

# Chapter 3

## Experimental Methodology

### 3.1 Introduction

In this chapter I describe the experimental methodology adopted throughout the thesis. This includes the datasets selected as the testbed for the retrieval experiments (Section 3.2), the definition of groundtruth for evaluation (Section 3.3) and the metrics adopted to ascertain retrieval effectiveness (Section 3.6). I attempt to keep the evaluation strategy aligned as closely as possible to previously related work in the learning to hash literature. However, as I will discuss throughout this section, the evaluation methodology used by previous related work is not consistent across publications and exhibits certain flaws in the experimental design. This chapter describes my remedy for these flaws and places the evaluation on a standard foundation.

### 3.2 Datasets

The focus of this thesis is learning hash functions for the task of large-scale image retrieval. This task will be split into three sub-tasks: 1) an image is used to retrieve related images from a still image archive, 2) a text query is used to retrieve related images, 3) an image query is used to retrieve relevant annotations for that image. I will therefore conduct both unimodal and cross-modal retrieval experiments in this thesis which will cover a wide range of important use-cases in image retrieval from query-by-example search to image annotation. Unimodal datasets are those where the query and the database are in the same visual modality such as bag-of-visual-word feature descriptors. Cross-modal datasets permit retrieval experiments that straddle two different modalities such as a textual query executed against an image database. The latter task

mimics the familiar image search scenario offered by many modern web search engines. In all cases I will constrain the evaluation to baselines that perform these tasks using hashing-based ANN search and do not seek to compare against, for example, fully fledged image annotation models. To align the evaluation closely with the learning to hash literature I select a subset of the most popular datasets from both categories as my experimental testbed (Sections 3.2.1-3.2.2). My desiderata for dataset selection is two-fold: firstly, the datasets must be publicly available to enable replication of experimental results by a third party; and secondly the datasets must be standard in the sense that they have been widely used in related publications. This ensures that the experimental results published in this thesis are reproducible and directly comparable to previously published research.

### 3.2.1 Unimodal Retrieval Experiments

For the unimodal experiments, I select four popular and freely available image datasets: LabelMe, CIFAR-10, NUS-WIDE and SIFT1M. The datasets are of widely varying size (22,019-1 million images), are represented by an array of different feature descriptors (from GIST, SIFT to bag of visual words) and cover a diverse range of different image topics from natural scenes to personal photos, logos and drawings. These properties ensure that the datasets will provide a challenging test suite for evaluation in this thesis. All datasets are identical to those used in many recent publications (Kong and Li (2012a), Shen et al. (2015), Liu et al. (2012)) and are available online to the research community.

- **LABELME:** 22,019 images represented as 512 dimensional GIST descriptors (Torralba et al. (2008); Russell et al. (2008))<sup>1</sup> The dataset is mean centred.
- **CIFAR-10:** 60,000  $32 \times 32$  colour images sampled from the 80 million Tiny Images dataset (Krizhevsky and Hinton (2009)). Each image is encoded with a 512 dimensional GIST descriptor (Oliva and Torralba (2001)) and is manually assigned a label from a selection of 10 classes<sup>2</sup>. Each class has 6,000 associated images. The visual feature descriptors are mean centered.
- **NUS-WIDE:** 269,648 images downloaded from Flickr each annotated with multiple ground truth concept tags (e.g. nature, dog, animal, swimming, car) from

---

<sup>1</sup><http://www.cs.toronto.edu/~norouzi/research/mlh/>

<sup>2</sup><http://www.cs.toronto.edu/~kriz/cifar.html>



Figure 3.1: The NUS-WIDE dataset consists of around 270,000 images randomly sampled from Flickr. Given the diversity of images (from people, animals, landscapes to buildings and drawings) and widely varying resolution the dataset provides a challenging testbed for image retrieval.

an 81 concept vocabulary<sup>3</sup> (Chua et al. (2009)). The images are represented by a 500 dimensional bag-of-visual-words (BoW) feature descriptor formed by vector quantising SIFT descriptors via k-means clustering. The visual feature descriptors are  $L_2$ -normalised to unit length and mean centered. For illustrative purposes I show a random sampling of images from the NUS-WIDE dataset in Figure 3.1.

- **SIFT1M:** 1,000,000 images from Flickr encoded with 128-dimensional SIFT descriptors<sup>4</sup>. This dataset was first introduced by Jegou et al. (2011) and has since become a standard image collection for evaluating nearest neighbour search methods (Kong and Li (2012a), He et al. (2013), Wang et al. (2010b)). The dataset is mean centred.

<sup>3</sup><http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm>

<sup>4</sup><http://lear.inrialpes.fr/~jegou/data.php>

Dataset	# images	# labels	Labels/image	Images/label	Descriptor
LABELME	22,019	–	–	–	512-D Gist
CIFAR-10	60,000	10	1	6,000	512-D Gist
NUS-WIDE	269,648	81	1.87	6,220	500-D BoW
SIFT1M	1,000,000	–	–	–	128-D SIFT

Table 3.1: Salient statistics of the four datasets used in my unimodal experimental evaluation. The Labels/image and Images/label are the mean values computed on the entire dataset.

### 3.2.2 Cross-modal Retrieval Experiments


The cross-modal retrieval experiments are conducted on the two most popular cross-modal datasets in the learning to hash literature, namely the ‘Wiki’ dataset and NUS-WIDE (Kumar and Udupa (2011); Zhen and Yeung (2012); Song et al. (2013); Rastegari et al. (2013); Bronstein et al. (2010)). Both datasets come with images and associated paired textual descriptors, a key requirement for training and evaluating a cross-modal retrieval model. As for the unimodal retrieval datasets described in Section 3.2.1 these two cross-modal datasets are also freely available to the research community.

- **Wiki:** is generated from 2,866 Wikipedia articles<sup>5</sup> derived from Wikipedia’s “feature articles” (Rasiwasia et al. (2010)). The featured articles segment of Wikipedia hosts the highest quality articles on the site as judged by a panel of independent Wikipedia editors. Each feature article is a document consisting of multiple sections and annotated with at least one relevant image from the Wikimedia commons. Each article is designated with a manually labelled category out of 29 possibilities. Rasiwasia et al. (2010) only keep the articles pertaining to the 10 most populated categories. Each article is further split by section and the image manually placed in that section by the author(s) is used as the corresponding visual description of the text in that section. Any section that ends up without an associated image is discarded. This leaves 2,866 short and focused “articles” of a median length of 200 words, with each article having at least 70 words. I use the image and text feature set provided by Rasiwasia et al. (2010) which is used in most related cross-modal hashing research (Zhen and Yeung (2012)). The visual modality is represented as a 128-dimensional SIFT (Lowe

<sup>5</sup><http://www.svcl.ucsd.edu/projects/crossmodal/>

### Wales and Offa's Dyke [edit]

Offa was frequently in conflict with the various Welsh kingdoms. There was a battle between the Mercians and the Welsh at [Hereford](#) in 760, and Offa is recorded as campaigning against the Welsh in 778, 784 and 796 in the tenth-century *Annales Cambriae*.<sup>[84][85]</sup>




Looking along Offa's Dyke, near Knill, Herefordshire

The best known relic associated with Offa's time is [Offa's Dyke](#), a great earthen barrier that runs approximately along the border between England and [Wales](#). It is mentioned by the monk [Asser](#) in his biography of Alfred the Great: "a certain vigorous king called Offa ... had a great dyke built between Wales and Mercia from sea to sea"<sup>[86]</sup> The dyke has not been dated by archaeological methods, but most historians find no reason to doubt Asser's attribution<sup>[87]</sup> Early names for the dyke in both Welsh and English also support the attribution to Offa<sup>[88]</sup> Despite Asser's comment that the dyke ran "from sea to sea", it is now thought that the original structure only covered about two-thirds of the length of the border: in the north it ends near [Llanfynydd](#), less than five miles (8 km) from the coast, while in the south it stops at [Rushock Hill](#), near [Kingston](#) in Herefordshire, less than fifty miles (80 km) from the [Bristol Channel](#). The total length of this section is about sixty-four miles (103 km)<sup>[87]</sup> Other earthworks exist along the Welsh border, of which [Wat's Dyke](#) is one of the largest, but it is not possible to date them relative to each other and so it cannot be determined whether Offa's Dyke was a copy of or the inspiration for [Wat's Dyke](#).<sup>[89]</sup>

The construction of the dyke suggests that it was built to create an effective barrier and to command views into Wales. This implies that the Mercians who built it were free to choose the best location for the dyke.<sup>[87]</sup> There are settlements to the west of the dyke that have names that imply they were English by the eighth century, so it may be that in choosing the location of the barrier the Mercians were consciously surrendering some territory to the native Britons.<sup>[80]</sup> Alternatively it may be that these settlements had already been retaken by the Welsh, implying a defensive role for the barrier. The effort and expense that must have gone into building the dyke are impressive, and suggest that the king who had it built (whether Offa or someone else) had considerable resources at his disposal. Other substantial construction projects of a similar date do exist, however, such as [Wat's Dyke](#) and the [Danevirke](#), in what is now [Denmark](#), as well as such sites as [Stonehenge](#) from millennia earlier. The dyke can be regarded in the light of these counterparts as the largest and most recent great construction of the preliterare

Inhabitants of Britain.<sup>[81]</sup>



The Tasmanian devil's whiskers help it to locate prey in the dark.

The Tasmanian devil is the largest surviving carnivorous [marsupial](#). It has a squat, thick build, with a large head and a tail which is about half its body length. Unusually for a marsupial, its forelegs are slightly longer than its hind legs, and devils can run up to 13 km/h (8.1 mph) for short distances. The fur is usually black, often with irregular white patches on the chest and rump (although approximately 16% of wild devils do not have white patches).<sup>[37][38]</sup> These markings suggest that the devil is most active at dawn and dusk, and they are thought to draw biting attacks toward less important areas of the body, as fighting between devils often leads to a concentration of scars in that region.<sup>[38]</sup> Males are usually larger than females, having an average head and body length of 652 mm (25.7 in), a 258 mm (10.2 in) tail and an average weight of 8 kg (18 lb). Females have an average head and body length of 570 mm (22 in), a 244 mm (9.6 in) tail and an average weight of 6 kg (13 lb).<sup>[37]</sup> although devils in western Tasmania tend to be smaller.<sup>[39]</sup> Devils have five long toes on their forefeet, four pointing to the front and one coming out from the side, which gives the devil the ability to hold food. The hind feet have four toes, and the devils have non-retractable claws.<sup>[35]</sup> The stocky devils have a relatively low [centre of mass](#).<sup>[40]</sup>

Devils are fully grown at two years of age.<sup>[34]</sup> and few devils live longer than five years in the wild.<sup>[41]</sup> Possibly the longest-lived Tasmanian devil recorded was Coolah, a male devil which lived in captivity for more than seven years.<sup>[42]</sup> Born in January 1997 at the [Cincinnati Zoo](#), Coolah died in May 2004 at the [Fort Wayne Children's Zoo](#).<sup>[43]</sup>

### Anthills and paralysis [edit]

In 1987 Achebe released his fifth novel, *[Anthills of the Savannah](#)*, about a military coup in the fictional West African nation of Kangan. A finalist for the [Booker Prize](#), the novel was hailed in the *[Financial Times](#)*: "in a powerful fusion of myth, legend and modern styles, Achebe has written a book which is wise, exciting and essential, a powerful antidote to the cynical commentators from 'overseas' who see nothing ever new out of Africa."<sup>[132]</sup> An opinion piece in the magazine *[West Africa](#)* said the book deserved to win the Booker Prize, and that Achebe was "a writer who has long deserved the recognition that has already been accorded him by his sales figures."<sup>[132]</sup> The prize went instead to [Penelope Lively](#)'s novel *[Moon Tiger](#)*.


On 22 March 1990, Achebe was riding in a car to Lagos when an axle collapsed and the car flipped. His son Ikechukwu and the driver suffered minor injuries, but the weight of the vehicle fell on Achebe and his spine was severely damaged. He was flown to the Paddocks Hospital in [Buckinghamshire](#), England, and treated for his injuries. In July doctors announced that although he was recuperating well, he was paralyzed from the waist down and would require the use of a wheelchair for the rest of his life.<sup>[133]</sup>

Soon afterwards, Achebe became the Charles P. Stevenson Professor of Languages and Literature at [Bard College](#) in [Annandale-on-Hudson](#), New York; he held the position for more than fifteen years.<sup>[134]</sup> In the autumn of 2009 he joined the Brown University faculty as the David and Marianna Fisher University Professor of Africana Studies.<sup>[135]</sup>

### Later life and death [edit]

In October 2005, the London *[Financial Times](#)* reported that Achebe was planning to write a novella for the *[Canongate Myth Series](#)*, a series of short novels in which ancient myths from myriad cultures are reimagined and rewritten by contemporary authors.<sup>[136]</sup> Achebe's novella has not yet been scheduled for publication.

In June 2007, Achebe was awarded the [Man Booker International Prize](#).<sup>[137]</sup> The judging panel included US critic [Elaine Showalter](#), who said he "illuminated the path for writers around the world seeking new words and forms for new realities and societies".<sup>[138]</sup> and South African writer [Nadine Gordimer](#), who said Achebe has achieved "what one of his characters brilliantly defines as the writer's purpose: 'a new-found utterance' for the capture of life's complexity".<sup>[138]</sup> In 2010, Achebe was awarded *The Dorothy and Lillian Gish Prize* for \$300,000, one of the richest prizes for the arts.<sup>[139]</sup>



Stone Row at the centre of the [Bard College](#) campus

Figure 3.2: Three example Wikipedia article sections (Offa King of Mercia ([https://en.wikipedia.org/wiki/Offa\\_of\\_Mercia](https://en.wikipedia.org/wiki/Offa_of_Mercia)), the Tasmanian devil ([https://en.wikipedia.org/wiki/Tasmanian\\_devil](https://en.wikipedia.org/wiki/Tasmanian_devil)) and a description from the life of Nigerian novelist Chinua Achebe ([https://en.wikipedia.org/wiki/Chinua\\_Achebe](https://en.wikipedia.org/wiki/Chinua_Achebe)) and their aligned images taken from the cross-modal Wiki dataset.

(2004)) bag-of-words histogram, while the textual modality is represented as 10-dimensional probability distribution over Latent Dirichlet Allocation (LDA) topics (Blei et al. (2003)). Three example Wikipedia article sections and their associated images are shown in Figure 3.2.

- NUS-WIDE:** is identical to the unprocessed NUS-WIDE dataset described in Section 3.2.1 (Chua et al. (2009)). For my cross-modal experiments I pre-process the dataset in a different manner to the strategy described in Section 3.2.1 so that my experiments are compatible with those presented in the relevant literature (Zhen and Yeung (2012)). More specifically, for cross-modal retrieval the multiple image tags associated with an image are used to define the textual modality. I



keep the image-text pairs associated with the most frequent 10 classes. Each image is associated with a subset of 5,018 tags manually assigned by Flickr users. I perform a PCA dimensionality reduction on the  $269,648 \times 5018$  dimensional tag co-occurrence matrix to form a 1,000-dimensional tag feature set. This projected tag feature set is then mean-centered and used as a representation of the textual modality. This is a standard pre-processing step in the literature (Zhen and Yeung (2012)). The visual modality is represented by the same 500 dimensional bag-of-words (BoW) feature descriptors described in the context of NUS-WIDE in Section 3.2.1. The visual descriptors are  $L_2$ -normalised to unit length and mean centered.

### 3.3 Nearest Neighbour Groundtruth Definition

In order to evaluate the retrieval effectiveness of a hashing model we need to define which data-points in the database are considered to be nearest neighbours of the query data-points. I refer to these data-points as the *true* nearest neighbours of a query. The system is penalised depending on the degree to which it fails to return the true nearest neighbours for a query. The definition of the groundtruth nearest neighbours varies widely between publications. In this thesis I consider two of the strategies commonly used to define groundtruth which involves either constructing an  $\epsilon$ -ball around the query data-point (Section 3.3.1) or using human assigned class-labels (Section 3.3.2). To the best of my knowledge, there has been no work to verify whether or not the  $\epsilon$ -ball groundtruth definition correlates with user search satisfaction. I discuss this point further in Chapter 8 as part of possible future work.

#### 3.3.1 $\epsilon$ -Ball Nearest Neighbours

I opt primarily for the  $\epsilon$ -nearest neighbour ( $\epsilon$ -NN) definition in this thesis (Figure 3.3a)<sup>6</sup>. In this paradigm a ball of radius  $\epsilon$  is defined around a query data-point in the input feature space and the true nearest neighbours are defined as those data-points enclosed within the ball. To compute the  $\epsilon$ -NN groundtruth I follow previous related work (Kong et al. (2012); Kong and Li (2012a); Kulis and Darrell (2009); Gong and Lazebnik (2011)) and randomly sample 100 data-points from the training dataset to

---

<sup>6</sup>Defining ground-truth nearest neighbours can also be achieved by computing a k-NN graph. In this case related data-points to a query are those that have the k smallest distances to the query. I leave this type of evaluation as future work.

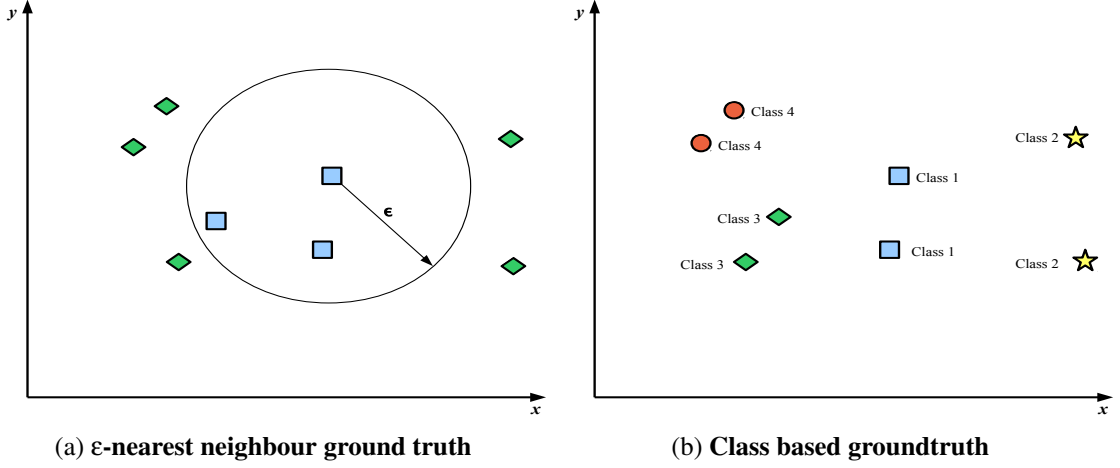


Figure 3.3: Two definitions of groundtruth nearest neighbours (NN). In Figure (a) I show how to define nearest neighbours of a data-point using an  $\epsilon$ -ball. All data-points enclosed by the ball are nearest neighbours of the data-point at the center. In Figure (b) I show a class-based definition of nearest neighbours. Nearest neighbours are defined as those data-points sharing at least one class label in common.

compute the Euclidean distance at which each data-point has  $R$  nearest neighbours on average. The  $\epsilon$ -ball radius is then set to equal this average distance. The parameter  $R$  is set to 50 nearest neighbours in the literature (Kong et al. (2012); Kong and Li (2012a,b); Kulis and Darrell (2009); Raginsky and Lazebnik (2009); Gong and Lazebnik (2011)), and for compatibility I use the same setting throughout this thesis. The groundtruth matrix  $\mathbf{S} \in \{0, 1\}^{N_{trd} \times N_{trd}}$  is then derived by computing the Euclidean distance  $\mathbf{D} \in \mathbb{R}^{N_{trd} \times N_{trd}}$  between a small subset of the data-points ( $N_{trd} \ll N$ ) and thresholding the distances by  $\epsilon$ . This method of groundtruth generation is presented in Equation 3.1 and Figure 3.3a.

$$\mathbf{S} = \begin{cases} S_{ij} = 1, & \text{if } D_{ij} \leq \epsilon \\ S_{ij} = 0, & \text{if } D_{ij} > \epsilon \end{cases} \quad (3.1)$$

### 3.3.2 Class-Based Nearest Neighbours

Experimental results will be presented based on class labelled derived groundtruth (Figure 3.3b) in situations where the  $\epsilon$ -NN evaluation paradigm is not possible such as for cross-modal retrieval (it is not possible to directly compute the Euclidean distance between two feature vectors of a different type) or where I wish to present additional

results that can be directly compared to a specific portion of the literature that traditionally only uses a class-label based evaluation (e.g. the data-dependent supervised models discussed in Section 2.6.4). In this scenario  $S_{ij} = 1$  for an element of the groundtruth matrix  $\mathbf{S}$  if the corresponding pair of data-points  $\mathbf{x}_i, \mathbf{x}_j$  share *at least one* class label or annotation in common, and  $S_{ij} = 0$  otherwise. This is the same strategy used by most related research in the learning to hash literature (Gong and Lazebnik (2011); Liu et al. (2012)).

## 3.4 Evaluation Paradigms

There are two main paradigms for evaluating hashing models: the *Hamming ranking* evaluation paradigm (Section 3.4.1) and the *hashtable bucket* evaluation paradigm (Section 3.4.2). Both paradigms are illustrated in Figure 3.4. The Hamming ranking paradigm is standard within the learning to hash research literature while the hashtable bucket evaluation paradigm is frequently used in practical hashing applications in which a fast query-time is of prime importance. I introduce both paradigms in Sections 3.4.1-3.4.2 before explaining why I use the Hamming ranking evaluation paradigm exclusively throughout this thesis in Section 3.4.3.

### 3.4.1 Hamming Ranking Evaluation

In all the experiments in this thesis I will follow previous related research (Kong et al. (2012); Kong and Li (2012a,b); Liu et al. (2012, 2011); Gong and Lazebnik (2011); Zhang et al. (2010b); Kulis and Grauman (2009)) and evaluate retrieval effectiveness using the widely accepted *Hamming ranking evaluation* paradigm. In this evaluation paradigm, binary hashcodes are generated for both the query and the database images. The Hamming distance is then computed from the query images to all of the database images, with the database dataset images ranked in ascending order of the Hamming distance. The resulting ranked lists are then used to compute retrieval evaluation metrics such as area under the precision recall curve (AUPRC) (Section 3.6.3) and mean average precision (mAP) (Section 3.6.4). The Hamming ranking evaluation paradigm is a proxy for evaluating hashing accuracy over the range of user preferences (precision/recall) and without having to specify the parameters  $(K, L)$  of a specific hashtable implementation. I discuss this latter point further in Section 3.4.3.



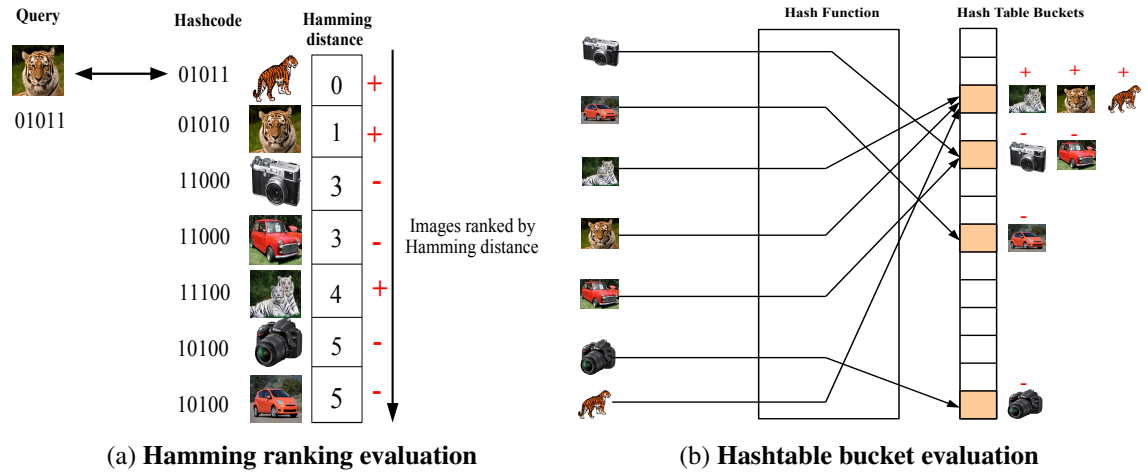


Figure 3.4: Two evaluation paradigms for hashing. In both cases relevant images are those depicting similar objects. In Figure (a) the Hamming distance between the query hashcode and the database images is computed. The images are ranked and the resulting ranked list used to compute a retrieval metric such as average precision (AP). In this case we find an  $AP = 0.87$ . The average precision scores are aggregated across queries by computing the mean average precision (mAP). In Figure (b) I show the hashtable evaluation strategy. Each image is hashed to a bucket. A count is then made of the number of true positives (TPs), false positives (FPs) and false negatives (FNs) colliding in the same buckets. In this toy example,  $TP = 3$ ,  $FP = 1$ ,  $FN = 2$  which equates to a micro-average  $F_1$ -measure of 0.78. On both diagrams + indicates a true positive while a - indicates a false positive/negative.

### 3.4.2 Hashtable Bucket-Based Evaluation

The Hamming ranking evaluation paradigm is by definition of  $O(N)$  time complexity for a single query data-point. The *hash bucket-based evaluation* has a constant  $O(1)$  search time independent of the dataset size. A hashtable lookup evaluation is much closer to how the hashing models would be used in a real-world application where a fast query time is a necessity. Despite this fact a hashtable evaluation is rarely reported in the learning to hash literature with the Hamming ranking paradigm being the preferred evaluation methodology. I previously described the application of hashtables in the context of Locality Sensitive Hashing (LSH) (Chapter 2, Section 2.4). In this evaluation paradigm the hashcodes are generated for the query and database points which are then used as the indices into the buckets of  $L$  hashtables. The union is then taken over all the data-points that collide in the same buckets as the query across the

$L$  hashtables. The set of data-points thus formed can then be used to compute the effectiveness metrics of precision, recall and  $F_\beta$ -measure. I describe these metrics in more detail in Section 3.6, but the intuition is that we want to reward the algorithm if it returns many true nearest neighbours to the query in the retrieved set while penalising it for returning unrelated data-points. As we examine all colliding data-points for a query we are therefore using the second hashtable query strategy which was discussed in the description of LSH in Chapter 2, Section 2.4.

### 3.4.3 Hamming Ranking versus Hashtable Bucket Evaluation

The hashtable bucket evaluation paradigm is heavily dependent on both the particular hashtable implementation (i.e. values of  $K$ ,  $L$ , whether or not chaining is used, etc) and on the end application itself, for example is the hashtable on a drone and therefore do we have limited available main memory? If I opt for a hashtable evaluation paradigm I either need to pick a default setting of  $K$  and  $L$  and tie my evaluation to this specific hashtable implementation, or alternatively I can measure the hashing model performance over many different values of the hashtable parameters leading to an explosion in the number of results to be reported. To abstract away from the specifics of a particular hashtable implementation and to obtain a single number summarising the quality of the hashcodes, researchers in the Computer Vision literature prefer to evaluate their hashing models by Hamming ranking which involves computing the *Hamming distance* between the hashcodes, rather than using a hashtable-based setup (Gong and Lazebnik (2011), Liu et al. (2011)). The Hamming distance given in Equation 2.4 (Chapter 2, Section 2.3) measures the number of bits that are different between two hashcodes and is therefore likely to be a good indicator of the quality of a hashtable lookup using those hashcodes. The more bits in common the greater the likelihood of a collision between the corresponding data-points.

There is an interesting, but not immediately obvious link between the Hamming ranking evaluation paradigm and the hashtable evaluation paradigm. Measuring the quality of a set of ranked lists using mAP and AUPRC is effectively acting as a *proxy* for *many* different settings of  $K$  and  $L$  in a corresponding hashtable evaluation. To confirm this fact, I make reference to Figure 3.5 in which I show five hashcodes ranked in ascending order of Hamming distance from the query (marked in the diagram in bold font). Observe that each threshold effectively defines a set of “colliding” data-points, that is those above the threshold with the lowest Hamming distance to the query. The

		Hash Table Configurations	
Ranked List ↓	Hamming Distance	<b>110111</b> Query	$K=3, L=2$ $K=6, L=1$
	0	110111	$K=2, L=3$ $K=1, L=6$
	1	110011	$K=3, L=2$ $K=1, L=6$
	2	100011 010011	$K=2, L=3$
	5	011000	$K=1, L=6$

Figure 3.5: An analogy between the Hamming ranking evaluation paradigm and the hashtable bucket evaluation paradigm. I rank five hashcodes in ascending order by Hamming distance from the query hashcode (shown here in bold). The ranked list is thresholded at *three* different points ( $t_1$ ,  $t_2$ ,  $t_3$ ), with the thresholds indicated by the dashed horizontal lines. The hashcodes above the threshold can be considered to be “colliding” with the query hashcode. The settings of  $K$  and  $L$  that would cause the collision are shown for the four different Hamming distances. For example, for the *bottom most* thresholding splitting the hashcodes into  $L = 6$  segments of  $K = 1$  bits will cause the hashcode at Hamming distance 5 to collide in the same bucket as the hashcodes at Hamming distance 2, 1 and 0. In this way we see that a particular thresholding of a ranked list of hashcodes is equivalent to many different settings of  $K$  and  $L$  in a hashtable evaluation.

thresholded ranked list therefore corresponds to settings of  $K$  and  $L$  that ensure the data-points above the threshold will collide in at least one hypothetical hashtable. The  $L$  hashtables in Figure 3.5 are formed by splitting the hashcodes into  $L$   $K$ -bit segments, with each  $K$ -bit segment indexing into a specific bucket of one of the  $L$  hashtables. Just as choosing a particular setting of  $K$  and  $L$  is application specific so is choosing a particular threshold in the Hamming ranking evaluation paradigm. Usefully the mAP and AUPRC provide a single number measure of ranking quality that is computed by aggregating across many different settings of the ranked list threshold, and consequently many different values of  $K$  and  $L$ . The Hamming ranking evaluation paradigm is therefore a more general evaluation strategy for hashing that is able to measure the overall quality of hashcodes without being tied to a particular end-application. In effect it indicates how good the hashing-based ANN search would be if we found the best setting of  $K$  and  $L$  in a hashtable bucket evaluation. Given this attractive advantage I accord with the relevant literature and follow the Hamming ranking evaluation strategy throughout

this thesis (Chapters 4-7). I leave a hashtable bucket evaluation to future work.

## 3.5 Constructing Random Dataset Splits

In this section I will describe how the datasets introduced in Section 3.2 are partitioned to form testing and validation queries and training and database splits for the purposes of learning and evaluating the hash functions. Two strategies for forming splits will be described: in Section 3.5.1 I describe the literature standard strategy that is widely used by the research community, while in Section 3.5.2 I describe my proposed splitting methodology that seeks to remedy concerns with the accepted evaluation strategy. In both cases, when class-based ground-truth is used, I sample the splits so as to obtain a balanced distribution of classes within each partition. This sampling strategy is discussed in more detail in Chapter 6, Section 6.3.

### 3.5.1 Literature Standard Splits

In previous work *repeated random subsampling* cross-validation over ten independent runs is used to evaluate the quality of the learnt hash functions (Liu et al. (2012); Kong et al. (2012); Kong and Li (2012a); Liu et al. (2014); Wang et al. (2012)). Figure 3.6 shows an example of a random dataset split for one run. The entire dataset is denoted as  $\mathbf{X} \in \mathbb{R}^{N \times D}$ . This dataset is divided into a held-out set of test queries  $\mathbf{X}_{teq} \in \mathbb{R}^{N_{teq} \times D}$  and a database split  $\mathbf{X}_{db} \in \mathbb{R}^{N_{db} \times D}$ . The test queries are used once when I come to compute the evaluation metric by ranking the database split (Section 3.4.1). The database split also doubles as the training dataset for learning the hash functions. The best setting of model *hyperparameters*<sup>7</sup> is found by grid search on the validation split of the dataset. In practice this grid search is conducted by running a set of validation queries  $\mathbf{X}_{vaq} \in \mathbb{R}^{N_{vaq} \times D}$  against a validation database  $\mathbf{X}_{vad} \in \mathbb{R}^{N_{vad} \times D}$ , both of which are sampled from the database  $\mathbf{X}_{db}$ . I am therefore using a form of nested cross-validation in which the optimal hyperparameters are determined for each run. The training database  $\mathbf{X}_{trd} \in \mathbb{R}^{N_{trd} \times D}$  is used to learn the *parameters* (hyperplanes, quantisation thresholds) of the hash functions and is itself a subset of  $\mathbf{X}_{db}$ . In the remainder of this dissertation I refer to this splitting strategy as the *literature standard splitting strategy*. I illustrate this method of forming dataset splits in Figure 3.6.

---

<sup>7</sup>Hyperparameters are parameters *other* than the hashing hyperplanes or quantisation thresholds. Example of hyperparameters are the flexibility of margin  $C$  for the SVM and the kernel bandwidth parameter  $\gamma$  for the RBF kernel.

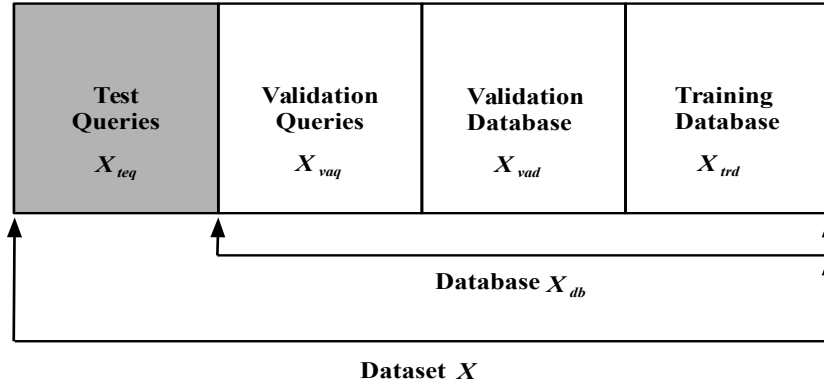


Figure 3.6: The literature standard dataset splitting procedure. The standard procedure used in the literature for splitting a dataset into testing and training partitions. The entire dataset  $\mathbf{X} \in \mathbb{R}^{N \times D}$  is represented as the concatenation of the individual rectangles, each of which highlights a particular partition. The rectangle in grey represents the split of the dataset that is held-out and only used once for computing the final measure of retrieval effectiveness.

### 3.5.2 Improved Splitting Strategy

Unfortunately, there is a potential overfitting concern with the standard dataset splitting strategy described in Section 3.5.1 given that the database points which are ranked or indexed with respect to the test queries are also used as the training dataset for learning the hash functions themselves. Ideally there should be a clean separation between the split of the dataset that is used to learn the hash functions and the split of the dataset that is ranked/indexed in order to compute the final measure of retrieval effectiveness. This ensures that I can evaluate the true generalisation performance of the hash functions when there is not only unseen queries but also an unseen database that is to be ranked/indexed with respect to those queries. Currently the literature is only concerned with the generalisation performance with respect to unseen query data-points and where the database is known a-priori and can be used for hash function learning. To the best of my knowledge I am the first in the literature to note this technical flaw in the standard method for forming dataset splits. To mitigate this overfitting concern I propose a new method for generating splits of the dataset. In this new strategy I again perform repeated random subsampling cross-validation over ten runs. However, the makeup of a random split for a run now differs from the literature standard splitting strategy. In my suggested dataset splitting strategy I divide the dataset into *five* splits as shown in Figure 3.7. I have a set of held-out test queries  $\mathbf{X}_{teq} \in \mathbb{R}^{N_{teq} \times D}$  and also a

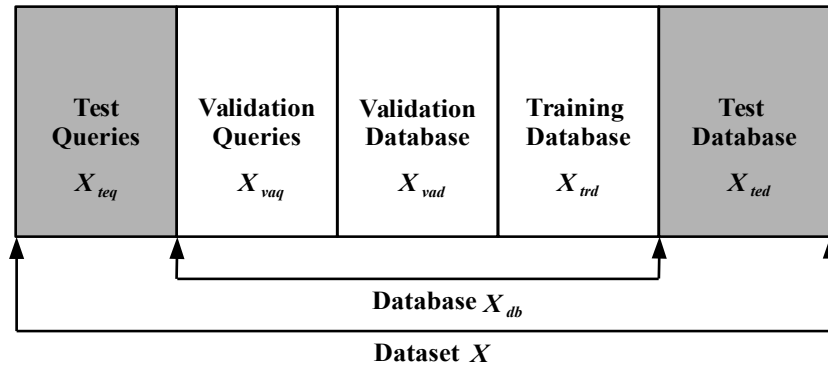


Figure 3.7: My improved dataset splitting procedure. My proposed splitting strategy for overcoming the overfitting concern with the literature standard strategy. In addition to the test queries I also advocate holding out a split of the dataset to act as the testing database. The held-out splits of the dataset are shown in grey. At test time the testing queries are used to retrieve related items from the testing database. This retrieval run is used to compute the final measure of effectiveness for determining the quality of the hash functions.

held-out test database  $\mathbf{X}_{ted} \in \mathbb{R}^{N_{ted} \times D}$  against which those test queries are run. Both of the test queries and test database are only used once when I come to compute the final retrieval effectiveness metric for that particular run. The remainder of the dataset forms the database split  $\mathbf{X}_{db} \in \mathbb{R}^{N_{db} \times D}$  which is used for setting the parameters and hyperparameters of the hashing models. The database split is further divided into a set of validation queries  $\mathbf{X}_{vaq} \in \mathbb{R}^{N_{vaq} \times D}$ , a validation database split  $\mathbf{X}_{vad} \in \mathbb{R}^{N_{vad} \times D}$  which is ranked/indexed against the validation queries and a training split that is used to learn the hash functions  $\mathbf{X}_{trd} \in \mathbb{R}^{N_{trd} \times D}$ . For the remainder of this dissertation I refer to this splitting strategy as the *improved splitting strategy*.

## 3.6 Evaluation Metrics

I follow previous research in the learning to hash literature and judge the retrieval effectiveness by the standard Information Retrieval (IR) metrics of *precision*, *recall*,  $F_\beta$ -measure (Section 3.6.1), *area under the precision recall curve (AUPRC)* (Section 3.6.3) and *mean average precision (mAP)* (Section 3.6.4).



### 3.6.1 Precision, Recall, $F_\beta$ -Measure

*Precision*, *recall* and their harmonic mean, the  $F_\beta$ -measure, are set-based evaluation metrics that can be used to ascertain the quality of an unranked collection of images. The retrieved set can be determined by looking into the colliding hashtable buckets for the Hashtable bucket evaluation or by defining a Hamming radius threshold for the Hamming ranking evaluation. In terms of the Hamming ranking evaluation, precision and recall can be computed by counting the number of true nearest neighbours that are within a fixed Hamming radius (true positives, TPs), the number of non nearest neighbours that are within a fixed Hamming radius (false positives, FPs) and the number of related data-points that are *not* are within a fixed Hamming radius to the query (false negatives, FNs).

More formally, I denote the groundtruth matrix as  $\mathbf{S} \in \{0, 1\}^{N_{trd} \times N_{trd}}$  (Sections 3.3.1-3.3.2). The groundtruth adjacency matrix specifies which data-points are *true nearest neighbour* pairs ( $S_{ij} = 1$ ) and which data-point pairs are unrelated ( $S_{ij} = 0$ ). As we discussed in Section 3.3, in the context of hashing-based ANN search, a data-point  $\mathbf{x}_j$  is denoted as a true nearest neighbour ( $S_{ij} = 1$ ) if it is within an  $\epsilon$ -ball of the query data-point  $\mathbf{q}_i$  or shares at least one class label in common with the query. Following a retrieval run, the ranked data-points within a certain Hamming radius ( $D$ ) of the query are those data-points considered to be related to the query, while those data-points outside of the Hamming radius  $D$  are considered to be unrelated<sup>8</sup>. The results of a ranked retrieval for a certain Hamming distance threshold  $D$  are represented by the square matrix  $\mathbf{R} \in \{0, 1\}^{N \times N}$  given in Equation 3.2.

$$R_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_j \text{ is within Hamming radius } D \text{ to the query } \mathbf{q}_i \\ 0, & \text{otherwise.} \end{cases} \quad (3.2)$$

Given the definitions of  $\mathbf{S}$  and  $\mathbf{R}$ , the number of *true positives* for a single query data-point  $\mathbf{q}_i$  is defined in Equation 3.3

$$TP(\mathbf{q}_i) = \sum_j S_{ij} \cdot R_{ij} \quad (3.3)$$

A *false negative* (FN) is a true nearest neighbour ( $S_{ij} = 1$ ) that is outside of the Hamming radius around the query  $\mathbf{q}_i \in \mathbb{R}^D$ . The total false negative count for the query is

<sup>8</sup>This is the Hamming ranking evaluation paradigm discussed in Section 3.4.1.

given in Equation 3.4

$$FN(\mathbf{q}_i) = \sum_j S_{ij} - TP(\mathbf{q}_i) \quad (3.4)$$

A *false positive (FP)* is a non-nearest neighbour ( $S_{ij} = 0$ ) that falls within the query Hamming radius  $\mathbf{q}_i \in \mathbb{R}^D$  (Equation 3.5)

$$FP(\mathbf{q}_i) = \sum_j (1 - S_{ij}) \cdot R_{ij} \quad (3.5)$$

Given Equations 3.3-3.5 the precision and recall metrics can then be defined as in Equations 3.6-3.7

$$P(\mathbf{q}_i) = \frac{TP(\mathbf{q}_i)}{TP(\mathbf{q}_i) + FP(\mathbf{q}_i)} \quad (3.6)$$

*Precision* is therefore the fraction of true nearest neighbours that are within the fixed Hamming radius out of all data-points that are within the fixed Hamming radius to the query data-point

$$R(\mathbf{q}_i) = \frac{TP(\mathbf{q}_i)}{TP(\mathbf{q}_i) + FN(\mathbf{q}_i)} \quad (3.7)$$

*Recall* is then the fraction of true nearest neighbours that are within the fixed Hamming radius to the query out of all possible true nearest neighbours for that query, regardless whether or not they are within the specified Hamming radius.

In a typical image retrieval experiment we have more than one query data-point. The question arises as to how we aggregate the precision and recall scores for all  $Q$  queries. There are effectively two ways which involve either taking a *micro-average* or a *macro-average*. To accord with the literature I am most interested in the *micro-average* in this thesis which would sum the TPs, FPs and FNs across all queries before computing the total precision and recall (Equations 3.8-3.9).

$$P_{micro} = \frac{\sum_{i=1}^Q TP(\mathbf{q}_i)}{\sum_{i=1}^Q TP(\mathbf{q}_i) + \sum_{i=1}^Q FP(\mathbf{q}_i)} \quad (3.8)$$

$$R_{micro} = \frac{\sum_{i=1}^Q TP(\mathbf{q}_i)}{\sum_{i=1}^Q TP(\mathbf{q}_i) + \sum_{i=1}^Q FN(\mathbf{q}_i)} \quad (3.9)$$

Finally, the weighted harmonic mean of recall and precision is known as the  $F_\beta$ -measure and is presented in Equation 3.10 (Rijsbergen (1979)):

$$\begin{aligned}
F_\beta &= \frac{(1 + \beta^2)P_{micro}R_{micro}}{\beta^2P_{micro} + R_{micro}} \\
&= \frac{(1 + \beta^2)TP_{micro}}{(1 + \beta^2)TP_{micro} + \beta^2FN_{micro} + FP_{micro}}
\end{aligned} \tag{3.10}$$

$F_\beta$ -measure can be used to combine precision and recall resulting from both a macro or micro-average. The free parameter  $\beta \in \mathbb{R}_+$  is used to adjust the contribution from the precision and recall. Setting  $\beta < 1$  in Equation 3.10 weights precision higher than recall, and vice-versa for a setting of  $\beta > 1$ . In most applications  $\beta$  is set to 1.0 giving the commonly used  $F_1$ -measure that provides an equal balance between the contribution of precision and recall to the final score. The greater the  $F_\beta$ -measure the more effective are the hash functions at returning true nearest neighbours in the same hashtable buckets.

### 3.6.2 Precision Recall Curve (PR Curve)

The precision and recall set-based evaluation metrics discussed in Section 3.6.1 are computed at a fixed operating point of the hashing algorithm. This operating point is usually derived from a particular parameter setting that is itself driven by user or system constraints. For example, in the context of a hashtable bucket evaluation paradigm (Section 3.4.2) this threshold could be implicitly defined by varying the number of hashtables  $L$  and the number of hashcode bits  $K$ . For the Hamming ranking evaluation paradigm (Section 3.4.1) the threshold is the radius of the Hamming ball around the queries. Database points with a Hamming distance to the query that puts them outside of the radius are not considered part of the retrieved set and therefore do not contribute to the computation of the precision and recall metrics. In contrast to the set-based evaluation metrics, the *precision-recall (PR) curve* measures the effectiveness of a ranked list of items across a range of different operating points. For the Hamming ranking evaluation paradigm, the PR curve is constructed by finding all the data-points within a certain Hamming radius  $D$  of the query set and computing the precision and recall over the corresponding retrieved set. By varying the Hamming radius from unity to the maximum Hamming radius  $D_{max}$  exhibited by database hashcodes we can trace out a PR curve using the resulting  $D_{max}$  precision-recall values. This curve depicts the trade-off between precision and recall as the Hamming radius from the queries is gradually increased. We expect that as the Hamming radius is increased the precision

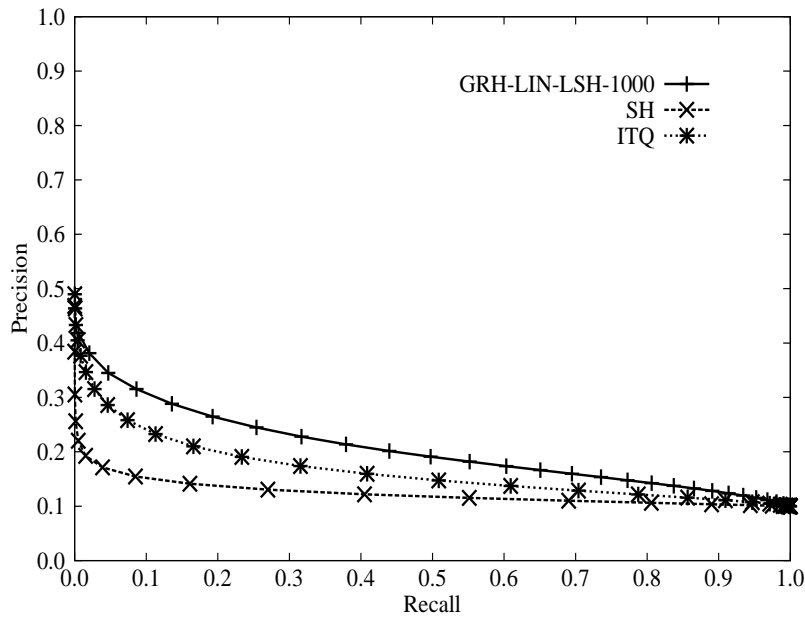


Figure 3.8: Precision recall curves for the CIFAR-10 dataset for a hashcode length of 32 bits. The three hashing algorithms indicated on this graph are studied in Chapter 6.

will drop (as more non-relevant data-points are encountered) while the recall will increase (as more relevant data-points are retrieved). An example precision-recall curve is presented in Figure 3.8.

### 3.6.3 Area Under the Precision Recall Curve

In many situations a single number summarising the ranking effectiveness captured by the precision-recall curve is required. Given its wide application in previously related research (Kong et al. (2012); Kong and Li (2012a,b); Moran et al. (2013a,b)) I settle for the *area under the precision-recall curve (AUPRC)* as the main single number effectiveness metric used consistently throughout this dissertation. The AUPRC is a real-valued number constrained to be within the limits of 0 and 1 and provides a summary of the retrieval effectiveness across all levels of recall. The computation of AUPRC is defined in Equation 3.11.

$$\begin{aligned}
 AUPRC &= \int_0^1 P(R) dR \\
 &= \sum_{d=1}^{D_{max}} P(d) \delta R(d)
 \end{aligned} \tag{3.11}$$

where  $P(R)$  denotes the micro precision at micro recall  $R$ ,  $P(d)$  is the precision at

Hamming radius  $d$  and  $\delta R(d)$  is the change in micro recall between Hamming radius  $d - 1$  and  $d$ <sup>9</sup>. The greater the area under the PR curve (AUPRC) the higher the retrieval effectiveness of the associated hashing model. The ideal PR curve has a precision of 1.0 across all recall levels leading to an AUPRC of 1.0.

### 3.6.4 Mean Average Precision (mAP)

Mean average precision (mAP) is also a commonly applied single-number evaluation metric for summarising the effectiveness of a ranking. However, in contrast to AUPRC which is directly computed from the precision-recall curve, mAP is calculated from the  $Q$  ranked lists that are obtained by computing the Hamming distance from every query data-point  $\{\mathbf{q}_i \in \mathbb{R}^D\}_{i=1}^Q$  to all the database data-points  $\{\mathbf{x}_j \in \mathbb{R}^D\}_{j=1}^N$ . Given a set of  $Q$  ranked lists, mAP is defined as follows (Wu et al. (2015)): denote as  $L$  the number of true nearest neighbours for query  $\mathbf{q}$  among the retrieved data-points,  $P_{\mathbf{q}}(r)$  as the precision for query data-point  $\mathbf{q}$  when the top  $r$  data-points are returned, and  $\delta(r)$  as an indicator function which returns ‘1’ when the  $r^{th}$  data-point is a true nearest neighbour of the query and ‘0’ otherwise. The average precision (AP) for a single query  $\mathbf{q}$  is then given in Equation 3.12 while the average of this quantity across all  $Q$  queries, the mean average precision or mAP, is defined in Equation 3.13.

$$AP(\mathbf{q}) = \frac{1}{L} \sum_{r=1}^R P_{\mathbf{q}}(r) \delta(r) \quad (3.12)$$

$$mAP = \frac{1}{|Q|} \sum_{i=1}^Q AP(\mathbf{q}_i) \quad (3.13)$$

Equation 3.12 computes the precision at each point when a new relevant image is retrieved. The average precision (AP) for a single query  $\mathbf{q}$  is then the mean of these precision values. The mAP is then computed by simply taking the mean of the average precisions across all  $Q$  queries (Equation 3.13). mAP is a real-valued number between 0.0 and 1.0, with a higher number indicating a more effective ranked retrieval and favours relevant images retrieved at higher (better) ranks. mAP is frequently used as a single-number evaluation metric in certain sub-fields of the learning to hash literature, particularly supervised and unsupervised data-dependent projection (Liu et al. (2011), Liu et al. (2012), Gong and Lazebnik (2011), Zhang et al. (2010b)). When comparing the contributions in this thesis to those particular sub-fields I will also report mAP in

<sup>9</sup>The finite sum representation for the AUPRC can be computed using the trapezoidal rule. This is implemented as the `trapz` function in Matlab.

addition to AUPRC so that my experimental results are directly comparable to previously published research.

### 3.6.5 Comparing and Contrasting AUPRC and mAP

The application of AUPRC and mAP as an evaluation metric is not consistent across the learning to hash literature, with some sub-fields (particularly binary quantisation) favouring AUPRC while others (such as data-dependent projection) appear to favour mAP. It is well-known that mAP is approximately the average of the AUPRC for a set of queries (Turpin and Scholer (2006)) so it is interesting to briefly consider here the retrieval scenarios where both metrics are expected to be in agreement and when they are likely to differ.

AUPRC is a *micro-average* in which the individual true positives, false positives and false negatives are aggregated across all  $Q$  queries for a specific threshold. The total aggregated counts are then used to compute the precision and recall for each possible setting of the threshold. The resulting precision and recall values can then be used to compute the AUPRC as given by Equation 3.11. In contrast the mAP is a *macro-average* which is found by computing the true positives, false positives and resulting precision per query, per relevant document retrieved and then averaging those precision values across all  $Q$  queries (Equations 3.12-3.13).

In practice, differences between the mAP and AUPRC will only arise in retrieval applications in which the distribution of relevant documents across queries is skewed. In this scenario the AUPRC will favour models that return more relevant documents from the queries with a larger number of relevant documents to the detriment of those queries that have a smaller number of relevant documents. In contrast the mAP will weight the contribution of every query equally even if many documents are relevant to some queries and very few to other queries. This equal weighting of queries ensures that mAP is insensitive to the performance variation between those queries that have many relevant documents and other queries that have very few relevant documents. To achieve a high mAP score the system must aim to do well across all queries and not just those with many relevant documents.

In a practical scenario, where the distribution of relevant documents per query is highly imbalanced, the choice of summarising the ranking effectiveness with either mAP or AUPRC is application specific (Sebastiani (2002)). In some cases we may be primarily interested in high effectiveness for the queries with a greater number of rel-



evant documents (AUPRC). This may be appropriate for evaluating system orientated tasks in which we wish to quantify how well the system does as a whole in returning pairs of true nearest neighbours (e.g. plagiarism detection). In other cases we may be equally interested in queries with a much smaller number of true positives (mAP). The latter scenario may arise in a user evaluation situation such as web search where the information retrieval system must not be seen to prioritise retrieval effectiveness for one user over another.

## 3.7 Summary

In this chapter I introduced the evaluation methodology that is commonly employed in the related research literature and which will be used to measure the effectiveness of my own contributions in this thesis. I began in Section 3.2 by outlining a collection of image and document datasets that will be used for my nearest neighbour (NN) search experiments. The datasets were divided into unimodal (image only) and cross-modal datasets (image-document), and were shown to encompass a large variability in the feature descriptors used to encode the images and documents, as well as the type of objects depicted in the images, their resolution and the total number of images (from 22,019 up to 1 million images) per dataset.

The definition of groundtruth is an important facet of any experimental methodology. In Section 3.3, I introduced two main strategies for judging the quality of a nearest neighbour search algorithm. The first strategy constructs a ball of radius  $\epsilon$  around a query and any data-points falling within that radius are deemed true nearest neighbours (Section 3.3.1). The second strategy sets true nearest neighbours to be those data-points that share at least one class label in common with the query (Section 3.3.2). The latter groundtruth definition is required for cross-modal retrieval experiments in which the feature descriptors occupy incommensurate feature spaces making an  $\epsilon$ -NN evaluation impractical.

In Section 3.4, I then defined the nearest neighbour search strategy to be used in evaluating the quality of the hashcodes. One natural option is to index the database and query images into hashtable buckets and count the number of true nearest neighbours that fall within the same buckets as the query (Section 3.4.2). Surprisingly I discussed how this *hashtable lookup* evaluation strategy is not at all common in the learning to hash literature. Instead most publications of note use what is termed the *Hamming ranking* evaluation paradigm where the Hamming distance is exhaustively computed

from the query to every data-point in the dataset (Section 3.4.1). The data-points are then ranked in ascending order of Hamming distance and the resulting ranked list is used to compute ranking-based evaluation metrics.

The next point I addressed in Section 3.5 was how to split the datasets into random partitions. In a retrieval setting I need a set of held-out test queries and a database over which retrieval will be performed. The accepted methodology in the literature (the *literature standard splitting strategy*) was to randomly select a set of held-out test queries and to use the remaining data-points as the database to be ranked *and* as the training dataset for learning the hash functions (Section 3.5.1). I identified a potential overfitting concern with this strategy and advocated an approach (the *improved splitting strategy*) where a certain split of the dataset forms a held-out database that cannot be used to learn the hash functions at training time (Section 3.5.2).

The final part of this chapter, in Section 3.6, introduced the evaluation metrics I will use to quantify the retrieval effectiveness of my algorithms with respect to prior art. In this thesis I use the standard Information Retrieval (IR) metrics of *area under the precision recall curve (AUPRC)* and *mean average precision (mAP)* to evaluate the quality of the hashcodes (Sections 3.6.3-3.6.4).

## 3.8 Conclusion

Having defined the research landscape within which this thesis is firmly embedded in Chapters 2-3 I am now in a position to introduce my own novel contributions to the field. I begin in Chapter 4 with a new multi-threshold quantisation algorithm that relaxes the limiting assumption of Single Bit Quantisation (SBQ) (Chapter 2, Section 2.5.1) in that only one threshold should be used for binarisation per projected dimension, and furthermore that the threshold position should remain unoptimised. My model assigns more than one threshold per dimension and dynamically optimises their positions based on the distribution of the input data, showing a significant increase in retrieval effectiveness versus a host of state-of-the-art quantisation models.