Chapter 7

Learning Hypersurfaces and Quantisation Thresholds

The research presented in this Chapter has been previously published in Moran (2016).

7.1 Introduction

In Chapter 1 I motivated this thesis by arguing how both steps of the hashcode generation pipeline, namely projection and quantisation, were readily amenable to improvement through data-driven learning of the hashing hypersurfaces and quantisation thresholds. In Chapters 4-5 I focused my attention on improving the quantisation step of the hashcode generation pipeline. I found that optimising the position of multiple thresholds per projected dimension was more effective than placing a single threshold by default at zero for mean centered data. In Chapter 6 I improved the projection step by learning hashing hypersurfaces that were adapted to the distribution of the data. In this latter case I found that learnt hypersurfaces fractured the input feature space in a way that resulted in more true nearest neighbours falling within the same partitioned regions compared to a purely random partitioning. In both cases I was encouraged to find statistically significant increases in image retrieval effectiveness compared to existing models that set both parameters independent of the data distribution. In this chapter, I bring together and consolidate the main contributions of this dissertation by performing two experiments. Firstly, in Section 7.2, I combine the multi-threshold quantisation models of Chapters 4-5 with the graph regularised projection function of Chapter 6 to form a hashing model that learns both the hashing hypersurfaces and multiple quantisation thresholds per projected dimension. I measure the performance of the combined model with respect to baselines that learn either parameter in isolation. Secondly, in Section 7.3.3.2, I then appeal to the Computer Vision practitioner by conducting an experiment which seeks to find the most effective combination of the novel models introduced in this thesis for the task of image retrieval.

7.2 Learning Hypersurfaces and Thresholds

7.2.1 Problem Definition

In this chapter I depart from my earlier contributions by learning both the quantisation thresholds and the hashing hypersurfaces *in the same model* so that neighbouring points $\mathbf{x}_i \in \mathbb{R}^D, \mathbf{x}_j \in \mathbb{R}^D$ are more likely to have similar hashcodes $\mathbf{b}_i \in \{-1,1\}^K, \mathbf{b}_j \in \{-1,1\}^K$.

7.2.2 Overview of the Approach

To achieve the learning objective outlined in Section 7.2.1. I take the most straightforward pipeline-based approach in this chapter. Specifically I couple both models by quantising the projections generated by my supervised projection function introduced in Chapter 6 using the multi-threshold quantisation models of Chapters 4-5. The objective is to learn a set of K linear hypersurfaces[1] $\left\{\mathbf{w}_k \in \mathbb{R}^D\right\}_{k=1}^K$ and a set of T quantisation thresholds $\mathbf{t}_k = [t_{k1}, t_{k2}, \ldots, t_{kT}]$ where $t_k \in \mathbb{R}$ and $t_{k1} < t_{k2} \ldots < t_{kT}$ for each of the K projected dimensions $\left\{\mathbf{y}^k \in \mathbb{R}^{N_{trd}}\right\}_{k=1}^K$ produced by those hypersurfaces. The K hyperplanes are learnt using Algorithm 9 which was presented on page 203 in Chapter 6 while the T thresholds are optimised by maximising Equation 4.6 presented on page 122 in Chapter 4. Equation 4.6 is maximised using Evolutionary Algorithms which were found to be the best performing method of stochastic search for multiple threshold optimisation in Chapter 4 Section 4.3.3.3.

More concretely the model is firstly run for M iterations, in which the following three steps (A-C) are applied during each iteration:

- **A) Regularisation:** Regularise the hashcodes using Equation 7.1:

$$\mathbf{B}_m \leftarrow \operatorname{sgn}\left(\alpha\, \mathbf{S}\mathbf{D}^{-1}\mathbf{B}_{m-1} + (1-\alpha)\mathbf{B}_0\right) \tag{7.1}$$

[1] I leave exploration of non-linear hypersurfaces induced by the radial basis function (RBF) kernel to future work.

$\mathbf{B}_m \in \{-1,1\}^{N_{trd}\times K}$ are the hashcodes for the N_{trd} training data-points at iteration m, the bits in B_0 are initialised using any existing projection function such as LSH or ITQ+CCA, $\alpha \in [0,1]$ is a scalar interpolation parameter, $\mathbf{S} \in \{0,1\}^{N_{trd}\times N_{trd}}$ is the adjacency matrix, $\mathbf{D}$ is a diagonal matrix containing the degree of each node in the graph.

- **B) Partitioning:** Solve the following K constrained optimisation problems in Equation 7.2:

$$\begin{aligned} \text{for } k = 1\ldots K: \quad & \min \; ||\mathbf{w}_k||^2 + C\textstyle\sum_{i=1}^{N_{trd}} \xi_{ik} \\ \text{s.t.} \quad & B_{ik}(\mathbf{w}_k^\intercal \mathbf{x}_i) \geq 1 - \xi_{ik} \qquad \text{for } i = 1\ldots N_{trd} \end{aligned} \tag{7.2}$$

  $\mathbf{w}_k \in \mathbb{R}^D$ is the hyperplane normal vector, $\xi_{ik} \in \mathbb{R}_+$ are slack variables that allow some points $\mathbf{x}_i$ to fall on the wrong side of the hyperplane $\mathbf{h}_k$ and $C \in \mathbb{R}_+$ is the flexibility of margin.

- **C) Prediction:** Solve for the K projected dimensions $\{\mathbf{y}^k \in \mathbb{R}^{N_{trd}}\}_{k=1}^{K}$ by computing the following N_{trd} dot products in Equation 7.3:

$$y_i^k = \mathbf{w}_k^\intercal \mathbf{x}_i \;\text{ for } i{=}\{1\ldots N_{trd}\} \text{ and } k{=}\{1\ldots K\} \tag{7.3}$$

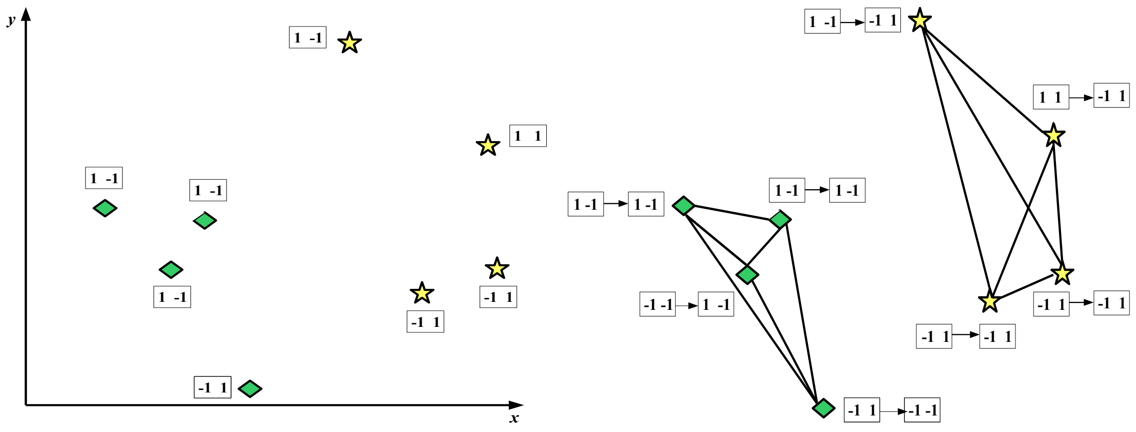
  Standard single bit quantisation (SBQ) is used to generate the updated hashcode matrix $\mathbf{B}_m$. Steps A-C are then repeated for M iterations, which is simply the standard GRH algorithm outlined in Chapter 6, Section 6.2.

Having learnt the projections $\{\mathbf{y}^k \in \mathbb{R}^{N_{trd}}\}_{k=1}^{K}$ of the N_{trd} data-points in Steps A-C, we then quantise the projections by performing a multi-threshold quantisation using NPQ (Chapter 4) in Step D:

- **D) Quantisation:** Quantise the K projected dimensions $\{\mathbf{y}^k \in \mathbb{R}^{N_{trd}}\}_{k=1}^{K}$ with thresholds $\mathbf{t}_k = [t_{k1}, t_{k2}, \ldots, t_{kT}]$ learnt by optimising $\mathcal{J}_{npq}(\mathbf{t}_k)$ in Equation 7.4:

$$\mathcal{J}_{npq}(\mathbf{t}_k) = \hat{\alpha} F_1(\mathbf{t}_k) + (1-\hat{\alpha})(1-\Omega(\mathbf{t}_k)) \tag{7.4}$$

  where $\hat{\alpha} \in [0,1]$ is a scalar interpolation parameter. The resulting N_{trd} bits for projected dimension k are placed in the *k-th* column of matrix $\mathbf{B}_m$ and the procedure repeated from Step A. Here $T \in [1,3,7,15]$ is the number of thresholds per projected dimension and $F_1(\mathbf{t}_k)$ is a supervised term that leverages the neighbourhood structure encoded in $\mathbf{S}$. More specifically, for a fixed set of



(a) **LSH initialisation** (b) **Regularisation step**

Figure 7.1: In Figure (a) I show a toy 2D space in which I have initialised 2-bit hashcodes for the data-points with LSH. In Figure (b) I show Step A of the model in which the hashcodes are regularised over the adjacency matrix. Lines between points indicate a nearest neighbour relationship. The hashcodes are updated as shown by the directed arrows.

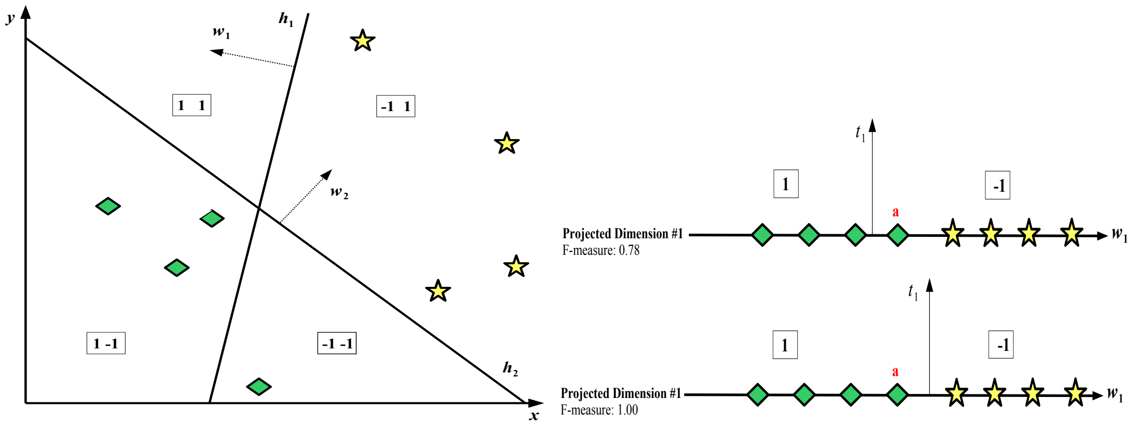
thresholds $\mathbf{t}_k = [t_{k1} \dots t_{kT}]$, I define a per-projected dimension indicator matrix $\mathbf{P}^k \in \{0,1\}^{N_{trd} \times N_{trd}}$ with the property given in Equation 7.5

$$P^k_{ij} = \begin{cases} 1, & \text{if} \quad \exists_\gamma \quad s.t. \quad t_{k\gamma} \leq (y^k_i, y^k_j) < t_{k(\gamma+1)} \\ 0, & \text{otherwise.} \end{cases} \tag{7.5}$$

where the index $\gamma \in \mathbb{Z}$ spans the range: $0 \leq \gamma \leq T$, and the scalar quantity T denotes the total number of thresholds partitioning a given projected dimension. Intuitively, matrix $\mathbf{P}^k$ indicates whether or not the projections (y^k_i, y^k_j) of any pair of data-points $(\mathbf{x}_i, \mathbf{x}_j)$ fall within the same thresholded region of the projected dimension $\mathbf{y}^k \in \mathbb{R}^{N_{trd}}$. Given a particular instantiation of the thresholds $[t_{k1} \dots t_{kT}]$, the algorithm counts the number of *true positives* (TP), *false negatives* (FN) and *false positives* (FP) across all regions (Equations 7.6-7.8).

$$TP = \frac{1}{2}\sum_{ij} P_{ij} S_{ij} = \frac{1}{2}\|\mathbf{P} \circ \mathbf{S}\|_1 \tag{7.6}$$

$$FN = \frac{1}{2}\sum_{ij} S_{ij} - TP = \frac{1}{2}\|\mathbf{S}\|_1 - TP \tag{7.7}$$



(a) **Partitioning step** (b) **Quantisation step**

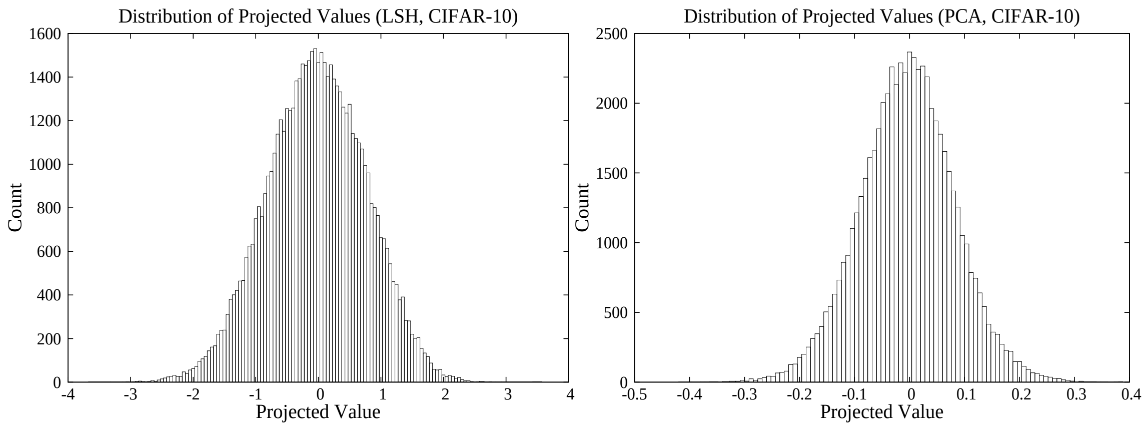
Figure 7.2: In Figure (a) I show the partitioning step (Step B) where linear hypersurfaces are positioned to separate opposing bits (-1,1) with maximum margin. In Figure (b) right I show Step D using the projections for hyperplane $\mathbf{h}_1$ as an example. The top-most diagram shows the quantisation obtained with a threshold placed at zero. Point a is on the wrong side of the threshold. The bottom diagram shows the result obtained by optimising the threshold t_1 to maximise pairwise F_1. In this case the threshold is shifted so that point a receives the same bits as its neighbours. NPQ therefore corrects quantisation boundary errors committed by GRH: this is the crux of the contribution in this chapter.

$$FP = \frac{1}{2}\sum_{ij} P_{ij} - TP = \frac{1}{2}\|\mathbf{P}\|_1 - TP \tag{7.8}$$

where $\circ$ denotes the Hadamard (elementwise) product and $\|.\|_1$ is the L_1 matrix norm defined as $\|\mathbf{X}\|_1 = \sum_{ij}|X_{ij}|$. Intuitively TP is the number of positive pairs that are found within the same thresholded region, FP is the proportion of negative pairs found within the same region, and FN are the proportions of positive pairs found in different regions. The TP, FP and FN counts are combined using the familiar set-based F_1-measure (Equation 7.9):

$$F_1(\mathbf{t}_k) = \frac{2\|\mathbf{P}\circ\mathbf{S}\|_1}{\|\mathbf{S}\|_1 + \|\mathbf{P}\|_1} \tag{7.9}$$

Intuitively maximisation of Equation 7.9 encourages a clustering of the projected dimension so that as many of the must-link and cannot-link constraints encoded in the adjacency matrix $\mathbf{S}$ are respected. $\mathcal{J}_{npq}$ is the optimisation objective of



(a) **Locality Sensitive Hashing (LSH)** (b) **Principal Component Analysis (PCA)**

Figure 7.3: Distribution of projected values for two randomly chosen GRH projected dimensions on the CIFAR-10 dataset. In Figure (a) GRH was initialised with LSH hashcodes while in Figure (b) I initialised GRH with hashcodes computed by PCA. In both cases the region of highest projected value density is around zero.

my multi-threshold quantisation algorithm which is defined in more detail in Chapter 4, Section 4.2.2.

In Figures 7.1-7.2, I provide an intuitive overview of the algorithm with an example two-dimensional example problem.

The question arises as to why we might expect learning the hypersurfaces and thresholds within the same model to lead to enhanced retrieval effectiveness. In Figure 7.3 I show projected value histograms created by my graph regularised projection function (GRH). It is clear that GRH projections tend to cluster around zero along each projected dimension. Following a similar argument as for LSH and PCA projections in Chapter 2 Section 2.5.1, I argue that placing a single threshold directly at zero along a GRH projected dimension can be sub-optimal given that it may separate out many true nearest neighbours on opposite sides of the threshold as this is the region of highest point density. In this chapter, I apply the experimental findings of Chapter 4 and hypothesise that optimising the position of multiple thresholds based on the distribution of the data can mitigate this issue with quantisation boundary errors and boost the retrieval effectiveness of GRH. To study this research question I therefore combine GRH with the threshold learning algorithms presented in Chapters 4-5 (NPQ, VBQ) and compare the resulting model to competitive baselines that learn either parameter individually. I present my experimental results in Section 7.3.

7.3 Experimental Evaluation

7.3.1 Experimental Configuration

The experiments in this section are directed towards answering the following single hypothesis:

- H_1: *Learning the hashing hypersurfaces and multiple quantisation thresholds as part of the same model can give a higher retrieval effectiveness than a model which learns either the hypersurfaces or thresholds.*

To answer this hypothesis I use an identical experimental setup to that employed in Chapters 4-5. This experimental configuration is standard in the relevant literature (Kong et al. (2012); Kong and Li (2012a,b); Kulis and Darrell (2009); Raginsky and Lazebnik (2009); Gong and Lazebnik (2011)) and is shown in Table 7.1 for reading convenience. Concretely I define the groundtruth nearest neighbours using the ε-NN paradigm, that is if a data-point is within a radius of ε to the query then it is deemed to be true nearest neighbour for the purposes of evaluation as explained in Chapter 3, Section 3.3.1. Retrieval effectiveness is computed using the standard Hamming ranking evaluation paradigm (Chapter 3, Section 3.4.1) and the area under the precision recall curve (AUPRC) (Chapter 3, Section 3.6.3). Note that I am using a different groundtruth definition (ε-NN) and main evaluation metric (AUPRC) compared to Chapter 6. This means that the quantitative results in this Chapter and Chapter 6 are mostly not directly comparable.

I test the model on the standard CIFAR-10 dataset introduced in Chapter 3, Section 3.2, and leave examination of the performance on other image datasets to future work. To define the test queries ($\mathbf{X}_{teq} \in \mathbb{R}^{N_{teq} \times D}$) I randomly sample $N_{teq} = 1,000$ data points with the remaining points forming the database ($\mathbf{X}_{db} \in \mathbb{R}^{N_{db} \times D}$). The precise dataset split is shown in Table 7.2. In all experiments the hypersurfaces and thresholds are learnt on the training dataset ($\mathbf{X}_{trd} \in \mathbb{R}^{N_{trd} \times D}$). These hypersurfaces and thresholds are then used to quantise the test dataset projections ($\mathbf{X}_{teq} \in \mathbb{R}^{N_{teq} \times D}$). The reported AUPRC figures are computed using repeated random sub-sampling cross-validation over ten independent runs. The Wilcoxon signed rank test (Smucker et al. (2007)) is used for measuring statistical significance. The unit of the significance test is a pair of AUPRC values pertaining to the two systems under comparison and resulting from a particular random split of the dataset. In all presented result tables the symbol

Parameter	**Setting**	**Chapter Reference**
Groundtruth Definition	ϵ-NN	Chapter 3, Section 3.3.1
Evaluation Metric	AUPRC	Chapter 3, Section 3.6.3
Evaluation Paradigm	Hamming Ranking	Chapter 3, Section 3.4.1
Random Partitions	10	Chapter 3, Section 3.5
Number of Bits (K)	16-128	Chapter 2, Section 2.4

Table 7.1: Configuration of the main experimental parameters for the results presented in this chapter.

Partition	**CIFAR-10**
Test queries (N_{teq})	1,000
Validation queries (N_{vaq})	1,000
Validation database (N_{vad})	10,000
Training database (N_{trd})	2,000
Test database (N_{ted})	59,000

Table 7.2: Literature standard splitting strategy partition sizes for the experiments in the chapter. This breakdown is based on the splitting strategy introduced in Chapter 3, Section 3.5.

▲▲/▼▼ indicates a statistically significant increase/decrease with $p < 0.01$, while ▲/▼ indicates a statistically significant increase/decrease with $p < 0.05$.

7.3.2 Parameter Optimisation

Several hyperparameters need to be set for the graph regularised projection function (GRH) and the multiple threshold quantisation model (NPQ). For NPQ I use evolutionary algorithms with the number of individuals H set to 15 and the number of iterations M set to 15 as was found to be optimal in Chapter 4, Section 4.3.3.3. The NPQ interpolation parameter α is set to $\alpha = 1$ and I use three thresholds ($T = 3$) per projected dimension and the Manhattan hashing codebook and hashcode ranking strategy (Chapter 2, Section 2.5.4). For GRH, I set the number of iterations $M \in \mathbb{Z}_+$, the interpolation parameter $\alpha \in [0,1]$ and the flexibility of margin $C \in \mathbb{R}_+$ for the SVM by closely following the parameter setting strategy outlined in Chapter 6, Section 6.3.2. Specifically the parameters are set on the held-out validation dataset $\mathbf{X}_{vaq}, \mathbf{X}_{vad}$. I set $C = 1$ and perform a grid search over $M \in \{1 \dots 20\}$ and $\alpha \in \{0.1, \dots, 0.9, 1.0\}$, selecting the setting that gives the highest validation AUPRC. To find M, I stop the sweep when the validation dataset AUPRC falls for the first time, and set M to be the number of the penultimate iteration. Finally M and α are then held constant at their optimised values, and $C \in \{0.01, 0.1, 1, 10, 100\}$. I equally weigh both classes (-1 and 1) in the SVM.

7.3.3 Experimental Results

7.3.3.1 Experiment I: Learning Hashing Hypersurfaces and Multiple Quantisation Thresholds

In this experiment I seek to answer hypothesis H_1 as to whether combining the multiple threshold quantisation algorithm from Chapter 4 with the graph regularised projection function developed in Chapter 6 can be more effective than model either algorithm in isolation. To answer this hypothesis I will follow the experimental setup in Chapter 4 and treat the graph regularised projection function (GRH) as a method for *improving* the projections of existing data-dependent and independent projection functions such as LSH and PCA. I will therefore take the hashcodes produced by the existing projection functions of LSH, PCA, ITQ, SH and SKLSH and use those bits as the initialisation point for GRH in the training hashcode matrix $\mathbf{B}_0$[2]. The results for the retrieval experiments on the CIFAR-10 image collection are shown in Table 7.3, Figures 7.4 (a)-(b) and Figures 7.5 (a)-(b).

In Table 7.3 it is clear that, apart from an ITQ initialisation, learning the hypersurfaces and thresholds as part of the same combined model (GRH+NPQ) yields the highest retrieval effectiveness compared to learning the hypersurfaces (GRH+SBQ) or the thresholds (NPQ, VBQ) independently, as I did in Chapter 6 and Chapters 4-5, respectively[3]. For example, for LSH projections GRH+NPQ gains a relative increase in AUPRC of 60% over NPQ and 28% over GRH+SBQ. Furthermore, the combination of GRH+NPQ outperforms an adaptive threshold allocation (VBQ) by a relative margin of 27%. Each of these increases are found to be statistically significant using a Wilcoxon signed rank test (p-value < 0.01). I am encouraged to find that the superior retrieval effectiveness of GRH+NPQ is maintained when the hashcode length is varied between 16-128 bits for both LSH, PCA, SKLSH and SH projections (Figure 7.4 (a)-(b), Figure 7.5 (a)-(b)). Based on these experimental results I confirm my primary

[2]Please refer to Chapter 2 Sections 2.4, 2.6.2 and Section 2.6.3 for an overview of LSH, PCA, ITQ, SH and SKLSH

[3]I compare to the binary integer linear programme (BILP) variant of VBQ (VBQ_{bilp}) in this Chapter.

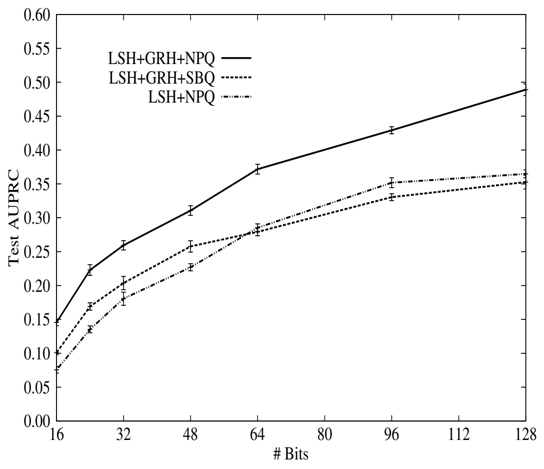
	Model					
	SBQ	**NPQ**	**GRH+SBQ**	**VBQ**	**GRH+NPQ**	**GRH+VBQ**
LSH	0.0954	0.1621	0.2023	0.2035	**0.2593▲▲**	0.2420
ITQ	0.2669	**0.3917▲▲**	0.2558	0.3383	0.3670	0.3185
SH	0.0626	0.1834	0.2147	0.2380	0.2958	**0.3003▲▲**
PCA	0.0387	0.1660	0.2186	0.2579	0.2791	**0.2897▲▲**
SKLSH	0.0513	0.1063	0.1652	0.2122	**0.2566▲▲**	0.2334

Table 7.3: AUPRC on the CIFAR-10 dataset with a hashcode length of 32 bits. The quantisation algorithms listed on the first row are used to quantise the projections from the hash functions in the first column. ▲▲ indicates statistical significance (Wilcoxon signed rank test, $p < 0.01$) when comparing NPQ and GRH+NPQ (or GRH+VBQ).

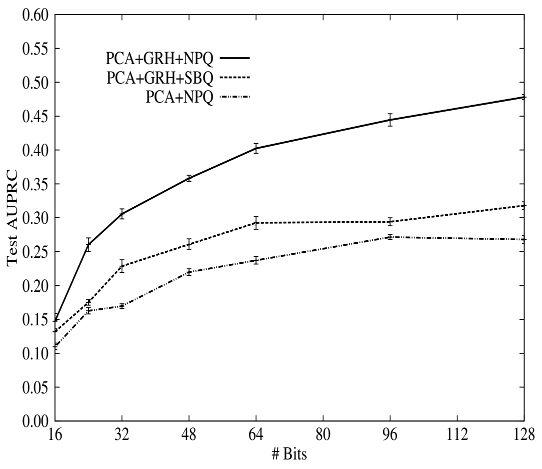
hypothesis (H_1) that learning of the hypersurfaces and thresholds in the same combined model can outperform a model which learns either the thresholds or the hypersurfaces, but not both.

The fact that the retrieval effectiveness for PCA+GRH+NPQ does not tail off in Figure 7.4 (b) as the hashcode length increases is a particularly significant finding. In Chapters 4-5 I observed that PCA projections quantised with *multiple* thresholds per projected dimension tend to approach a plateau in AUPRC with increasing hashcode length. This observation is related to the more unreliable PCA hyperplanes with low eigenvalue which tend to capture very little variance in the input feature space. Using the normal vectors of those hyperplanes to generate hashcode bits is therefore likely to add little to the overall retrieval effectiveness. In contrast, refining PCA hyperplanes with NPQ and GRH appears to entirely overcome this issue, yielding a retrieval effectiveness that continues to increase past a hashcode length of $K = 128$ bits. PCA+GRH+NPQ is therefore an effective means of overcoming the imbalanced variance problem associated with PCA hyperplanes, an issue that has attracted significant research effort in the learning to hash community (Weiss et al. (2008), Kong and Li (2012b), Gong and Lazebnik (2011)).

In Chapter 5 I devised an algorithm (VBQ) that intelligently picked the most discriminative subset of hyperplanes while simultaneously deciding how many thresholds to allocate to those hyperplanes. The pertinent question in this Chapter is whether there is an additive benefit obtainable from refining the PCA, LSH, SH and SKLSH hyperplanes with GRH and then subsequently quantising those GRH projections with VBQ.



(a) **LSH projections**

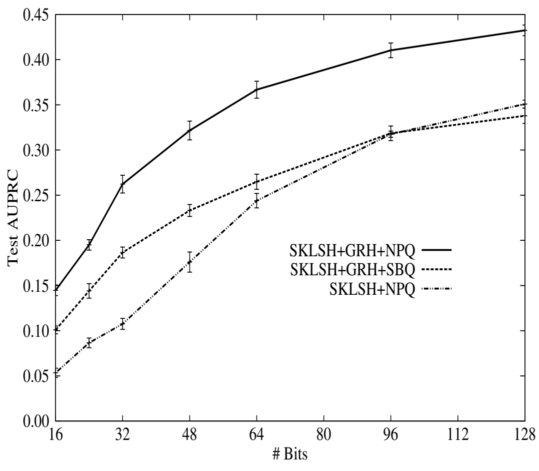


(b) **PCA projections**

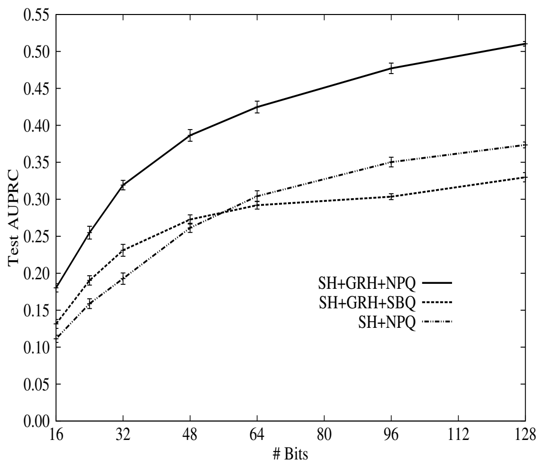
Figure 7.4: Learning the hashing hypersurfaces and quantisation thresholds jointly (GRH+NPQ) versus independently (GRH+SBQ, SBQ) over a haschode length of 16-128 bits. Results are shown for LSH projections (Figure (a)) and PCA projections (Figure (b)).

The results in Table 7.3 confirm that an additive benefit *does* exist for these projections, namely GRH+VBQ has a significantly higher effectiveness than either VBQ or GRH+SBQ alone when quantising PCA, LSH, SH and SKLSH projections. Nevertheless, I also observe the result that for these four projection functions GRH+VBQ *does not* significantly outperform GRH+NPQ. This latter result suggests that both VBQ and NPQ are *equivalently effective* in quantising the GRH refined projections and no further boost in effectiveness is possible by assigning different number of thresholds per projected dimension. The end user can therefore save computation time by using $K/2$ of the K available hyperplanes and quantising all $K/2$ with a uniform three thresholds per projected dimension, rather than rely on VBQ to find a good data-driven allocation.

The retrieval results obtained when ITQ provides the initialisation of the hyperplanes suggest a somewhat different story. Firstly I observe from Table 7.3 that there is no significant difference between the AUPRC from simply quantising ITQ projections directly with single bit quantisation (SBQ) (0.2669) and further refining the hyperplanes with GRH (0.2558). This result therefore strongly suggests that the ITQ hyperplanes are already providing a very good partitioning of the input feature space under the ϵ-NN groundtruth definition, with GRH being unable to better that positioning as it readily does for LSH, SH, PCA and SKLSH projection functions. However quantising the projections from the *GRH refined* ITQ hyperplanes with both NPQ



(a) **SKLSH projections**



(b) **SH projections**

Figure 7.5: Learning the hashing hypersurfaces and quantisation thresholds jointly (GRH+NPQ) versus independently (GRH+SBQ, SBQ) over a haschode length of 16-128 bits. Results are shown for SKLSH projections (Figure (a)) and SH projections (Figure (b)).

(GRH+NPQ) and VBQ (GRH+VBQ) gives significantly lower retrieval effectiveness (0.3670, 0.3185) than simply taking the ITQ hyperplanes directly and quantising their projections with NPQ (0.3917). Using both GRH and NPQ or VBQ in the same model therefore does *not* give an additive benefit in retrieval effectiveness as it does for the LSH, SKLSH, PCA and SH projection functions. This finding suggests that ITQ+NPQ is the best model combination and leads to the highest image retrieval effectiveness for the literature standard groundtruth definition (ϵ-NN) and Hamming ranking evaluation metric (AUPRC). I explore this result further in Section 7.3.3.2.

7.3.3.2 Experiment II: Finding the Model Combination with Highest Retrieval Effectiveness

In my final experiment of this chapter I will compare the effectiveness of the best combination of models resulting from the novel contributions made in this thesis to the highest performing baselines from the literature. Table 7.4 presents the retrieval results on the CIFAR-10 dataset. I select ITQ+NPQ as my best model which quantises ITQ projections with the multi-threshold quantisation algorithm (NPQ). NPQ is parameterised with $T = 3$ thresholds per projected dimensions and is configured to use the Manhattan Quantisation binary codebook and Manhattan distance hashcode ranking strategy. This combination was found to give the overall highest retrieval effectiveness out of all the model combinations I considered in preliminary experiments. As baselines for comparison I select two of the strongest performing models from the relevant literature. In previous chapters I found Iterative Quantisation (ITQ) (Gong and Lazebnik (2011)) to give high retrieval effectiveness, in addition to the Supervised Hashing with Kernels (KSH) model of Liu et al. (2012). I observe in Table 7.4 that ITQ+NPQ significantly (Wilcoxon signed rank test, $p < 0.01$) outperforms the strong non-linear baseline model of KSH, while also substantially outperforming the other baseline of ITQ on the task of image retrieval. Combining my proposed quantisation model (NPQ) with ITQ therefore leads to a new state-of-the-art retrieval performance on the standard CIFAR-10 image dataset.

	16	**32**	**48**	**64**	**128**
ITQ+NPQ	**0.2439▲▲**	**0.3991▲▲**	**0.4567▲▲**	**0.5019▲▲**	**0.5501▲▲**
KSH	0.1654	0.2878	0.3511	0.3820	0.4397
ITQ	0.1734	0.2638	0.3072	0.3382	0.3646

Table 7.4: The best model combination resulting from research in this thesis ITQ+NPQ compared against the models with the highest retrieval effectiveness from the learning to hash literature (ITQ, KSH). ▲▲ indicates statistical significance (Wilcoxon signed rank test, $p < 0.01$) versus KSH.

7.4 Conclusions

In this chapter I have consolidated the novel contributions made throughout this thesis from two angles. Firstly I conducted an initial investigation into coupling the hypersurface and threshold learning algorithms developed in Chapters 4-6. The projections formed by my graph regularised projection function (GRH) were quantised with my multi-threshold quantisation models (NPQ, VBQ). My findings resulting from these model combinations were:

- Learning the hashing hypersurfaces and quantisation thresholds in the same hashing model (GRH+NPQ) can give a retrieval effectiveness higher than a model which learns either the hashing hypersurfaces (GRH) or the quantisation thresholds (NPQ, VBQ). Quantitative results supporting this claim are presented in Section 7.3.3.1, Table 7.3.

In my experiments I treated the GRH and NPQ models as methods for refining the hypersurfaces of existing projection functions such as LSH or PCA. With this view in mind my experimental evaluation found that the GRH+NPQ model combination was shown to be most beneficial for a broad selection of data-independent and dependent projection functions other than the Iterative Quantisation (ITQ) model of Gong and Lazebnik (2011).

I then consolidated my research from a second angle that involved finding the overall best performing model combination that leveraged one or more of the ideas presented throughout this dissertation. I made the following experimental finding in this regard:

- The model combination ITQ+NPQ significantly outperforms a set of strong baseline models from the learning-to-hash literature yielding state-of-the-art retrieval effectiveness. This claim is supported by the results in Section 7.3.3.2, Table 7.4.

For the Computer Vision practitioner, I would therefore advocate the use of the ITQ+NPQ model combination for image retrieval. I found this model combination to be simple to engineer, fast to run in both training and hashcode prediction time for the relatively low-dimensional image datasets considered in this thesis while also exhibiting state-of-the-art image retrieval effectiveness in my experimental evaluation.

In Chapter 8 I conclude this dissertation by providing a summary of my main research findings, alongside several pointers to potentially fruitful avenues for future work in the field.