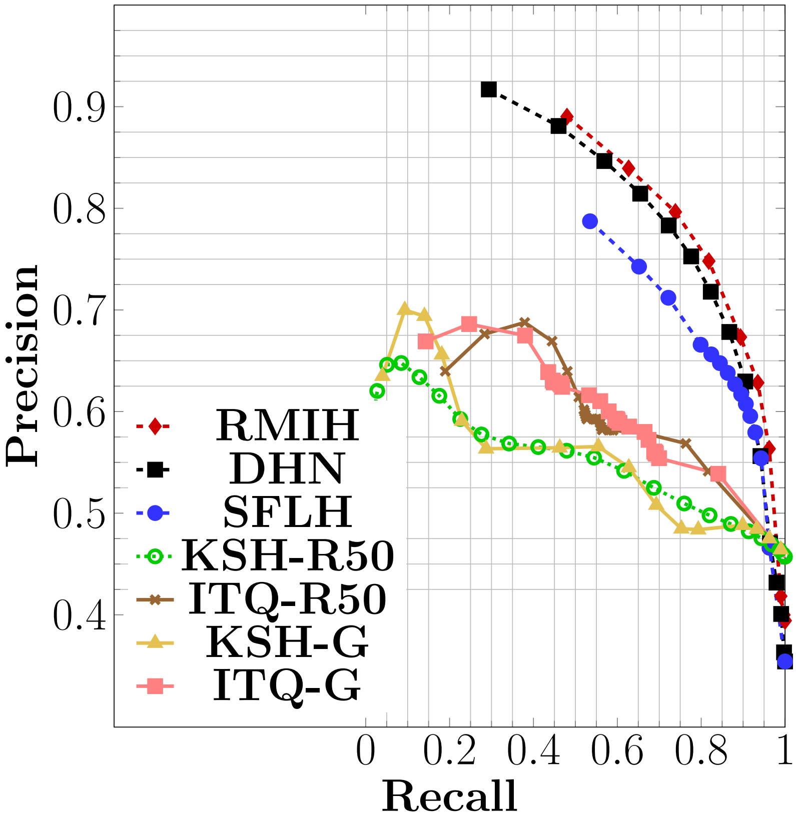


Fig. 5.11. *Precision-recall curves for retrieval performance in DDSM database*: For a code size of 32 bits, we observe that RMIH has the best area under the PR curve closely followed by the multiple instance variant of deep hashing network. We note a significant margin of improvement over two-stage hashing approaches. Reprint from [37], with permission of Springer.

This is necessary in the cases of detecting malignancy, which potentially co-exists in the same bag with a variety of suspect normal and benign tissues.

As a whole, the two-pronged proposed approach, including robustness and trade-off, along with quantization loss delivers the highest performance, proving that RMIH is able to learn effectively, despite the ambiguity induced by the SI hashing arm. Fig. 5.13 demonstrates the retrieval performance of RMIH on the target databases. For *IUPHL*, the retrieved images are semantically similar to the query as consistent anatomical signatures are evident in the retrieved neighbors. For *DDSM*, in the cancer and normal cases the retrieved neighbors are consistent, however it is hard to distinguish between benign and malignant. The retrieval time for a single query for RMIH was observed at 31.62 ms (for *IUPHL*) and 17.48 ms (for *DDSM*) showing potential for fast and scalable search.

From an overall perspective, we observe a consistent performance gap between the *DDSM* and *IUPHL* datasets. This could be attributed to the following reasons:

- learning with pre-trained architectures (VGGF [26], ResNet 50 [73] and GoogLeNet [187]) that have been trained for RGB images collected in the wild could be potentially biased to histology images against grayscale mammogram images in DDSM

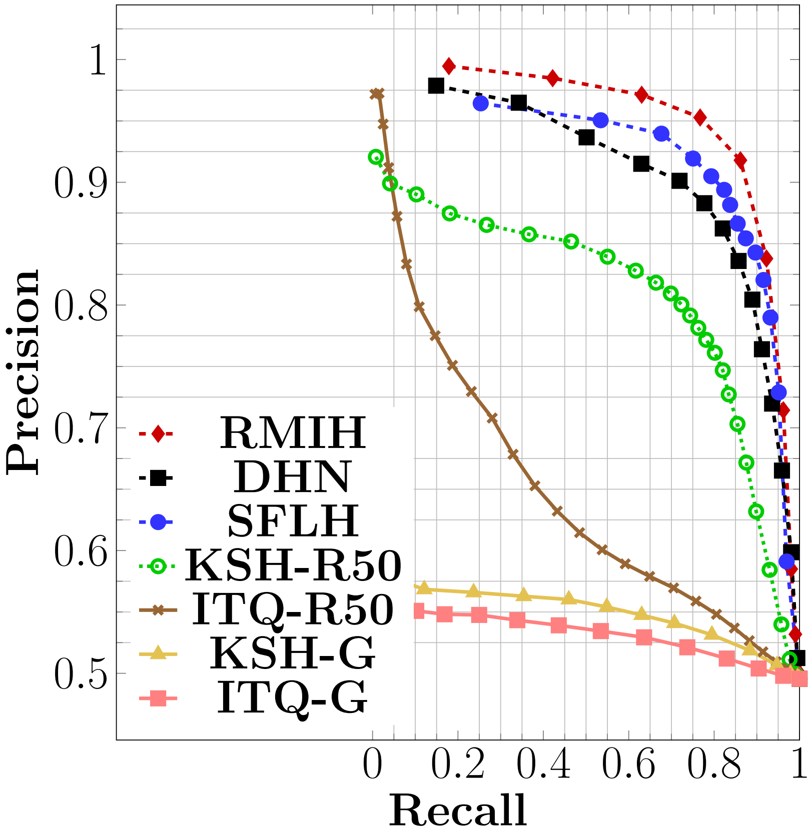


Fig. 5.12. *Precision-recall curves for retrieval performance in IUPHL database*: For a code size of 32 bits, we observe that RMIH has the best area under the PR curve closely followed by the multiple instance variant of SFLH. We note a significant margin of improvement over two-stage hashing approaches and note that the choice of GIST features fails to capture fine-grained semantics within the dataset despite introduction of supervision during hashing. Reprint from [37], with permission of Springer.

- mammogram is used as a screening modality and is more prone to ambiguity in interpretation against invasive histopathology.

This also reflects in sample retrieval cases shown in Fig. 5.13, where actionable cases (benign and malignant) are hard to distinguish. However, in case of *IUPHL*, the retrieved images are more semantically similar owing to consistent signatures of anatomical gland-like structures and cellular diversity.

5.5.4 Comparative Methods

In the contrastive experiments (tabulated in Table 5.7) against ITQ and KSH, hand-crafted GIST [142] features underperformed significantly, while the improvement with the R50 features ranged from 5%-30%. However, RMIH still performed 10%-25% better, proving that even if deep learnt features severely boost the performance, the shallow methods cannot fully breach the gap to the deep ones. Comparing the SI with the MI variations of DHN, SFLH and RMIH in Table 5.7, it is observed that the performance improved in the range of 3%-11%, suggesting that end-to-end learning of MI hash codes is preferred over two-stage hashing *i.e.* hashing at SI level and comparing at bag level with Eq. (5.13). However, RMIH fares

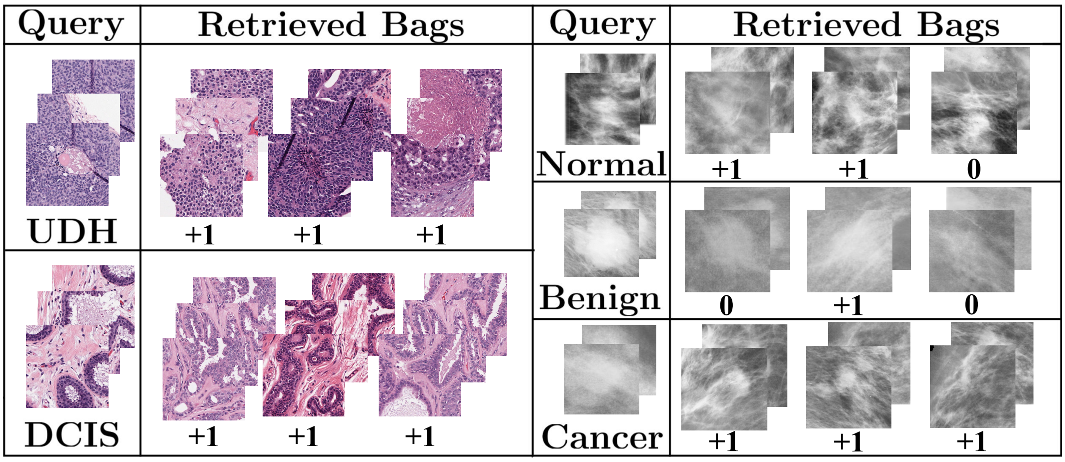


Fig. 5.13. *Retrieval results for RMIH at code size 16 bits*: When presented with an unseen bag of images representing the query, RMIH generates a bag-level hash code which is in turn compared against the learnt hash table to fetch similar image bags. With visual inspection, we can observe that within the *IUPHL* dataset, the retrieval is observed to preserve the semantics well. In case of mammography (*DDSM*), the performance of retrieval on the non-actionable cases needs further improvement and investigation. This is potentially due to label noise as mammography labels are fraught with high inter- and intra-observer variability. Here, +1 indicates retrieval from class consistent with query and -1 indicates otherwise. Reprint from [37], with permission of Springer.

comparably better than both the SI and MI versions of SFLH and DHN, owing to the robustness of the proposed retrieval loss function to potentially noisy labels and inconsistent instances within bags (also evident from PR curves in Fig. 5.11 and Fig. 5.12). In all fairness, the concepts of training with aux-SI hashing arm with gradual trade-off and robustness could be potentially adapted to SFLH and DHN to improve their MI hashing performance. As also seen from the associated PR curves in Fig. 5.11 and Fig. 5.12, the performance gap between shallow and deep hashing methods remains significant despite using R50 features. Comparative results strongly support our premise that end-to-end learning of MI hash codes is preferred over conventional two-stage approaches.

	Method	A/F	B/I	*DDSM*		*IUPHL*		*Time* (in mS)	
				16-bit	32-bit	16-bit	32-bit	*DDSM*	*IUPHL*
Shallow	ITQ [66]	R50	○	66.35	67.71	78.58	80.28	15	28
		R50	●	64.56	71.98	89.58	79.69	16	29
		G	○	65.22	66.55	51.79	51.42	9	21
		G	●	59.73	61.03	57.29	58.85	11	23
	KSH [120]	R50	○	61.88	64.81	87.74	86.51	17	30
		R50	●	59.81	72.17	70.83	80.21	18	32
		G	○	60.50	61.91	57.36	57.83	11	24
		G	●	55.34	55.67	60.94	58.85	14	25
Deep	SFLH [107]	R50	○	73.54	77.46	83.33	85.94	19	32
		R50M	●	71.98	75.93	85.42	88.54	18	31
	DHN [224]	R50	○	65.64	74.79	82.29	86.46	18	32
		R50M	●	72.88	80.43	88.02	90.62	17	30
	RMIH-SIL	R50	○	76.02	78.37	87.92	88.58	20	34
	RMIH	R50M	●	**85.68**	**89.47**	**95.83**	**93.23**	17	31
Legend	**A/F:**	**A**: Architecture, **F**: Features R50: ResNet, R50M: ResNet+MIPool, G: GIST							
	B/L:	**B**: Bag Level, **I**: Instance Level ○ = Instance Level, ● = Bag Level							

Tab. 5.7. *Comparison of RMIH against state of the art deep hashing methods*: Comparing SI variant of deep hashing methods to their MI variants, we observe that the retrieval performance improves significantly, which justifies formulation as a MI retrieval problem. Further, there exists a significant performance gap between two stage hashing methods in comparison to end-to-end hashing. The time for retrieval with RMIH is highly competitive and comparable to other deep hashing methods. Reprint from [37], with permission of Springer.

Part III

Conclusion

Conclusions 6

In this thesis, several advanced machine learning models have be presented for learning efficient hash functions that can be leveraged for large-scale retrieval. The presented mathematical formulations are generic and within this thesis we particularly focus on medical image retrieval. This chapter will summarize the contributions of this work, all of which have been successfully validated on a range of large-scale retrieval tasks, and provide an outlook to potential future work.

Medical image retrieval is a particularly challenging topic of investigation due to domain-specific challenges such as how to incorporate medical context associated with these images, how to effectively factor anatomical variability out, how to deal with potential complexities associated with image acquisition such as artifacts, scanner variability *etc.* In addition to the aforementioned challenges, medical images often have very high dimensionality and limited availability. However, investigating medical image retrieval also presents with a lot of opportunities as technically they are ultra-fine grained data and have significant impact in image-guided diagnosis, clinical-decision support, evidence-based medicine, precision medicine, pedagogical goals within medical education and effective management of clinical data. Particularly, the contributions within this thesis target applications that generate data in large-scale and aims at developing learning models that facilitate efficient clinical reasoning in such scenarios using techniques for scalable and accurate retrieval. We envisage that adopting these hashing methods into clinical practice can facilitate real-time querying and feedback for challenging cases by providing the clinician with fast access to similar past cases and enable a structured exploration of large-scale medical databases.

6.1 Unsupervised Hashing Forests

In Chapter 2, we demonstrated that random forests can be suitably modified for the task of hashing and leveraged for the task of large-scale retrieval within heterogeneous neuroscientific image databases. Hashing forests are ensemble of dedicated hashing trees that recursively parse and encode the feature space with binary similarity preserving code words generated by hashing functions that enable efficient query, retrieval and analysis. Particularly for the task of neuron retrieval, we observe that the use of ensemble of trees and hierarchical tree-structure makes hashing forests more robust to noisy neuromorphological features (observed due to inconsistent 3D digital reconstruction of neuron). We establish that the proposed HF formulation has superior neighborhood approximation and retrieval performance in comparison to existing generalized hashing methods by quantifying the results over 31266 neuron reconstructions dataset curated from 147 different archives. To the best of our knowledge, this is the first research to present hashing in neuroscientific databases and demonstrates higher flexibility for reference-based retrieval over existing alternative methods [228]. The proposed formulation utilizes inverse coding in HF, which helps avoiding pairwise comparisons across the database while retrieving, without compromising on accuracy. With the inclusion of oblique split functions in conjunction with cluster validity measures, we ensure that the native neighborhoods and clusters within the neuromorphological space are maximally preserved during hashing.

Towards the future work, improving efficiency of hashing forests includes investigating improvements to the node-scoring functions like introducing maximum-margin approaches to improve parsing of the feature space [94]. Improvising scalability of the HF is also a direction for future pursuit [212]. Improving search specificity through a multi-view hashing approach can allow for the user to freely define a flexible search criterion, such as retrieval based on similarity with specific subparts of the neuron / image.

6.2 Metric Hashing Forests

In Chapter 3, we extend hashing forests into semantics-preserving hashing by introducing a novel supervised hashing method, termed as metric Hashing Forests (mHF). We work towards the objective of generating compact, similarity preserving binary codewords using metric Hashing Trees (mHTs). We introduced a principled node-optimization strategy by measuring the quality of neighborhood improvement (tailored for k-NN nearest neighbor search). We leveraged metric learning paradigms at local node level to project the data reaching each internal node into a latent subspace where classes have better separability. This ensured both class-separability and consistency within the hash codes generated by a mHT. Creating discriminative local subspace embeddings also helped bridge the gap between the low-level features and high-level semantic concepts that the hash bits have to preserve while generating encodings.

The proposed mHF comprises of ensemble of independently trained hashing trees, that hierarchically parse and encode the feature space such that local class neighborhoods are preserved and encode class similarity relationships in a highly compressed binary representation. Additionally, the asymptotic nature of forests and the independence between the hash functions helped achieve a steady improvement in hashing performance with inclusion of additional bits in the encoding.

The exposition of the proposed mHF framework is generic and we present the proof-of-concept for the challenging task of retrieval in neuroscientific image databases. Working on the neuromorphological feature space, we leveraged the mHF for classification by retrieval of neurons into seven cell types depending on their morphologies. We validated the proposed mHF algorithm on a large-scale, highly heterogeneous databases of 22,265 neurons curated from over 120 different archives, which are publicly available through neuromorpho.org [9]. The mHF framework is user-friendly as it is purely content-driven and query based, hence, does not require an expert to laboriously define the search criterion [228]. To the best of our knowledge, this is the first supervised hashing framework targeting neuroscientific image databases and we demonstrated superior retrieval and indexing performance over the state-of-the-art non-hashing and hashing based methods. By leveraging the concepts of hash tables and inverse lookup, we could achieve a time complexity independent of the database size, thus ensuring easy scalability of the mHF for time-efficient retrieval in large databases. We envisage that as the presented formulation is generic, it can be extended with success to other large-scale search and retrieval problems in medical imaging.

Towards the future work, we will look into retrieving neighbors that share similarity in local neuronal structures by introducing part-wise search specificity by providing the neuroscientists with the flexibility of selecting the volume of interest is a possible direction of future investigation. For further improvement in time efficiency, the mHF can be deployed in a parallel fashion. Introducing an on-line update variant like in Zhang et al. [223] can also help the mHF to evolve as the database scales up in size and heterogeneity. The current work focused on retrieval under constraints of anatomical similarities. Within neuroscientific databases, the extension of mHF to other classification schemes (like neurotransmitter based functional classification) remains an open topic for future investigation. In such cases, one can include associated multi-view features like taxonomical attributes (like species, gender, age *etc*.), experimental conditions (like protocol, staining, reconstruction method *etc*.), anatomical attributes (brain region) and other associated meta-data in tandem with neuromorphological features within the hashing scheme.

6.3 Hashing with Residual Networks

In Chapter 4, we have presented a novel deep learning based hashing approach leveraging upon residual learning, termed as Deep Residual Hashing (DRH). DRH integrates representation learning and hash coding into a joint optimization framework with dedicated losses for improving retrieval performance and hashing related losses to control the quantization error and improve the hash code quality. Our approach demonstrated very promising results on a challenging chest x ray dataset with co-occurring morbidities. In the future, from a clinical perspective, we would investigate into the visualizing what DRH learns *e.g.* if a particular image is classified as cardiomegaly, is DRH actually picking on clinical indicators around the heart and lungs rather as opposed to contextual clues from the background. We also plan to evaluate the transferability to datasets with unseen disease manifestations. It might also be interesting to extend DRH into a multi-modal / multi-view setting to jointly perform hashing by exploiting the relationship amongst these multiple modalities/views and for support specialized tasks such as cross-modal search.

From a technical perspective, a significant overhead is the hash code computation and use of efficient network designs such as binary convolutional networks [155] or alternatively training networks with low-precision weights [85] can be adopted. The training of DRH and many other state of the art deep hashing methods depends on pairwise similarity or variants of the same. Higher the sampling rate of such pairs during training, better the search accuracy at the cost of high training overhead. Such an approach can become prohibitively expensive when handling very large data and opens up a further questions of how to learn hash functions in a scalable fashion.

6.4 Robust Multiple Instance Hashing

In Chapter 5, for the first time, we proposed an end-to-end deep robust hashing framework, termed Robust Multiple Instance Hashing (RMIH), for retrieval under a multiple instance (MI) setting. We incorporate the multiple instance pooling (MIPool) layer to aggregate representations across instances to generate a bag-level discriminative hash code. We introduce the notion of robustness into our supervised retrieval loss inspired by neighborhood component analysis and improve the trainability of RMIH by utilizing an auxiliary single instance hashing arm regulated by a weight trade-off. Extensive validations and ablative testing on two public breast cancer datasets (mammography and histology) demonstrate the superiority of RMIH and its potential for future extension to other MI applications.

The inclusion of robustness within the training of deep hashing methods facilitates learning in presence of label noise. This implies that we can leverage potential sources of weak annotations such as biomedical crowd-sourcing [3], bounding box annotations or whole image level annotations (in place of fine-grained localization of pathology) [96] to learn these models. This is particularly useful in scenarios of limited availability of expert annotations. Incorporating semi-supervised learning paradigms such as [196] together with intelligently designed deep hashing models can be of high potential impact for medical image retrieval. In addition to learning with weak supervision, the incorporation of relevance feedback within retrieval could potentially supplement weak supervision. It is relatively inexpensive and highly effective if complex semantic concepts are not sufficiently incorporated in training data during supervision [152]. Approaches of hashing in combination with relevance feedback can also be effectively leveraged for interactive exploration of large-scale medical images such as whole-slide histopathology images (typically 1-2 Gbytes in size), starting with weakly explored queries and iteratively refining the search criterion with feedback.

6.5 Outlook

Retrieval in medical images is an open and active area of research. The availability of large-scale datasets, annotated training examples and scientific challenges has created immense opportunities to contribute within this field. Moving from visual information based CBIR to a more holistic multi-modal approach has much potential for creating real impact. Xu et al. [204] argue that fusion techniques that combine information from heterogeneous sources such as electronic health records with associated images will be helpful in exploring influences of patient related non-visual cues on specific disease change patterns in imaging information. It is likely that retrieval *per se* as an individual step is not sufficient for clinical practice, but rather integrating it with detection and classification would have practical impact. With increasing integration of multi-scale data such as genomics, proteomics and metabolomics *etc.* moving from a similar image retrieval to a *case retrieval* is needed. This implies integration of all available patient data to understand what *similarity* means in a specific scenario is needed.

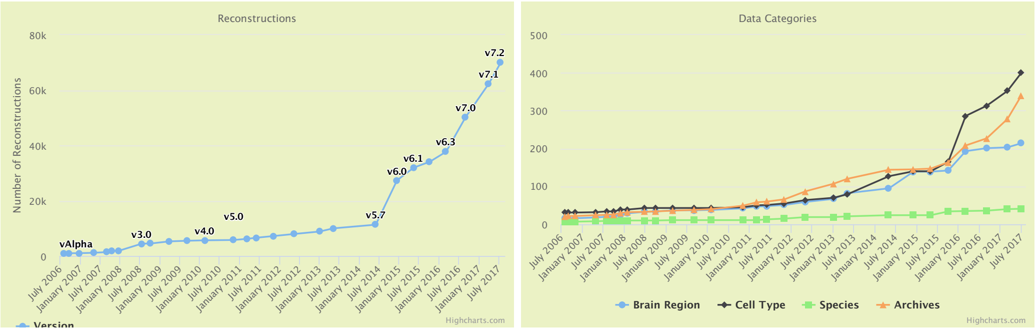
On the technical aspect of retrieval, methods that require weakly annotated data or even no annotations are still open to investigation. Active learning to reduce annotation effort, relevance feedback to incorporate supervision on the fly, and crowd-sourcing for annotations with quality control can be potentially leveraged in such scenarios. With mobile health technologies taking shape, adapting retrieval methods that will allow access to relevant clinical data and search function through light-weight mobile applications could create huge impact. Creating compute infrastructure within hospitals for medical research data analysis and retrieval would be a crucial step towards making digital medicine a reality and is a stepping stone for developing quality decision support tools with clinical impact.

Part IV

Appendix

A

Pattern Exploration in Neuroscientific Image Databases



(a) Total Number of Reconstructions

(b) Categories within Neuromorpho.Org

Fig. A.1. *Evolution of the number of neurons in the NeuroMorpho database*: In Fig. A.1a, amongst different released versions till October 2017, we can clearly see an exponential growth in the total number of neuronal reconstructions being made public. In addition to increasing number, as shown in Fig. A.1b we observe a similar trend in the heterogeneity within the database in terms of brain regions explored, cell types, species and archives.

Neurons facilitate exchange and processing of information and transmission of decisions within an organism and with their environment. They are typically tree-like structures with distinct morphological characteristics (in comparison to other body cells). The 3D morphometry of a neuron is of high interest to the neuroscience community as it is correlated to its physiology and functionality [44]. As a result, number of digitally reconstructed neuroscientific image databases has been rapidly increased over the past decade and become available publicly through web-based platforms like Neuromorpho (neuromorpho.org) [9], NeuronDepot [156], FlyCircuit (flycircuit.org) [29] to name a few. These repositories consist of heterogeneous multi-center data acquired from different species, brain regions, and experimental settings [9, 156] which allows for wider sharing of research data and outcomes and provide neuroscientists with an opportunity to study neuronal behavior and its morphological attributes across different species, brain regions, and experimental settings [9]. Fig. A.1 demonstrates the evolution of the number of neurons in one such popular public database (Neuromorpho.org). This has motivated researchers, particularly computer scientists, to develop new search systems for image retrieval and processing over large-scale datasets for the purpose of neuron categorization and ultimately comprehension of its functionality.

Digital reconstruction of neurons are digital representations of traced neuritic arborizations that encode information about the location, orientation and branching patterns as a set of interconnected cylinders (often using a tree-based data structure). These are typically acquired during the course of studies on neuron electrophysiology, pharmacology or histology and have are often used to derive morphological and stereological attributes about the neuron [9].

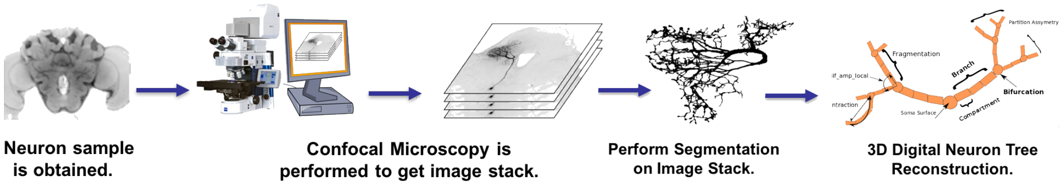

Neuron sample is obtained.
Confocal Microscopy is performed to get image stack.
Perform Segmentation on Image Stack.
3D Digital Neuron Tree Reconstruction.
Partition Assymetry
Fragmentation
Branch
Bifurcation
Soma Surface
Compartment

Fig. A.2. *Schematic of generating digital neuron reconstruction*: The neuron sample is imaged at high resolution using a confocal microscope to obtain an image stack. The image stack is then segmented either manually or using neuron-tracing tools to generate the digital reconstruction of the neuron.

Typically, neuron images are acquired using a myriad of microscopic imaging modalities like confocal light microscopy, laser scanning microscopy or electron microscopy [146]. Upon acquiring these images, neurons are either manually-traced (labor-intensive typically requiring 1-person week per neuron) or semi-automatically traced using neuron-tracing tools like Vaa3D [147] and saved as digital reconstructions. Fig. A.2 illustrates schematically the process of generating 3D digital reconstructions of neurons.

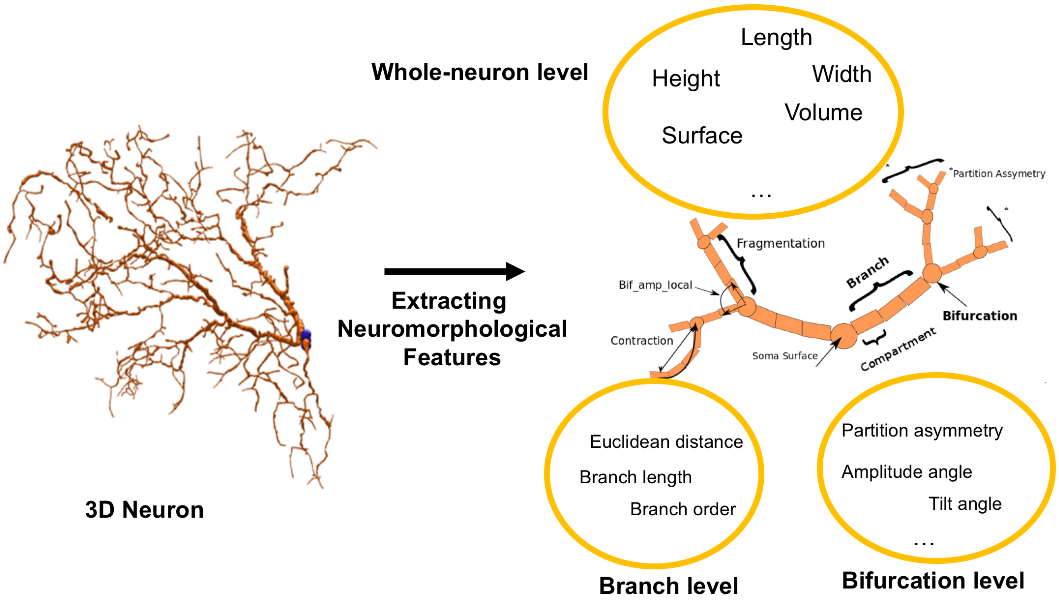


Fig. A.3. *Schematic of neuromorphological space*: The neuromorphological space is a high-dimensional feature representation encoding multiple aspects of neuron morphology. Using 3D reconstructions of neurons, features are extracted at the whole-neuron level such as height, length, volume *etc.*, at the branch-level including mean branch length, branch order *etc.* and at the neuronal bifurcation-level including partition assymetry, angles between children branches *etc.*

Today, more than ever, the increasing volume and growing diversity of neuroscientific data demand for computationally efficient methods for systematic neuron categorization and indexing. For example, within a particular species (say, mouse) the volume of a neuron could vary from $< 1\mu\text{m}^3$ for hippocampal neurons to $> 10^7\mu\text{m}^3$ for somatic neurons from the peripheral nervous system. Within a particular class of neurons (say granule cell), the neuronal volume could vary from approximately $300\mu\text{m}^3$ for a chimpanzee to $> 3000\mu\text{m}^3$ for a mouse. Neuron retrieval techniques can help researchers to compare and contrast the outcomes of their studies in terms of neuromorphology, organize the neurons in biologically meaningful clusters, and characterize and study neuronal networks [146, 192]. Query-

Search specificity	Features
Whole-neuron Level	arbor length, arbor height, arbor width, arbor depth, total volume, total number of tips, number of bifurcations, total surface, number of branches, diameter, soma surface, number of stems, contraction, fragmentation, Pk- classic + Branch level and Bifurcation level features.
Branch Level	average and max helicity, average and max fractal dimension, average and max branch path length, max branch order,average terminal degree, path distance, euclidean distance.
Bifurcation Level	average partition asymmetry, average and max local amplitude angle, average and max remote amplitude angle, average and max local tilt angle,average and max remote tilt angle, average and max local torque angle, average and max remote torque angle

Tab. A.1. *Quantitative morphological measurements*: This is the exhaustive features are extracted from 3D digital reconstructions of neurons that together constitute the neuromorphological space.

based retrieval of relevant neurons within databases can also be leveraged for comparative morphological analysis, which are used to study age related changes [157]. All these have motivated researchers to investigate and develop search, retrieval, and encoding algorithms upon large-scale neuron image datasets for purpose of neuron categorization, classification, and ultimately comprehension of its functionality. For example, Scorcioni et al. [166] extracted morphological descriptors to identify neuron structures with significant geometrical variations. Costa et al. [45] proposed a neuron search algorithm, where pairwise 3D structural alignment was employed to find similar neurons. In a similar approach, Ganglberger et al. [63] proposed a structure-based retrieval algorithm that compared local neural structures by capturing the orientation of neurons with a structure tensor field. During retrieval, this method compares the field generated for a query neuron across the cases in the target database and fetches neighbors with similar fields. In another approach, Polavaram et al. [149] focused on the evaluation of morphological similarities and dissimilarities between groups of neurons deploying unsupervised clustering technique using expert-labeled meta data (like species, brain region, and cell type). Recently, Wan et al. [192] designed a tool called BlastNeuron for comparing neurons in terms of their global appearance, detailed patterns of arborization, and topological similarities. Adopting a two stage retrieval approach, Wan et al. [192] compared neurons based on their global morphological features and then performed local alignment between pairs of retrieved neurons with a tree-based dynamic programming approach.

The concept of the neuromorphological space has been introduced by Costa et al. [44], where each neuron is represented by a set of morphological and topological measures. This feature space is used for multidimensional analysis of neuronal shape and showed that the cells of the same brain regions, types, or species tend to cluster together. This motivated us to leverage this space for evaluating inter-neuron similarity and propose a data-driven retrieval system for large neuron databases. We used the publicly-available Lmeasure software toolbox for extracting quantitative morphological measurements from digital 3D reconstructions of neurons [166]. These include features quantifying neurons at the level of the whole neuron, the branch level, and the bifurcation level. A sample neuron with some of its quantitative morphological measurements is illustrated in Fig. A.3. In total, we selected 37 features tabulated in Table A.1. A detailed description of each feature metric has been presented in [230, 166].

B Awards and Scholarships

- **TUM Internationalization Grant** for attending MICCAI 2017, Quebec City, Canada.

- **Runner-up - Young Scientist Award** at MICCAI 2017, Quebec City, Canada (12 out of over 800 submissions; *Co-author*) [160].

- **Student Travel Award** for attending MICCAI 2017, Quebec City, Canada (*Co-author*) [160].

- Awarded **Deutscher Akademischer Austauschdienst (DAAD) - Universities Australia** scholarship for research stay at School of Computer Science and Engineering, University of New South Wales, Australia in July 2017.

- **Best Paper Award in Medical Image Analysis** at MICCAI 2016, Athens, Greece (*First author*) [40].

- **Runner-up - Young Scientist Award** at MICCAI 2015, Munich, Germany (13 out of over 800 submissions; *First author*) [130].

- Awarded **Shri Gopal Rajgahria International Student Fellowship** for an exchange visit to Kharagpur Learning, Imaging and Visualization (KLIV) Group, Indian Institute of Technology Kharagpur, India in May - June 2016.

- Won **Synapse Prostate Cancer Dream Subchallenge 1b** for best performing machine learning models to improve the prediction of survival and toxicity of docetaxel treatment in patients with metastatic castrate resistant prostate cancer (mCRPC), August 2015 (*Team Member*) [150].

- Awarded **Institute Silver Medal** from Indian Institute of Technology Kharagpur, for best academic performance in Post Graduate Course in Medical Imaging and Informatics, July 2014.

- Awarded **Deutscher Akademischer Austauschdienst (DAAD) Scholarship** to pursue Master's Thesis at Chair for Computer Aided Medical Procedures & Augmented Reality, Fakultät für Informatik, Technische Universität München, Germany from September 2013 to March 2014.

- Awarded **Ministry of Human Resources and Development, Government of India Scholarship** for pursuing graduate studies in Medical Imaging and Informatics after qualifying Graduate Aptitude Test in Engineering. (Percentile: 99.81 % among 1,10,125 candidates)

- **Best Outgoing Student of the Year 2012** awarded by the Department of of Electrical and Electronics Engineering, BITS Pilani for Overall Excellence.

- Awarded **BITSAA IRU Research Travel Grant**, 2012 for attending the IEEE EMBS BHI 2012, Shenzhen, China.

- **Finalist amongst 192 papers** for the Best Student Paper Award at the IEEE EMBS BHI 2012, Shenzhen, China (*First author*) [35].

- Awarded **National Talent Search(NTS) Scholarship** by National Council for Education, Research and Training(NCERT) for scholastic excellence in 2007.

C List of Authored and Co-authored Publications

Primary Author

2017

[37] **Conjeti, Sailesh**, Magdalini Paschali, Amin Katouzian, and Nassir Navab. "Deep Multiple Instance Hashing for Scalable Medical Image Retrieval." *In International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 550-558. Springer, Cham, 2017.*

[39] **Conjeti, Sailesh**, Abhijit Guha Roy, Amin Katouzian, and Nassir Navab. "Hashing with residual networks for image retrieval." *In International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 541-549. Springer, Cham, 2017.*

[163] **Conjeti, Sailesh**, Abhijit Guha Roy, Sri Phani Krishna Karri, Debdoot Sheet, Amin Katouzian, Christian Wachinger, and Nassir Navab. "ReLayNet: Retinal Layer and Fluid Segmentation of Macular Optical Coherence Tomography using Fully Convolutional Network." *Biomedical Optics Express 8, no. 8 (2017): 3527-3642.*

[36] **Conjeti, Sailesh**, Anees Kazi, Nassir Navab, and Amin Katouzian. "Coupled Manifold Learning for Retrieval Across Modalities" *International Conference on Computer Vision Workshop on Manifold Learning: From Euclidean to Reimannian (2017).*

2016

[40] **Conjeti, Sailesh**, Amin Katouzian, Anees Kazi, Sepideh Mesbah, David Beymer, Tanveer F. Syeda-Mahmood, and Nassir Navab. "Metric hashing forests." *Medical image analysis 34 (2016): 13-29.*

[33] **Conjeti, Sailesh**, Abhijit Guha Roy, Debdoot Sheet, Stephane Carlier, Tanveer Syeda-Mahmood, Nassir Navab, and Amin Katouzian. "Domain Adapted Model for In Vivo Intravascular Ultrasound Tissue Characterization." *Computing and Visualization for Intravascular Imaging and Computer-Assisted Stenting (2016): 157.*

[42] **Conjeti, Sailesh**, Sepideh Mesbah, Mohammadreza Negahdar, Philipp L. Rautenberg, Shaoting Zhang, Nassir Navab, and Amin Katouzian. "Neuron-Miner: An Advanced Tool for Morphological Search and Retrieval in Neuroscientific Image Databases." *Neuroinformatics 14, no. 4 (2016): 369-385.*

[43] **Conjeti, Sailesh**, Amin Katouzian, Abhijit Guha Roy, Loïc Peter, Debdoot Sheet, Stéphane Carlier, Andrew Laine, and Nassir Navab. "Supervised domain adaptation of decision forests: Transfer of models trained in vitro for in vivo intravascular ultrasound tissue characterization." *Medical image analysis 32 (2016): 1-17.*

2015

[130] **Conjeti, Sailesh**, Sepideh Mesbah, Ajayrama Kumaraswamy, Philipp Rautenberg, Nassir Navab, and Amin Katouzian. "Hashing forests for morphological search and retrieval in neuroscientific image databases." *In International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 135-143. Springer, Cham, 2015.*

[41] **Conjeti, Sailesh**, Mehmet Yigitsoy, Debdoot Sheet, Jyotirmoy Chatterjee, Nassir Navab, and Amin Katouzian. "Mutually coherent structural representation for image registration through joint manifold embedding and alignment." *In Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on, pp. 601-604. IEEE, 2015.*

[38] **Conjeti, Sailesh**, Mehmet Yigitsoy, Tingying Peng, Debdoot Sheet, Jyotirmoy Chatterjee, Christine Bayer, Nassir Navab, and Amin Katouzian. "Deformable registration of immunofluorescence and histology using iterative cross-modal propagation." *In Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on, pp. 310-313. IEEE, 2015.*

2012

[34] **Conjeti, Sailesh**, and Dhiraj Puroshottam Jetwani. "Patient identification using high-confidence wavelet based Iris Pattern recognition." *In Biomedical and Health Informatics (BHI), 2012 IEEE-EMBS International Conference on, pp. 628-631. IEEE, 2012.*

[35] **Conjeti, Sailesh**, Rajiv Ranjan Singh, and Rahul Banerjee. "Bio-inspired wearable computing architecture and physiological signal processing for on-road stress monitoring." *In Biomedical and Health Informatics (BHI), 2012 IEEE-EMBS International Conference on, pp. 479-482. IEEE, 2012.*

Co-author

2017

[160] Roy, Abhijit Guha, **Sailesh Conjeti**, Debdoot Sheet, Amin Katouzian, Nassir Navab, and Christian Wachinger. "Error Corrective Boosting for Learning Fully Convolutional Networks with Limited Data." *In International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 231-239. Springer, Cham, 2017.*

2016

[150] Pölsterl, Sebastian, Pankaj Gupta, Lichao Wang, **Sailesh Conjeti**, Amin Katouzian, and Nassir Navab. "Heterogeneous ensembles for predicting survival of metastatic, castrate-resistant prostate cancer patients." *F1000Research 5 (2016)*.

[151] Pölsterl, Sebastian, **Sailesh Conjeti**, Nassir Navab, and Amin Katouzian. "Survival analysis for high-dimensional, heterogeneous medical data: Exploring feature extraction as an alternative to feature selection." *Artificial intelligence in medicine 72 (2016): 1-11*.

[8] Anura, Anji, **Sailesh Conjeti**, Raunak Kumar Das, Mousumi Pal, Ranjan Rashmi Paul, Swarnendu Bag, Ajoy Kumar Ray, and Jyotirmoy Chatterjee. "Computer-aided molecular pathology interpretation in exploring prospective markers for oral submucous fibrosis progression." *Head & neck 38, no. 5 (2016): 653-669*.

[114] Li, Zhongyu, Fumin Shen, Ruogu Fang, **Sailesh Conjeti**, Amin Katouzian, and Shaoting Zhang. "Maximum inner product search for morphological retrieval of large-scale neuron data." *In Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on, pp. 602-606. IEEE, 2016*.

[162] Roy, Abhijit Guha, **Sailesh Conjeti**, Stephane G. Carlier, Khalil Houissa, Andreas König, Pranab K. Dutta, Andrew F. Laine, Nassir Navab, Amin Katouzian, and Debdoot Sheet. "Multiscale distribution preserving autoencoders for plaque detection in intravascular optical coherence tomography." *In Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on, pp. 1359-1362. IEEE, 2016*.

[168] Shah, Amit, **Sailesh Conjeti**, Nassir Navab, and Amin Katouzian. "Deeply learnt hashing forests for content based image retrieval in prostate MR images." *In Medical Imaging: Image Processing, p. 978414. 2016*.

2015

[159] Roy, Abhijit Guha, **Sailesh Conjeti**, Stephane G. Carlier, Andreas König, Adnan Kastrati, Pranab K. Dutta, Andrew F. Laine, Nassir Navab, Debdoot Sheet, and Amin Katouzian. "Bag of forests for modelling of tissue energy interaction in optical coherence tomography for atherosclerotic plaque susceptibility assessment." *In Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on, pp. 428-431. IEEE, 2015*.

[161] Roy, Abhijit Guha, **Sailesh Conjeti**, Stéphane G. Carlier, Pranab K. Dutta, Adnan Kastrati, Andrew F. Laine, Nassir Navab, Amin Katouzian, and Debdoot Sheet. "Lumen segmentation in intravascular optical coherence tomography using backscattering tracked and initialized random walks." *IEEE journal of biomedical and health informatics 20, no. 2 (2016): 606-614*.

[7] Anura, Anji, **Sailesh Conjeti**, Mousumi Pal, Rashmi Ranjan Paul, and Jyotirmoy Chatterjee. "PO-121: Computer-aided quantitative interpretation to HIF-1a, c-MYC and p53 expression in oral submucous fibrosis." *Radiotherapy and Oncology 114 (2015): 59-60*.

[4] Albarqouni, Shadi, Maximilian Baust, **Sailesh Conjeti**, Asharf Al-Amoudi, and Nassir Navab. "Multi-scale Graph-based Guided Filter for De-noising Cryo-Electron Tomographic Data." *In British Machine Vision Conference, pp. 17-1. 2015*.

2014

[13] Barui, Ananya, Provas Banerjee, Amrita Chaudhary, **Sailesh Conjeti**, Bikash Mondal, Susmita Dey, and Jyotirmoy Chatterjee. "Evaluation of angiogenesis in diabetic lower limb wound healing using a natural medicine: A quantitative approach." *Wound Medicine 6 (2014): 26-33*.

[148] Peng, Tingying, Lichao Wang, Christine Bayer, **Sailesh Conjeti**, Maximilian Baust, and Nassir Navab. "Shading correction for whole slide image using low rank and sparse decomposition." *In International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 33-40. Springer, Cham, 2014*.

[102] Kraft, Silvan, **Sailesh Conjeti**, Peter B. Noël, Stéphane Carlier, Nassir Navab, and Amin Katouzian. "Full-wave intravascular ultrasound simulation from histology." *In International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 627-634. Springer, Cham, 2014*.

[178] Singh, Rajiv Ranjan, **Sailesh Conjeti**, and Rahul Banerjee. "Assessment of driver stress from physiological signals collected under real-time semi-urban driving scenarios." *International Journal of Computational Intelligence Systems 7, no. 5 (2014): 909-923*.

2013

[176] Singh, Rajiv Ranjan, **Sailesh Conjeti**, and Rahul Banerjee. "A comparative evaluation of neural network classifiers for stress level analysis of automotive drivers using physiological signals." *Biomedical Signal Processing and Control 8, no. 6 (2013): 740-754*. Harvard

[171] Sheet, Debdoot, Sri Phani Krishna Karri, **Sailesh Conjeti**, Sambuddha Ghosh, Jyotirmoy Chatterjee, and Ajoy Kumar Ray. "Detection of retinal vessels in fundus images through transfer learning of tissue specific photon interaction statistical physics." *In Biomedical Imaging (ISBI), 2013 IEEE 10th International Symposium on, pp. 1452-1456. IEEE, 2013*.

[12] Bag, Swarnendu, **Sailesh Conjeti**, Raunak Kumar Das, Mousami Pal, Anji Anura, Ranjan Rashmi Paul, Ajoy Kumar Ray, Sanghamitra Sengupta, and Jyotirmoy Chatterjee. "Computational analysis of p63+ nuclei distribution pattern by graph theoretic approach in an oral pre-cancer (sub-mucous fibrosis)." *Journal of pathology informatics 4 (2013)*.

2012

[179] Singh, Rajiv Ranjan, **Sailesh Conjeti**, and Rahul Banerjee. "Biosignal based on-road stress monitoring for automotive drivers." *In Communications (NCC), 2012 National Conference on, pp. 1-5. IEEE, 2012.*

2011

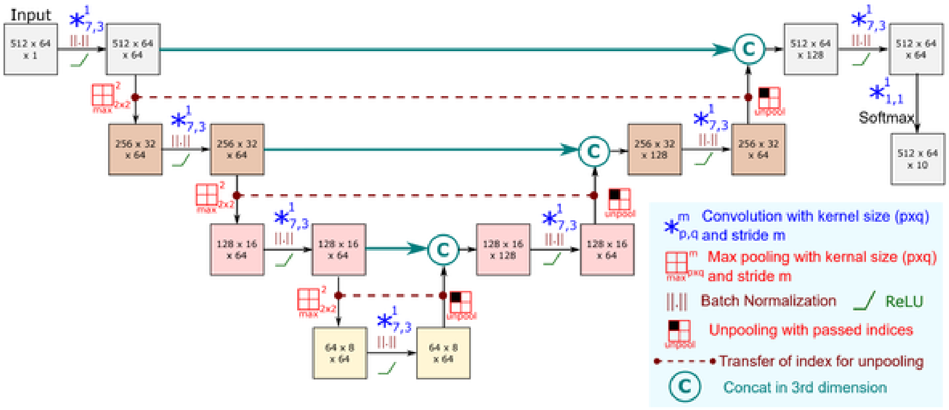
[177] Singh, Rajiv Ranjan, **Sailesh Conjeti**, and Rahul Banerjee. "An approach for real-time stress-trend detection using physiological signals in wearable computing systems for automotive drivers." *In Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on, pp. 1477-1482. IEEE, 2011.*

D

Abstracts of Publications not Discussed in this Thesis

ReLayNet: Retinal Layer and Fluid Segmentation of Macular Optical Coherence Tomography using Fully Convolutional Network

Abhijit Guha Roy, Sailesh Conjeti, Sri Phani Krishna Karri, Debdoot Sheet, Amin Katouzian, Christian Wachinger, Nassir Navab



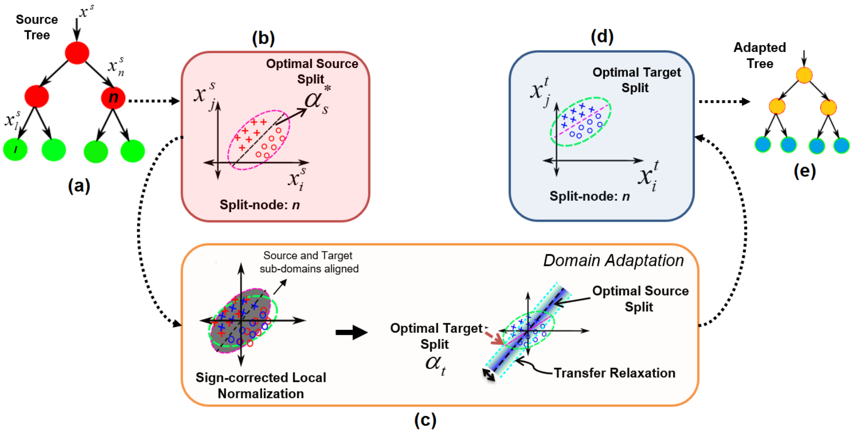
Proposed fully convolutional ReLayNet architecture. The spatial resolution of the feature maps are indicated in the boxes. The underlying layer symbols are indicated to the right.

Optical coherence tomography (OCT) is used for non-invasive diagnosis of diabetic macular edema assessing the retinal layers. In this paper, we propose a new fully convolutional deep architecture, termed ReLayNet, for end-to-end segmentation of retinal layers and fluid masses in eye OCT scans. ReLayNet uses a contracting path of convolutional blocks (encoders) to learn a hierarchy of contextual features, followed by an expansive path of convolutional blocks (decoders) for semantic segmentation. ReLayNet is trained to optimize a joint loss function comprising of weighted logistic regression and Dice overlap loss. The framework is validated on a publicly available benchmark dataset with comparisons against five state-of-the-art segmentation methods including two deep learning based approaches to substantiate its effectiveness.

Biomedical Optics Express (2017)

Supervised domain adaptation of decision forests: Transfer of models trained in vitro for in vivo intravascular ultrasound tissue characterization

Sailesh Conjeti, Amin Katouzian, Abhijit Guha Roy, Loïc Peter, Debdoot Sheet, Stéphane Carlier, Andrew Laine, Nassir Navab



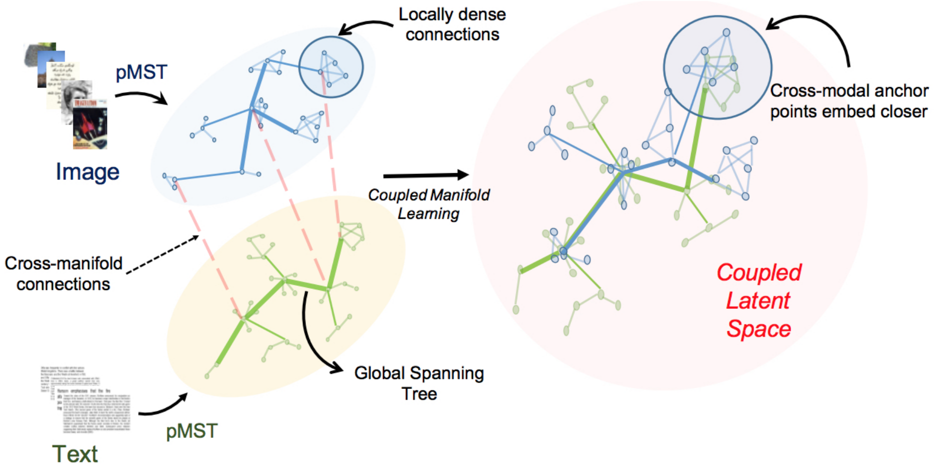
Overview of the proposed method for domain adaptation of *in vitro* trained random forests for *in vivo* intra-vascular ultrasound tissue characterization.

In this paper, we propose a supervised domain adaptation (DA) framework for adapting decision forests in the presence of distribution shift between training (source) and testing (target) domains, given few labeled examples. We introduce a novel method for DA through an error-correcting hierarchical transfer relaxation scheme with domain alignment, feature normalization, and leaf posterior reweighting to correct for the distribution shift between the domains. For the first time we apply DA to the challenging problem of extending in vitro trained forests (source domain) for in vivo applications (target domain). The proof-of-concept is provided for in vivo characterization of atherosclerotic tissues using intravascular ultrasound signals, where presence of flowing blood is a source of distribution shift between the two domains. This potentially leads to misclassification upon direct deployment of in vitro trained classifier, thus motivating the need for DA as obtaining reliable in vivo training labels is often challenging if not infeasible. Exhaustive validations and parameter sensitivity analysis substantiate the reliability of the proposed DA framework and demonstrates improved tissue characterization performance for scenarios where adaptation is conducted in presence of only a few examples. The proposed method can thus be leveraged to reduce annotation costs and improve computational efficiency over conventional retraining approaches.

Medical Image Analysis (2016)

Coupled Manifold Learning for Retrieval Across Modalities

Sailesh Conjeti, Anees Kazi, Nassir Navab, Amin Katouzian


Locally dense connections
pMST
Image
Cross-modal anchor points embed closer
Coupled Manifold Learning
Cross-manifold connections
Coupled Latent Space
Global Spanning Tree
pMST
Text

Schematic of proposed cross-modal search retrieval scheme using Coupled Manifold Learning (CpML). Given modalities (image and text) and limited co-occurring instances, we model the intra-modal proximity with pMST which creates locally dense connections with a global spanning tree representing the underlying data manifolds global topology. Next, we leverage correspondences through CpML, we learn to map to the coupled latent space that is makes the modalities metric-comparable.

Coupled Manifold Learning (CpML) is targeted at aligning data manifolds across two related modalities to facilitate similarity preserving cross-modal retrieval. Local and global topologies of the data cloud reflect intra-class variability and overall heterogeneity respectively making it critical to retain both for meaningful retrieval. Towards this we propose a learning paradigm which simultaneously aligns global topology while preserving local manifold structure. The global topologies are maintained by recovering underlying mapping functions in the joint manifold space by deploying partially corresponding instances. The inter-, and intra-modality affinity matrices are then computed to reinforce original data skeleton using perturbed minimum spanning tree (pMST), and maximizing the affinity among similar cross-modal instances, respectively. The performance of proposed algorithm is evaluated upon two benchmark multi-modal image-text datasets (Wikipedia and PascalVOC2012 - Sentence). We further show versatility and interdisciplinary application by extending it to cross-modal retrieval between multi-stain atherosclerosis histology medical image dataset. We exhaustively validate and compare CpML to other joint-manifold learning methods and demonstrate superior performance across datasets and tasks.

International Conference on Computer Vision Workshop on Manifold Learning: From Euclidean to Reimannian (2017)

Error Corrective Boosting for Learning Fully Convolutional Networks with Limited Data

Abhijit Guha Roy, Sailesh Conjeti, Debdoot Sheet, Amin Katouzian, Nassir Navab, Christian Wachinger

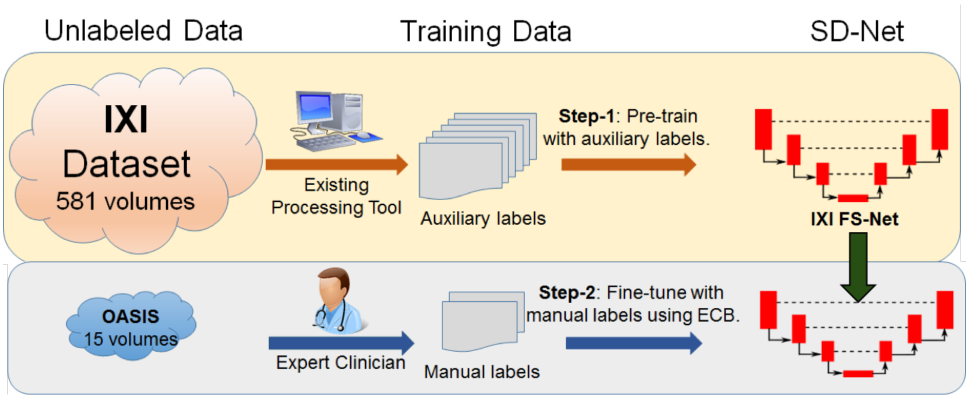


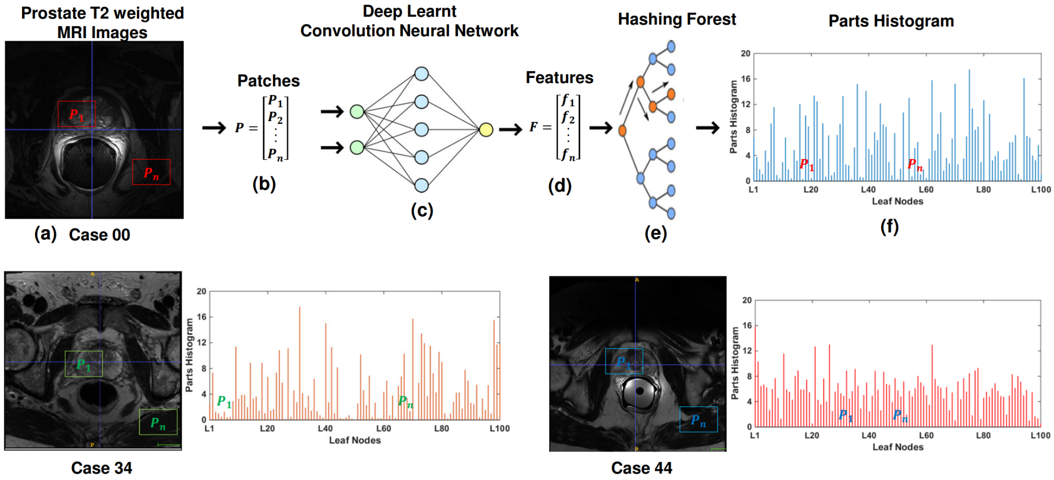
Illustration of the different steps involved in training of fully convolutional CNNs with surplus auxiliary labeled data and limited manually labeled data.

Training deep fully convolutional neural networks (F-CNNs) for semantic image segmentation requires access to abundant labeled data. While large datasets of unlabeled image data are available in medical applications, access to manually labeled data is very limited. We propose to automatically create auxiliary labels on initially unlabeled data with existing tools and to use them for pre-training. For the subsequent fine-tuning of the network with manually labeled data, we introduce error corrective boosting (ECB), which emphasizes parameter updates on classes with lower accuracy. Furthermore, we introduce SkipDeconv-Net (SD-Net), a new F-CNN architecture for brain segmentation that combines skip connections with the unpooling strategy for upsampling. The SD-Net addresses challenges of severe class imbalance and errors along boundaries. With application to whole-brain MRI T1 scan segmentation, we generate auxiliary labels on a large dataset with FreeSurfer and fine-tune on two datasets with manual annotations. Our results show that the inclusion of auxiliary labels and ECB yields significant improvements. SD-Net segments a 3D scan in 7 secs in comparison to 30 hours for the closest multi-atlas segmentation method, while reaching similar performance. It also outperforms the latest state-of-the-art F-CNN models.

International Conference on Medical Image Computing and Computer-Assisted Intervention 2017

Deeply Learnt Hashing Forests for Content Based Image Retrieval in Prostate MR Images

Amit Shah, Sailesh Conjeti, Nassir Navab, Amin Katouzian


Prostate T2 weighted MRI Images
(a) Case 00
Patches
$P = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_n \end{bmatrix}$
(b)
Deep Learnt Convolution Neural Network
(c)
Features
$F = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}$
(d)
Hashing Forest
(e)
Parts Histogram
(f)
Parts Histogram
Leaf Nodes
L1
L20
L40
L60
L80
L100
P_1
P_n
Case 34
Case 44

Overview of CBIR framework for Prostate MR images using Deep Learnt Hashing Forests.

Deluge in the size and heterogeneity of medical image databases necessitates the need for content based retrieval systems for their efficient organization. In this paper, we propose such a system to retrieve prostate MR images which share similarities in appearance and content with a query image. We introduce deeply learnt hashing forests (DL-HF) for this image retrieval task. DL-HF effectively leverages the semantic descriptiveness of deep learnt Convolutional Neural Networks. This is used in conjunction with hashing forests which are unsupervised random forests. DL-HF hierarchically parses the deep-learnt feature space to encode subspaces with compact binary code words. We propose a similarity preserving feature descriptor called Parts Histogram which is derived from DL-HF. Correlation defined on this descriptor is used as a similarity metric for retrieval from the database. Validations on publicly available multi-center prostate MR image database established the validity of the proposed approach. The proposed method is fully-automated without any user-interaction and is not dependent on any external image standardization like image normalization and registration. This image retrieval method is generalizable and is well-suited for retrieval in heterogeneous databases other imaging modalities and anatomies.

SPIE Medical Imaging (2016)

Bibliography

[1] M. Agarwal and J. Mostafa. "Content-based image retrieval for Alzheimer's disease detection". In: *Content-Based Multimedia Indexing (CBMI), 2011 9th International Workshop on*. IEEE. 2011, pp. 13–18 (cit. on p. 26).

[2] A. Albalate and D. Suendermann. "A combination approach to cluster validation based on statistical quantiles". In: *Bioinformatics, Systems Biology and Intelligent Computing, 2009. IJCBS'09. International Joint Conference on*. IEEE. 2009, pp. 549–555 (cit. on p. 35).

[3] S. Albarqouni, C. Baur, F. Achilles, V. Belagiannis, S. Demirci, and N. Navab. "Aggnet: deep learning from crowds for mitosis detection in breast cancer histology images". In: *IEEE transactions on medical imaging* 35.5 (2016), pp. 1313–1321 (cit. on p. 124).

[4] S. Albarqouni, M. Baust, S. Conjeti, A. Al-Amoudi, and N. Navab. "Multi-scale Graph-based Guided Filter for De-noising Cryo-Electron Tomographic Data." In: *BMVC*. 2015, pp. 17–1 (cit. on p. 138).

[5] B. André, T. Vercauteren, A. M. Buchner, M. B. Wallace, and N. Ayache. "Learning semantic and visual similarity for endomicroscopy video retrieval". In: *IEEE Transactions on Medical Imaging* 31.6 (2012), pp. 1276–1288 (cit. on p. 25).

[6] S. K. Antani, T. M. Deserno, L. R. Long, M. O. Güld, L. Neve, and G. R. Thoma. "Interfacing global and local CBIR systems for medical image retrieval". In: *Bildverarbeitung für die Medizin 2007* (2007), pp. 166–171 (cit. on p. 26).

[7] A Anura, S Conjeti, M Pal, R. Paul, and J Chatterjee. "PO-121: Computer-aided quantitative interpretation to HIF-1a, c-MYC and p53 expression in oral submucous fibrosis". In: *Radiotherapy and Oncology* 114 (2015), pp. 59–60 (cit. on p. 137).

[8] A. Anura, S. Conjeti, R. K. Das, et al. "Computer-aided molecular pathology interpretation in exploring prospective markers for oral submucous fibrosis progression". In: *Head & neck* 38.5 (2016), pp. 653–669 (cit. on p. 137).

[9] G. A. Ascoli, D. E. Donohue, and M. Halavi. "NeuroMorpho. Org: a central resource for neuronal morphologies". In: *Journal of Neuroscience* 27.35 (2007), pp. 9247–9251 (cit. on pp. 39, 62, 122, 129).

[10] U. Avni, H. Greenspan, E. Konen, M. Sharon, and J. Goldberger. "X-ray categorization and retrieval on the organ and pathology level, using patch-based visual words". In: *IEEE Transactions on Medical Imaging* 30.3 (2011), pp. 733–746 (cit. on p. 76).

[11] R. Baeza-Yates and G. Valiente. "An image similarity measure based on graph matching". In: *String Processing and Information Retrieval, 2000. SPIRE 2000. Proceedings. Seventh International Symposium on*. IEEE. 2000, pp. 28–38 (cit. on p. 11).

[12] S. Bag, S. Conjeti, R. K. Das, et al. "Computational analysis of p63+ nuclei distribution pattern by graph theoretic approach in an oral pre-cancer (sub-mucous fibrosis)". In: *Journal of pathology informatics* 4 (2013) (cit. on p. 138).

[13] A. Barui, P. Banerjee, A. Chaudhary, et al. “Evaluation of angiogenesis in diabetic lower limb wound healing using a natural medicine: A quantitative approach”. In: *Wound Medicine* 6 (2014), pp. 26–33 (cit. on p. 138).

[14] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. “Speeded-up robust features (SURF)”. In: *Computer vision and image understanding* 110.3 (2008), pp. 346–359 (cit. on p. 11).

[15] A. Bellet, A. Habrard, and M. Sebban. “A survey on metric learning for feature vectors and structured data”. In: *arXiv preprint arXiv:1306.6709* (2013) (cit. on pp. 54, 62).

[16] R. Bellman. *Dynamic programming*. Courier Corporation, 2013 (cit. on p. 4).

[17] J. L. Bentley. “Multidimensional binary search trees used for associative searching”. In: *Communications of the ACM* 18.9 (1975), pp. 509–517 (cit. on p. 33).

[18] A. Beygelzimer, S. Kakade, and J. Langford. “Cover trees for nearest neighbor”. In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 97–104 (cit. on p. 4).

[19] L. Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32 (cit. on p. 55).

[20] K. Bunte, M. Biehl, M. F. Jonkman, and N. Petkov. “Learning effective color features for content based image retrieval in dermatology”. In: *Pattern Recognition* 44.9 (2011), pp. 1892–1902 (cit. on p. 24).

[21] D. Cai, X. He, K. Zhou, J. Han, and H. Bao. “Locality Sensitive Discriminant Analysis.” In: *IJCAI*. Vol. 2007. 2007, pp. 1713–1726 (cit. on pp. 54, 65).

[22] W. Cai, S. Liu, L. Wen, S. Eberl, M. J. Fulham, and D. Feng. “3D neurological image retrieval with localized pathology-centric CMRGlc patterns”. In: *Image Processing (ICIP), 2010 17th IEEE International Conference on*. IEEE. 2010, pp. 3201–3204 (cit. on p. 25).

[23] J. C. Caicedo, F. A. González, and E. Romero. “Content-based histopathology image retrieval using a kernel-based semantic annotation framework”. In: *Journal of biomedical informatics* 44.4 (2011), pp. 519–528 (cit. on p. 24).

[24] J. C. Caicedo, F. A. Gonzalez, and E. Romero. “Content-Based Medical Image Retrieval Using Low-Level Visual Features and Modality Identification.” In: *CLEF*. Springer. 2007, pp. 615–622 (cit. on p. 25).

[25] Y. Cao, S. Steffey, J. He, et al. “Medical image retrieval: a multimodal approach”. In: *Cancer informatics* 13.Suppl 3 (2014), p. 125 (cit. on p. 26).

[26] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. “Return of the devil in the details: Delving deep into convolutional nets”. In: *arXiv preprint arXiv:1405.3531* (2014) (cit. on pp. 13, 80, 87, 88, 92, 102, 109, 114).

[27] D. Chen, X. Cao, F. Wen, and J. Sun. “Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification”. In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE. 2013, pp. 3025–3032 (cit. on p. 4).

[28] Y. Chen, J. E. Argentinis, and G. Weber. “IBM Watson: how cognitive computing can be applied to big data challenges in life sciences research”. In: *Clinical therapeutics* 38.4 (2016), pp. 688–701 (cit. on p. 27).

[29] A.-S. Chiang, C.-Y. Lin, C.-C. Chuang, et al. “Three-dimensional reconstruction of brain-wide wiring networks in Drosophila at single-cell resolution”. In: *Current Biology* 21.1 (2011), pp. 1–11 (cit. on p. 129).

[30] K. Clark, B. Vendt, K. Smith, et al. “The Cancer Imaging Archive (TCIA): maintaining and operating a public information repository”. In: *Journal of digital imaging* 26.6 (2013), pp. 1045–1057 (cit. on p. 11).

[31] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. “Fast and accurate deep network learning by exponential linear units (elus)”. In: *arXiv preprint arXiv:1511.07289* (2015) (cit. on p. 12).

[32] D. Comaniciu, P. Meer, and D. J. Foran. "Image-guided decision support system for pathology". In: *Machine Vision and Applications* 11.4 (1999), pp. 213–224 (cit. on p. 24).

[33] S Conjeti, A. Roy, D Sheet, et al. "Domain Adapted Model for In Vivo Intravascular Ultrasound Tissue Characterization". In: *Computing and Visualization for Intravascular Imaging and Computer-Assisted Stenting* (2016), p. 157 (cit. on p. 135).

[34] S. Conjeti and D. P. Jetwani. "Patient identification using high-confidence wavelet based Iris Pattern recognition". In: *Biomedical and Health Informatics (BHI), 2012 IEEE-EMBS International Conference on*. IEEE. 2012, pp. 628–631 (cit. on p. 136).

[35] S. Conjeti, R. R. Singh, and R. Banerjee. "Bio-inspired wearable computing architecture and physiological signal processing for on-road stress monitoring". In: *Biomedical and Health Informatics (BHI), 2012 IEEE-EMBS International Conference on*. IEEE. 2012, pp. 479–482 (cit. on pp. 134, 136).

[36] S. Conjeti, A. Kazi, N. Navab, and A. Katouzian. "Cross-Modal Manifold Learning for Cross-modal Retrieval". In: *arXiv preprint arXiv:1612.06098* (2016) (cit. on p. 135).

[37] S. Conjeti, M. Paschali, A. Katouzian, and N. Navab. "Deep Multiple Instance Hashing for Scalable Medical Image Retrieval". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2017, pp. 550–558 (cit. on pp. 19, 98, 101, 102, 108, 111–117, 135).

[38] S. Conjeti, M. Yigitsoy, T. Peng, et al. "Deformable registration of immunofluorescence and histology using iterative cross-modal propagation". In: *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*. IEEE. 2015, pp. 310–313 (cit. on p. 136).

[39] S. Conjeti, A. G. Roy, A. Katouzian, and N. Navab. "Hashing with residual networks for image retrieval". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2017, pp. 541–549 (cit. on pp. 19, 76–78, 90–95, 99, 135).

[40] S. Conjeti, A. Katouzian, A. Kazi, et al. "Metric hashing forests". In: *Medical image analysis* 34 (2016), pp. 13–29 (cit. on pp. 17, 52, 55–57, 59, 61, 63, 65–70, 87, 92, 133, 135).

[41] S. Conjeti, M. Yigitsoy, D. Sheet, J. Chatterjee, N. Navab, and A. Katouzian. "Mutually coherent structural representation for image registration through joint manifold embedding and alignment". In: *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*. IEEE. 2015, pp. 601–604 (cit. on p. 136).

[42] S. Conjeti, S. Mesbah, M. Negahdar, et al. "Neuron-Miner: An Advanced Tool for Morphological Search and Retrieval in Neuroscientific Image Databases". In: *Neuroinformatics* 14.4 (2016), pp. 369–385 (cit. on pp. 17, 32, 36, 37, 39, 42–47, 49, 135).

[43] S. Conjeti, A. Katouzian, A. G. Roy, et al. "Supervised domain adaptation of decision forests: Transfer of models trained in vitro for in vivo intravascular ultrasound tissue characterization". In: *Medical image analysis* 32 (2016), pp. 1–17 (cit. on p. 136).

[44] L. D. F. Costa, K. Zawadzki, M. Miazaki, M. P. Viana, and S. N. Taraskin. "Unveiling the neuromorphological space". In: *Frontiers in computational neuroscience* 4 (2010) (cit. on pp. 39, 62, 129, 131).

[45] M. Costa, J. D. Manton, A. D. Ostrovsky, S. Prohaska, and G. S. Jefferis. "NBLAST: Rapid, sensitive comparison of neuronal structure and construction of neuron family databases". In: *Neuron* 91.2 (2016), pp. 293–311 (cit. on p. 131).

[46] A. Criminisi, J. Shotton, E. Konukoglu, et al. "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning". In: *Foundations and Trends® in Computer Graphics and Vision* 7.2–3 (2012), pp. 81–227 (cit. on pp. 55, 59).

[47] A. Del Bimbo and P. Pala. "Visual image retrieval by elastic matching of user sketches". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.2 (1997), pp. 121–132 (cit. on p. 11).

[48] D. Demner-Fushman, M. D. Kohli, M. B. Rosenman, et al. “Preparing a collection of radiology examinations for distribution and retrieval”. In: *Journal of the American Medical Informatics Association* 23.2 (2015), pp. 304–310 (cit. on p. 86).

[49] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 248–255 (cit. on pp. 3, 11, 13).

[50] A. Depeursinge and H. Müller. “Fusion techniques for combining textual and visual information retrieval”. In: *ImageCLEF*. Springer, 2010, pp. 95–114 (cit. on p. 26).

[51] B. Desgraupes. “Clustering indices”. In: *University of Paris Ouest-Lab Modal'X* 1 (2013), p. 34 (cit. on p. 35).

[52] K. Doi. “Computer-aided diagnosis in medical imaging: historical review, current status and future potential”. In: *Computerized medical imaging and graphics* 31.4 (2007), pp. 198–211 (cit. on p. 22).

[53] A. Dosovitskiy and T. Brox. “Generating images with perceptual similarity metrics based on deep networks”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 658–666 (cit. on p. 13).

[54] S. Doyle, M. Hwang, S. Naik, M. Feldman, J. Tomaszeweski, and A. Madabhushi. “Using manifold learning for content-based image retrieval of prostate histopathology”. In: *MICCAI 2007 Workshop on Content-based Image Retrieval for Biomedical Image Archives: Achievements, Problems, and Prospects*. 2007, pp. 53–62 (cit. on p. 24).

[55] W. Duch and N. Jankowski. “Survey of neural transfer functions”. In: *Neural Computing Surveys* 2.1 (1999), pp. 163–212 (cit. on p. 11).

[56] L. Duijm, M. Louwman, J. Groenewoud, L. Van De Poll-Franse, J. Fracheboud, and J. W. Coebergh. “Inter-observer variability in mammography screening and effect of type and number of readers on screening outcome”. In: *British journal of cancer* 100.6 (2009), p. 901 (cit. on pp. 99, 104).

[57] M. M. Dundar, S. Badve, G. Bilgin, et al. “Computerized classification of intraductal breast lesions using histopathological images”. In: *IEEE Transactions on Biomedical Engineering* 58.7 (2011), pp. 1977–1984 (cit. on p. 109).

[58] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. “Deep hashing for compact binary codes learning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2475–2483 (cit. on p. 19).

[59] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. “The pascal visual object classes (voc) challenge”. In: *International journal of computer vision* 88.2 (2010), pp. 303–338 (cit. on pp. 3, 11).

[60] A. Folkers and H. Samet. “Content-based image retrieval using Fourier descriptors on a logo database”. In: *Pattern Recognition, 2002. Proceedings. 16th International Conference on*. Vol. 3. IEEE. 2002, pp. 521–524 (cit. on p. 11).

[61] D. J. Foran, L. Yang, W. Chen, et al. “Imageminer: a software system for comparative analysis of tissue microarrays using content-based image retrieval, high-performance computing, and grid technology”. In: *Journal of the American Medical Informatics Association* 18.4 (2011), pp. 403–415 (cit. on p. 26).

[62] K. Ganesan, U. R. Acharya, C. K. Chua, L. C. Min, K. T. Abraham, and K.-H. Ng. “Computer-aided breast cancer detection using mammograms: a review”. In: *IEEE Reviews in biomedical engineering* 6 (2013), pp. 77–98 (cit. on p. 98).

[63] F. Ganglberger, F. Schulze, L. Tirian, et al. “Structure-based neuron retrieval across Drosophila brains”. In: *Neuroinformatics* 12.3 (2014), pp. 423–434 (cit. on p. 131).

[64] A. Gionis, P. Indyk, R. Motwani, et al. "Similarity search in high dimensions via hashing". In: *VLDB*. Vol. 99. 6. 1999, pp. 518–529 (cit. on pp. 15, 66).

[65] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov. "Neighbourhood components analysis". In: (2005), pp. 513–520 (cit. on pp. 19, 54, 80, 84).

[66] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.12 (2013), pp. 2916–2929 (cit. on pp. 8, 14, 32, 82, 87, 92, 110, 117).

[67] A. Gordo, F. Perronnin, Y. Gong, and S. Lazebnik. "Asymmetric distances for binary embeddings". In: *IEEE transactions on pattern analysis and machine intelligence* 36.1 (2014), pp. 33–47 (cit. on p. 8).

[68] M. O. Güld, C. Thies, B. Fischer, and T. M. Lehmann. "Content-based retrieval of medical images by combining global features". In: *Workshop of the Cross-Language Evaluation Forum for European Languages*. Springer. 2005, pp. 702–711 (cit. on p. 25).

[69] S. Haas, R. Donner, A. Burner, M. Holzer, and G. Langs. "Superpixel-based interest points for effective bags of visual words medical image retrieval". In: *MICCAI International Workshop on Medical Content-Based Retrieval for Clinical Decision Support*. Springer. 2011, pp. 58–68 (cit. on p. 24).

[70] T. Hampton. "Cancer genome atlas". In: *JAMA* 296.16 (2006), pp. 1958–1958 (cit. on p. 11).

[71] A. Hanbury, C. Boyer, M. Gschwandtner, and H. Müller. "KHRESMOI: towards a multi-lingual search and access system for biomedical information". In: *Med-e-Tel, Luxembourg* 2011 (2011), pp. 412–416 (cit. on p. 27).

[72] J. He, W. Liu, and S.-F. Chang. "Scalable similarity search with optimized kernel hashing". In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2010, pp. 1129–1138 (cit. on p. 9).

[73] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (cit. on pp. 13, 79, 80, 102, 103, 109, 114).

[74] K. He, X. Zhang, S. Ren, and J. Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034 (cit. on p. 12).

[75] X. He and P. Niyogi. "Locality preserving projections". In: *Advances in neural information processing systems*. 2004, pp. 153–160 (cit. on pp. 54, 57, 58, 65).

[76] X. He, D. Cai, S. Yan, and H.-J. Zhang. "Neighborhood preserving embedding". In: *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. Vol. 2. IEEE. 2005, pp. 1208–1213 (cit. on pp. 41, 54, 65).

[77] M. Heath, K. Bowyer, D. Kopans, et al. "Current status of the digital database for screening mammography". In: *Digital mammography*. Springer, 1998, pp. 457–460 (cit. on p. 109).

[78] M. Heath, K. Bowyer, D. Kopans, R. Moore, and W. P. Kegelmeyer. "The digital database for screening mammography". In: *Proceedings of the 5th international workshop on digital mammography*. Medical Physics Publishing. 2000, pp. 212–218 (cit. on p. 11).

[79] R. Hecht-Nielsen et al. "Theory of the backpropagation neural network." In: *Neural Networks* 1.Supplement-1 (1988), pp. 445–448 (cit. on p. 12).

[80] G Hinton, N. Srivastava, and K. Swersky. "Rmsprop: Divide the gradient by a running average of its recent magnitude". In: *Neural networks for machine learning, Coursera lecture 6e* (2012) (cit. on p. 13).

[81] A. Holt, I. Bichindaritz, R. Schmidt, and P. Perner. “Medical applications in case-based reasoning”. In: *The Knowledge Engineering Review* 20.3 (2005), pp. 289–292 (cit. on p. 22).

[82] P. Howarth and S. M. Rüger. “Evaluation of texture features for content-based image retrieval”. In: *CIVR*. Vol. 3115. Springer. 2004, pp. 326–334 (cit. on p. 11).

[83] R. Hu, M. Barnard, and J. Collomosse. “Gradient field descriptor for sketch based retrieval and localization”. In: *Image Processing (ICIP), 2010 17th IEEE International Conference on*. IEEE. 2010, pp. 1025–1028 (cit. on p. 11).

[84] H. Huang. *PACS and imaging informatics: basic principles and applications*. John Wiley & Sons, 2011 (cit. on p. 22).

[85] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. “Quantized neural networks: Training neural networks with low precision weights and activations”. In: *arXiv preprint arXiv:1609.07061* (2016) (cit. on p. 123).

[86] P. J. Huber. “Robust statistics”. In: *International Encyclopedia of Statistical Science*. Springer, 2011, pp. 1248–1251 (cit. on p. 106).

[87] S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International Conference on Machine Learning*. 2015, pp. 448–456 (cit. on pp. 12, 80).

[88] H. V. Jagadish, B. C. Ooi, K.-L. Tan, C. Yu, and R. Zhang. “iDistance: An adaptive B+-tree based indexing method for nearest neighbor search”. In: *ACM Transactions on Database Systems (TODS)* 30.2 (2005), pp. 364–397 (cit. on p. 3).

[89] A. Jalalian, S. B. Mashohor, H. R. Mahmud, M. I. B. Saripan, A. R. B. Ramli, and B. Karasfi. “Computer-aided detection/diagnosis of breast cancer in mammography and ultrasound: a review”. In: *Clinical imaging* 37.3 (2013), pp. 420–426 (cit. on p. 98).

[90] H. Jegou, M. Douze, and C. Schmid. “Product quantization for nearest neighbor search”. In: *IEEE transactions on pattern analysis and machine intelligence* 33.1 (2011), pp. 117–128 (cit. on p. 3).

[91] A. Jemal, M. M. Center, C. DeSantis, and E. M. Ward. “Global patterns of cancer incidence and mortality rates and trends”. In: *Cancer Epidemiology and Prevention Biomarkers* 19.8 (2010), pp. 1893–1907 (cit. on p. 98).

[92] M. Jiang, S. Zhang, H. Li, and D. N. Metaxas. “Computer-aided diagnosis of mammographic masses using scalable image retrieval”. In: *IEEE Transactions on Biomedical Engineering* 62.2 (2015), pp. 783–792 (cit. on pp. 99, 109).

[93] M. Jiang, S. Zhang, J. Huang, L. Yang, and D. N. Metaxas. “Scalable histopathological image analysis via supervised hashing with multiple features”. In: *Medical image analysis* 34 (2016), pp. 3–12 (cit. on p. 25).

[94] A. Joly and O. Buisson. “Random maximum margin hashing”. In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. 2011, pp. 873–880 (cit. on p. 122).

[95] J. Kalpathy-Cramer, A. G. S. de Herrera, D. Demner-Fushman, S. Antani, S. Bedrick, and H. Müller. “Evaluating performance of biomedical image retrieval systems—An overview of the medical image retrieval task at ImageCLEF 2004–2013”. In: *Computerized Medical Imaging and Graphics* 39 (2015), pp. 55–61 (cit. on pp. 20, 26).

[96] M. Kandemir and F. A. Hamprecht. “Computer-aided diagnosis from weak supervision: A benchmarking study”. In: *Computerized Medical Imaging and Graphics* 42 (2015), pp. 44–50 (cit. on p. 124).

[97] N. Katayama and S. Satoh. “The SR-tree: An index structure for high-dimensional nearest neighbor queries”. In: *ACM Sigmod Record* 26.2 (1997), pp. 369–380 (cit. on p. 3).

[98] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 13).

[99] D. P. Kingma and M. Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013) (cit. on p. 13).

[100] W. Kong and W.-J. Li. “Isotropic hashing”. In: *Advances in neural information processing systems*. 2012, pp. 1646–1654 (cit. on pp. 8, 14).

[101] F. Kovács, C. Legány, and A. Babos. “Cluster validity measurement techniques”. In: *6th International symposium of hungarian researchers on computational intelligence*. 2005 (cit. on p. 35).

[102] S. Kraft, S. Conjeti, P. B. Noël, S. Carlier, N. Navab, and A. Katouzian. “Full-wave intravascular ultrasound simulation from histology”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2014, pp. 627–634 (cit. on p. 138).

[103] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105 (cit. on pp. 13, 19, 80, 87, 88, 92).

[104] A. Krogh and J. A. Hertz. “A simple weight decay can improve generalization”. In: *Advances in neural information processing systems*. 1992, pp. 950–957 (cit. on p. 13).

[105] B. Kulis et al. “Metric learning: A survey”. In: *Foundations and Trends® in Machine Learning* 5.4 (2013), pp. 287–364 (cit. on p. 54).

[106] A. Kumar, J. Kim, W. Cai, M. Fulham, and D. Feng. “Content-based medical image retrieval: a survey of applications to multidimensional and multimodality data”. In: *Journal of digital imaging* 26.6 (2013), pp. 1025–1039 (cit. on p. 23).

[107] H. Lai, Y. Pan, Y. Liu, and S. Yan. “Simultaneous feature learning and hash coding with deep neural networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3270–3278 (cit. on pp. 14, 19, 87, 88, 92, 99, 110, 117).

[108] R. Lan and Y. Zhou. “Medical Image Retrieval via Histogram of Compressed Scattering Coefficients”. In: *IEEE journal of biomedical and health informatics* (2017) (cit. on p. 25).

[109] N. Le Roux and Y. Bengio. “Representational power of restricted Boltzmann machines and deep belief networks”. In: *Neural computation* 20.6 (2008), pp. 1631–1649 (cit. on p. 13).

[110] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 13).

[111] L. Ledwich and S. Williams. “Reduced SIFT features for image retrieval and indoor localisation”. In: *Australian conference on robotics and automation*. Vol. 322. 2004, p. 3 (cit. on p. 11).

[112] Z. Li, R. Fang, F. Shen, A. Katouzian, and S. Zhang. “Indexing and mining large-scale neuron databases using maximum inner product search”. In: *Pattern Recognition* 63 (2017), pp. 680–688 (cit. on p. 25).

[113] Z. Li, X. Zhang, H. Müller, and S. Zhang. “Large-scale Retrieval for Medical Image Analytics: A Comprehensive Review”. In: *Medical Image Analysis* (2017) (cit. on p. 4).

[114] Z. Li, F. Shen, R. Fang, S. Conjeti, A. Katouzian, and S. Zhang. “Maximum inner product search for morphological retrieval of large-scale neuron data”. In: *Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on*. IEEE. 2016, pp. 602–606 (cit. on pp. 9, 137).

[115] T.-Y. Lin, M. Maire, S. Belongie, et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755 (cit. on pp. 3, 11).

[116] Y. Lin, R. Jin, D. Cai, S. Yan, and X. Li. “Compressed hashing”. In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE. 2013, pp. 446–451 (cit. on p. 9).

[117] G.-H. Liu and J.-Y. Yang. “Content-based image retrieval using color difference histogram”. In: *Pattern Recognition* 46.1 (2013), pp. 188–198 (cit. on p. 11).

[118] J. Liu, S. Zhang, W. Liu, C. Deng, Y. Zheng, and D. N. Metaxas. “Scalable Mammogram Retrieval Using Composite Anchor Graph Hashing with Iterative Quantization”. In: *IEEE Transactions on Circuits and Systems for Video Technology* (2016) (cit. on pp. 13, 25, 45).

[119] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. “Hashing with graphs”. In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 1–8 (cit. on pp. 9, 17, 66).

[120] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. “Supervised hashing with kernels”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 2074–2081 (cit. on pp. 7, 9, 14, 17, 18, 66, 72, 87, 88, 92, 99, 110, 117).

[121] E. Loupias, N. Sebe, S. Bres, and J.-M. Jolion. “Wavelet-based salient points for image retrieval”. In: *Image Processing, 2000. Proceedings. 2000 International Conference on*. Vol. 2. IEEE. 2000, pp. 518–521 (cit. on p. 11).

[122] G. Louppe. “Understanding random forests: From theory to practice”. In: *arXiv preprint arXiv:1407.7502* (2014) (cit. on p. 36).

[123] D. G. Lowe. “Object recognition from local scale-invariant features”. In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Vol. 2. Ieee. 1999, pp. 1150–1157 (cit. on p. 11).

[124] L. v. d. Maaten and G. Hinton. “Visualizing data using t-SNE”. In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605 (cit. on pp. 76, 91).

[125] L. Maier-Hein, S. Vedula, S. Speidel, et al. “Surgical data science: enabling next-generation surgery”. In: *arXiv preprint arXiv:1701.06482* (2017) (cit. on p. 22).

[126] C. Marcus, V Ladam-Marcus, C Cucu, O Bouché, L Lucas, and C Hoeffel. “Imaging techniques to evaluate the response to treatment in oncology: current standards and perspectives”. In: *Critical reviews in oncology/hematology* 72.3 (2009), pp. 217–238 (cit. on p. 22).

[127] D. Markonis, M. Holzer, F. Baroz, et al. “User-oriented evaluation of a medical image retrieval system for radiologists”. In: *International journal of medical informatics* 84.10 (2015), pp. 774–783 (cit. on p. 21).

[128] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. “Stacked convolutional auto-encoders for hierarchical feature extraction”. In: *Artificial Neural Networks and Machine Learning–ICANN 2011* (2011), pp. 52–59 (cit. on p. 13).

[129] B. H. Menze, B. M. Kelm, D. N. Splitthoff, U. Koethe, and F. A. Hamprecht. “On oblique random forests”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2011, pp. 453–469 (cit. on pp. 48, 58, 71).

[130] S. Mesbah, S. Conjeti, A. Kumaraswamy, P. Rautenberg, N. Navab, and A. Katouzian. “Hashing forests for morphological search and retrieval in neuroscientific image databases”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2015, pp. 135–143 (cit. on pp. 32, 40, 48, 49, 53, 62, 66, 70, 73, 133, 136).

[131] M. Muja and D. G. Lowe. “Fast approximate nearest neighbors with automatic algorithm configuration.” In: *VISAPP (1)* 2.331-340 (2009), p. 2 (cit. on p. 4).

[132] H. Müller. “Medical (visual) information retrieval”. In: *Information Retrieval Meets Information Visualization*. Springer, 2013, pp. 155–166 (cit. on p. 22).

[133] H. Müller, N. Michoux, D. Bandon, and A. Geissbuhler. “A review of content-based image retrieval systems in medical applications—clinical benefits and future directions”. In: *International journal of medical informatics* 73.1 (2004), pp. 1–23 (cit. on pp. 21–23).

[134] H. Muller, A. Rosset, J.-P. Vallee, and A. Geissbuhler. “Comparing features sets for content-based image retrieval in a medical- case database”. In: *Proceedings of SPIE*. Vol. 5371. 2004, pp. 99–109 (cit. on pp. 21, 24).

[135] H. Müller, P. Clough, T. Deselaers, B. Caputo, and I. CLEF. “Experimental evaluation in visual information retrieval”. In: *The Information Retrieval Series* 32 (2010) (cit. on p. 11).

[136] S. Murala, R. Maheshwari, and R Balasubramanian. “Directional binary wavelet patterns for biomedical image indexing and retrieval”. In: *Journal of Medical Systems* 36.5 (2012), pp. 2865–2879 (cit. on p. 25).

[137] V. Nair and G. E. Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814 (cit. on p. 12).

[138] A. P. Natsev, A. Haubold, J. Tešić, L. Xie, and R. Yan. “Semantic concept-based query expansion and re-ranking for multimedia retrieval”. In: *Proceedings of the 15th ACM international conference on Multimedia*. ACM. 2007, pp. 991–1000 (cit. on p. 4).

[139] F. Nielsen, P. Piro, and M. Barlaud. “Bregman vantage point trees for efficient nearest neighbor queries”. In: *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*. IEEE. 2009, pp. 878–881 (cit. on p. 4).

[140] D. Nister and H. Stewenius. “Scalable recognition with a vocabulary tree”. In: *Computer vision and pattern recognition, 2006 IEEE computer society conference on*. Vol. 2. Ieee. 2006, pp. 2161–2168 (cit. on p. 4).

[141] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov. “Hamming distance metric learning”. In: *Advances in neural information processing systems*. 2012, pp. 1061–1069 (cit. on pp. 9, 14).

[142] A. Oliva and A. Torralba. “Modeling the shape of the scene: A holistic representation of the spatial envelope”. In: *International journal of computer vision* 42.3 (2001), pp. 145–175 (cit. on pp. 11, 88, 115).

[143] M. Ou, P. Cui, F. Wang, J. Wang, and W. Zhu. “Non-transitive hashing with latent similarity components”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2015, pp. 895–904 (cit. on p. 14).

[144] D. L. Page, W. D. Dupont, L. W. Rogers, and M. S. Rados. “Atypical hyperplastic lesions of the female breast. A long-term follow-up study”. In: *cancer* 55.11 (1985), pp. 2698–2708 (cit. on p. 98).

[145] R. Parekh and G. A. Ascoli. “Neuronal morphology goes digital: a research hub for cellular and system neuroscience”. In: *Neuron* 77.6 (2013), pp. 1017–1038 (cit. on p. 62).

[146] H. Peng, B. Roysam, and G. A. Ascoli. “Automated image computing reshapes computational neuroscience”. In: *Bmc Bioinformatics* 14.1 (2013), p. 293 (cit. on p. 130).

[147] H. Peng, Z. Ruan, F. Long, J. H. Simpson, and E. W. Myers. “V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets”. In: *Nature biotechnology* 28.4 (2010), pp. 348–353 (cit. on p. 130).

[148] T. Peng, L. Wang, C. Bayer, S. Conjeti, M. Baust, and N. Navab. “Shading correction for whole slide image using low rank and sparse decomposition”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2014, pp. 33–40 (cit. on p. 138).

[149] S. Polavaram, T. A. Gillette, R. Parekh, and G. A. Ascoli. “Statistical analysis and data mining of digital reconstructions of dendritic morphologies”. In: *Frontiers in neuroanatomy* 8 (2014) (cit. on pp. 48, 131).

[150] S. Pölsterl, P. Gupta, L. Wang, S. Conjeti, A. Katouzian, and N. Navab. “Heterogeneous ensembles for predicting survival of metastatic, castrate-resistant prostate cancer patients”. In: *F1000Research* 5 (2016) (cit. on pp. 133, 137).

[151] S. Pölsterl, S. Conjeti, N. Navab, and A. Katouzian. “Survival analysis for high-dimensional, heterogeneous medical data: Exploring feature extraction as an alternative to feature selection”. In: *Artificial intelligence in medicine* 72 (2016), pp. 1–11 (cit. on p. 137).

[152] M. M. Rahman, P. Bhattacharya, and B. C. Desai. “A framework for medical image retrieval using machine learning and statistical similarity matching techniques with relevance feedback”. In: *IEEE transactions on Information Technology in Biomedicine* 11.1 (2007), pp. 58–69 (cit. on pp. 26, 124).

[153] R. M. Rangayyan, F. J. Ayres, and J. L. Desautels. “A review of computer-aided diagnosis of breast cancer: Toward the detection of subtle signs”. In: *Journal of the Franklin Institute* 344.3 (2007), pp. 312–348 (cit. on p. 98).

[154] M. Rastegari, A. Farhadi, and D. Forsyth. “Attribute discovery via predictable discriminative binary codes”. In: *European Conference on Computer Vision*. Springer. 2012, pp. 876–889 (cit. on p. 14).

[155] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. “Xnor-net: Imagenet classification using binary convolutional neural networks”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 525–542 (cit. on p. 123).

[156] P. L. Rautenberg, A. Kumaraswamy, A. Tejero-Cantero, et al. “NeuronDepot: keeping your colleagues in sync by combining modern cloud storage services, the local file system, and simple web applications”. In: *Frontiers in neuroinformatics* 8 (2014) (cit. on p. 129).

[157] P. L. Rautenberg, B. Grothe, and F. Felmy. “Quantification of the three-dimensional morphology of coincidence detector neurons in the medial superior olive of gerbils during late postnatal development”. In: *Journal of Comparative Neurology* 517.3 (2009), pp. 385–396 (cit. on p. 131).

[158] A Redondo, M Comas, F Macia, et al. “Inter-and intraradiologist variability in the BI-RADS assessment and breast density categories for screening mammograms”. In: *The British journal of radiology* 85.1019 (2012), pp. 1465–1470 (cit. on p. 98).

[159] A. G. Roy, S. Conjeti, S. G. Carlier, et al. “Bag of forests for modelling of tissue energy interaction in optical coherence tomography for atherosclerotic plaque susceptibility assessment”. In: *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*. IEEE. 2015, pp. 428–431 (cit. on p. 137).

[160] A. G. Roy, S. Conjeti, D. Sheet, A. Katouzian, N. Navab, and C. Wachinger. “Error Corrective Boosting for Learning Fully Convolutional Networks with Limited Data”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2017, pp. 231–239 (cit. on pp. 13, 133, 136).

[161] A. G. Roy, S. Conjeti, S. G. Carlier, et al. “Lumen segmentation in intravascular optical coherence tomography using backscattering tracked and initialized random walks”. In: *IEEE journal of biomedical and health informatics* 20.2 (2016), pp. 606–614 (cit. on p. 137).

[162] A. G. Roy, S. Conjeti, S. G. Carlier, et al. “Multiscale distribution preserving autoencoders for plaque detection in intravascular optical coherence tomography”. In: *Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on*. IEEE. 2016, pp. 1359–1362 (cit. on p. 137).

[163] A. G. Roy, S. Conjeti, S. P. K. Karri, et al. “ReLayNet: retinal layer and fluid segmentation of macular optical coherence tomography using fully convolutional networks”. In: *Biomed. Opt. Express* 8.8 (2017), pp. 3627–3642 (cit. on p. 135).

[164] R. Salakhutdinov and G. Hinton. “Semantic hashing”. In: *International Journal of Approximate Reasoning* 50.7 (2009), pp. 969–978 (cit. on pp. 18, 19).

[165] J. Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural networks* 61 (2015), pp. 85–117 (cit. on p. 12).

[166] R. Scorcioni, S. Polavaram, and G. A. Ascoli. “L-measure: a web-accessible tool for the analysis, comparison, and search of digital reconstructions of neuronal morphologies”. In: *Nature protocols* 3.5 (2008), p. 866 (cit. on pp. 39, 62, 74, 131, 132).

[167] D. C. Sgroi. “Preinvasive breast cancer”. In: *Annual Review of Pathological Mechanical Disease* 5 (2010), pp. 193–221 (cit. on p. 98).

[168] A. Shah, S. Conjeti, N. Navab, and A. Katouzian. "Deeply learnt hashing forests for content based image retrieval in prostate MR images." In: *Medical Imaging: Image Processing*. 2016, p. 978414 (cit. on p. 137).

[169] G. Shakhnarovich, P. Viola, and T. Darrell. "Fast pose estimation with parameter-sensitive hashing". In: *null*. IEEE. 2003, p. 750 (cit. on p. 14).

[170] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. "CNN features off-the-shelf: an astounding baseline for recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2014, pp. 806–813 (cit. on p. 13).

[171] D. Sheet, S. P. K. Karri, S. Conjeti, S. Ghosh, J. Chatterjee, and A. K. Ray. "Detection of retinal vessels in fundus images through transfer learning of tissue specific photon interaction statistical physics". In: *Biomedical Imaging (ISBI), 2013 IEEE 10th International Symposium on*. IEEE. 2013, pp. 1452–1456 (cit. on p. 138).

[172] F. Shen, C. Shen, Q. Shi, A. Van Den Hengel, and Z. Tang. "Inductive hashing on manifolds". In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE. 2013, pp. 1562–1569 (cit. on p. 13).

[173] F. Shen, C. Shen, W. Liu, and H. T. Shen. "Supervised Discrete Hashing." In: *CVPR*. Vol. 2. 3. 2015, p. 5 (cit. on p. 14).

[174] H.-C. Shin, K. Roberts, L. Lu, D. Demner-Fushman, J. Yao, and R. M. Summers. "Learning to read chest X-rays: recurrent neural cascade model for automated image annotation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2497–2506 (cit. on pp. 86, 88).

[175] C.-R. Shyu, C. E. Brodley, A. C. Kak, A. Kosaka, A. M. Aisen, and L. S. Broderick. "ASSERT: A physician-in-the-loop content-based retrieval system for HRCT image databases". In: *Computer Vision and Image Understanding* 75.1-2 (1999), pp. 111–132 (cit. on p. 27).

[176] R. R. Singh, S. Conjeti, and R. Banerjee. "A comparative evaluation of neural network classifiers for stress level analysis of automotive drivers using physiological signals". In: *Biomedical Signal Processing and Control* 8.6 (2013), pp. 740–754 (cit. on p. 138).

[177] R. R. Singh, S. Conjeti, and R. Banerjee. "An approach for real-time stress-trend detection using physiological signals in wearable computing systems for automotive drivers". In: *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*. IEEE. 2011, pp. 1477–1482 (cit. on p. 139).

[178] R. R. Singh, S. Conjeti, and R. Banerjee. "Assessment of driver stress from physiological signals collected under real-time semi-urban driving scenarios". In: *International Journal of Computational Intelligence Systems* 7.5 (2014), pp. 909–923 (cit. on p. 138).

[179] R. R. Singh, S. Conjeti, and R. Banerjee. "Biosignal based on-road stress monitoring for automotive drivers". In: *Communications (NCC), 2012 National Conference on*. IEEE. 2012, pp. 1–5 (cit. on p. 139).

[180] M. Slaney and M. Casey. "Locality-sensitive hashing for finding nearest neighbors [lecture notes]". In: *IEEE Signal Processing Magazine* 25.2 (2008), pp. 128–131 (cit. on pp. 7, 15, 17, 33, 41, 66, 79, 87, 92).

[181] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. "Content-based image retrieval at the end of the early years". In: *IEEE Transactions on pattern analysis and machine intelligence* 22.12 (2000), pp. 1349–1380 (cit. on p. 3).

[182] R. A. Smith, V. Cokkinides, and H. J. Eyre. "American Cancer Society guidelines for the early detection of cancer, 2006". In: *CA: a cancer journal for clinicians* 56.1 (2006), pp. 11–25 (cit. on p. 98).

[183] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese. “Deep metric learning via lifted structured feature embedding”. In: *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE. 2016, pp. 4004–4012 (cit. on p. 15).

[184] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Dropout: a simple way to prevent neural networks from overfitting.” In: *Journal of machine learning research* 15.1 (2014), pp. 1929–1958 (cit. on p. 13).

[185] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua. “LDAHash: Improved matching with smaller descriptors”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.1 (2012), pp. 66–78 (cit. on pp. 8, 17, 32).

[186] A. Sze-To, H. R. Tizhoosh, and A. K. Wong. “Binary codes for tagging x-ray images via deep de-noising autoencoders”. In: *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE. 2016, pp. 2864–2871 (cit. on p. 25).

[187] C. Szegedy, W. Liu, Y. Jia, et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9 (cit. on pp. 13, 102, 109, 114).

[188] O. A. Jiménez-del Toro, A. Hanbury, G. Langs, A. Foncubierta-Rodríguez, and H. Müller. “Overview of the VISCERAL retrieval benchmark 2015”. In: *Multimodal retrieval in the medical domain*. Springer, 2015, pp. 115–123 (cit. on p. 20).

[189] A. Torralba, R. Fergus, and Y. Weiss. “Small codes and large image databases for recognition”. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–8 (cit. on pp. 13, 19, 104, 112).

[190] A. Vedaldi and K. Lenc. “Matconvnet: Convolutional neural networks for matlab”. In: *Proceedings of the 23rd ACM international conference on Multimedia*. ACM. 2015, pp. 689–692 (cit. on pp. 88, 110).

[191] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. “Extracting and composing robust features with denoising autoencoders”. In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 1096–1103 (cit. on p. 13).

[192] Y. Wan, F. Long, L. Qu, et al. “BlastNeuron for automated comparison, retrieval and clustering of 3D neuron morphologies”. In: *Neuroinformatics* 13.4 (2015), pp. 487–499 (cit. on pp. 130, 131).

[193] J. Wang, H. T. Shen, J. Song, and J. Ji. “Hashing for similarity search: A survey”. In: *arXiv preprint arXiv:1408.2927* (2014) (cit. on pp. 3, 6, 8).

[194] J. Wang, A. Woznica, and A. Kalousis. “Learning neighborhoods for metric learning”. In: *Machine Learning and Knowledge Discovery in Databases* (2012), pp. 223–236 (cit. on p. 62).

[195] J. Wang, W. Liu, S. Kumar, and S.-F. Chang. “Learning to hash for indexing big data—a survey”. In: *Proceedings of the IEEE* 104.1 (2016), pp. 34–57 (cit. on pp. 6, 36).

[196] J. Wang, S. Kumar, and S.-F. Chang. “Semi-supervised hashing for scalable image retrieval”. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE. 2010, pp. 3424–3431 (cit. on pp. 14, 124).

[197] J. Wang, S. Kumar, and S.-F. Chang. “Sequential projection learning for hashing with compact codes”. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 1127–1134 (cit. on pp. 8, 14).

[198] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers. “ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases”. In: *arXiv preprint arXiv:1705.02315* (2017) (cit. on pp. 6, 11).

[199] K. Q. Weinberger and L. K. Saul. “Distance metric learning for large margin nearest neighbor classification”. In: *Journal of Machine Learning Research* 10.Feb (2009), pp. 207–244 (cit. on p. 54).

[200] Y. Weiss, R. Fergus, and A. Torralba. “Multidimensional spectral hashing”. In: *Computer Vision–ECCV 2012* (2012), pp. 340–353 (cit. on p. 17).

[201] Y. Weiss, A. Torralba, and R. Fergus. “Spectral hashing”. In: *Advances in neural information processing systems*. 2009, pp. 1753–1760 (cit. on pp. 6, 9, 13, 15–17, 33, 41, 53, 66).

[202] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell. “Understanding data augmentation for classification: when to warp?” In: *Digital Image Computing: Techniques and Applications (DICTA), 2016 International Conference on*. IEEE. 2016, pp. 1–6 (cit. on p. 13).

[203] J. Wu, Y. Yu, C. Huang, and K. Yu. “Deep multiple instance learning for image classification and auto-annotation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3460–3469 (cit. on p. 99).

[204] T. Xu, H. Zhang, X. Huang, S. Zhang, and D. N. Metaxas. “Multimodal deep learning for cervical dysplasia diagnosis”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2016, pp. 115–123 (cit. on p. 124).

[205] Z. Xu and M. Xia. “Distance and similarity measures for hesitant fuzzy sets”. In: *Information Sciences* 181.11 (2011), pp. 2128–2138 (cit. on p. 8).

[206] Z. Xue, S Antani, L. Long, J. Jeronimo, and G Thoma. “A Web-accessible content-based cervicographic image retrieval system”. In: *Proceedings of SPIE medical imaging*. Vol. 6919. 2008, pp. 691907–1 (cit. on p. 26).

[207] Z. Yan, Y. Zhan, Z. Peng, et al. “Multi-instance deep learning: Discover discriminative local anatomies for bodypart recognition”. In: *IEEE transactions on medical imaging* 35.5 (2016), pp. 1332–1343 (cit. on p. 99).

[208] Y. Yang, X.-S. Xu, X. Wang, S. Guo, and L. Cui. “Hashing Multi-Instance Data from Bag and Instance Level”. In: *Asia-Pacific Web Conference*. Springer. 2015, pp. 437–448 (cit. on pp. 99, 110).

[209] T. Yao, F. Long, T. Mei, and Y. Rui. “Deep Semantic-Preserving and Ranking-Based Hashing for Image Retrieval.” In: *IJCAI*. 2016, pp. 3931–3937 (cit. on p. 20).

[210] P. N. Yianilos. “Data structures and algorithms for nearest neighbor search in general metric spaces”. In: *SODA*. 194. 1993, pp. 311–321 (cit. on pp. 4, 33).

[211] C. Yu, Z. Han, W. Zeng, and S. Liu. “Morphology cluster and prediction of growth of human brain pyramidal neurons”. In: *Neural regeneration research* 7.1 (2012), p. 36 (cit. on p. 62).

[212] G. Yu and J. Yuan. “Scalable forest hashing for fast similarity search”. In: *Multimedia and Expo (ICME), 2014 IEEE International Conference on*. IEEE. 2014, pp. 1–6 (cit. on pp. 33, 41, 47, 122).

[213] H. Yu, M. Li, H.-J. Zhang, and J. Feng. “Color texture moments for content-based image retrieval”. In: *Image Processing. 2002. Proceedings. 2002 International Conference on*. Vol. 3. IEEE. 2002, pp. 929–932 (cit. on p. 11).

[214] X. Yu, S. Zhang, B. Liu, L. Zhong, and D. Metaxas. “Large scale medical image search via unsupervised PCA hashing”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2013, pp. 393–398 (cit. on pp. 8, 15, 32).

[215] X. Yuan, J. Yu, Z. Qin, and T. Wan. “A SIFT-LBP image retrieval model based on bag of features”. In: *IEEE International Conference on Image Processing*. 2011 (cit. on p. 11).

[216] H. Zaidi, H. Vees, and M. Wissmeyer. “Molecular PET/CT imaging-guided radiation therapy treatment planning”. In: *Academic Radiology* 16.9 (2009), pp. 1108–1133 (cit. on p. 22).

[217] L. Zelnik-Manor and P. Perona. “Self-tuning spectral clustering”. In: *Advances in neural information processing systems*. 2005, pp. 1601–1608 (cit. on p. 56).

[218] D. Zhang, J. Wang, D. Cai, and J. Lu. “Self-taught hashing for fast similarity search”. In: *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2010, pp. 18–25 (cit. on pp. 14, 41).

[219] D. Zhang, A. Wong, M. Indrawan, and G. Lu. “Content-based image retrieval using Gabor texture features”. In: *IEEE Transactions PAMI* (2000), pp. 13–15 (cit. on p. 11).

[220] L. Zhang, Y. Zhang, J. Tang, K. Lu, and Q. Tian. “Binary code ranking with weighted hamming distance”. In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE. 2013, pp. 1586–1593 (cit. on p. 8).

[221] X. Zhang, F. Xing, H. Su, L. Yang, and S. Zhang. “High-throughput histopathological image analysis via robust cell segmentation and hashing”. In: *Medical image analysis* 26.1 (2015), pp. 306–315 (cit. on p. 24).

[222] X. Zhang, W. Liu, M. Dundar, S. Badve, and S. Zhang. “Towards large-scale histopathological image analysis: Hashing-based image retrieval”. In: *IEEE Transactions on Medical Imaging* 34.2 (2015), pp. 496–506 (cit. on pp. 99, 109).

[223] X. Zhang, H. Su, L. Yang, and S. Zhang. “Weighted hashing with multiple cues for cell-level analysis of histopathological images”. In: *International Conference on Information Processing in Medical Imaging*. Springer. 2015, pp. 303–314 (cit. on pp. 99, 123).

[224] H. Zhu, M. Long, J. Wang, and Y. Cao. “Deep Hashing Network for Efficient Similarity Retrieval.” In: *AAAI*. 2016, pp. 2415–2421 (cit. on pp. 14, 19, 82, 87, 92, 99, 107, 110, 117).

Online Resources

[225] *Mesh: Medical subject headings*. 2017. URL: `https://www.nlm.nih.gov/mesh/meshhome.html` (visited on Oct. 11, 2017) (cit. on p. 86).

[226] Neuromorpho.Org. *Literature Search Main Results (2015)*. 2015. URL: `http://neuromorpho.org/neuroMorpho/LS_queryStatus.jsp` (visited on Aug. 25, 2015) (cit. on p. 39).

[227] Neuromorpho.Org. *Search by Cell Type*. 2015. URL: `http://neuromorpho.org/bycell.jsp` (visited on 2015) (cit. on p. 62).

[228] Neuromorpho.Org. *Search by Morphometry*. 2015. URL: `http://neuromorpho.org/neuroMorpho/MorphometrySearch.jsp` (visited on 2015) (cit. on p. 122).

[229] *Open-i: An open access biomedical search engine.* 2017. URL: `https://openi.nlm.nih.gov` (visited on Oct. 11, 2017) (cit. on p. 86).

[230] S. Polavaram. *L-Measure v.5.3*. 2018. URL: `http://cng.gmu.edu:8080/Lm/` (visited on Jan. 18, 2018) (cit. on p. 132).

List of Figures

List of Tables