Generative Adversarial Product Quantisation

Litao Yu
Griffith University
litao.yu@griffith.edu.au

Yongsheng Gao
Griffith University
yongsheng.gao@griffith.edu.au

Jun Zhou
Griffith University
jun.zhou@griffith.edu.au

ABSTRACT

Product Quantisation (PQ) has been recognised as an effective encoding technique for scalable multimedia content analysis. In this paper, we propose a novel learning framework that enables an end-to-end encoding strategy from raw images to compact PQ codes. The system aims to learn both PQ encoding functions and codewords for content-based image retrieval. In detail, we first design a trainable encoding layer that is pluggable into neural networks, so the codewords can be trained in back-forward propagation. Then we integrate it into a Deep Convolutional Generative Adversarial Network (DC-GAN). In our proposed encoding framework, the raw images are directly encoded by passing through the convolutional and encoding layers, and the generator aims to use the codewords as constrained inputs to generate full image representations that are visually similar to the original images. By taking the advantages of the generative adversarial model, our proposed system can produce high-quality PQ codewords and encoding functions for scalable multimedia retrieval tasks. Experiments show that the proposed architecture GA-PQ outperforms the state-of-the-art encoding techniques on three public image datasets.

KEYWORDS

Product Quantisation; Generative Adversarial Nets; Image Retrieval

ACM Reference Format:
Litao Yu, Yongsheng Gao, and Jun Zhou. 2018. Generative Adversarial Product Quantisation. In *2018 ACM Multimedia Conference (MM '18), October 22–26, 2018, Seoul, Republic of Korea.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3240508.3240590

1 INTRODUCTION

Content-based similarity search in large-scale databases has attracted an ever-increasing interest in multimedia content analysis and computer vision. However, exact nearest neighbour search for large-scale and high-dimensional data is computationally intractable in real applications. To accelerate the online search and reduce the storage cost, learning to represent feature vectors as compact codes becomes a popular approach for Approximate Nearest Neighbour (ANN) search.

Vector Quantisation (VQ) is an effective encoding technique that can compress high-dimensional data into compact codes[5]. The main idea of VQ is to first learn a dictionary as a codebook, which is used to quantise feature vectors into codewords, then the metrics (e.g., distance) are pre-computed and stored in a lookup table. When the original feature space is decomposed into the Cartesian product of several low-dimensional subspaces, vector quantisation becomes product quantisation (PQ)[9]. In content-based retrieval systems, the advantage of PQ encoding is it can keep the latent data distribution with a tolerable information loss. Another popular encoding technique is binary embedding (hashing), which is to encode feature vectors into compact binary codes, so the Euclidean distance can be approximated by Hamming distance[22]. The advantage of hashing lies in the fast search (commonly faster than VQ), which is implemented by the XOR and POPCOUNT operations. However, hashing often suffers from severe information loss, so in retrieval tasks, the accuracy is not always satisfactory. Both VQ and hashing are designed to compress high-dimensional data into discrete representations. Although they have different encoding and search strategies, the underlying mechanism is to incorporate the most discriminant components into the compact codes.

Building a visual retrieval system contains two dis-joint procedures: visual feature pre-processing, and encoding function learning. Visual feature pre-processing aims to aggregate the visual descriptors into feature vectors, while encoding function learning is to generate the codewords with the minimal information loss. Recently, deep learning has been adopted in developing hashing systems [3, 17, 19, 25]. With the powerful visual representation, the retrieval accuracy has been significantly improved. However, applying deep learning techniques to PQ has rarely been implemented. The difficulties are: (1) the direct quantisation of visual feature vectors may not be the best option for the target retrieval system, because the approximate distances among codewords do not reflect the semantic correlations; (2) the PQ codewords are learned via k-means, which is not differentiable in the intermediate layers of neural networks.

To tackle the above two problems, in this paper, we propose a new PQ encoding method that can directly compute the PQ codes from raw image inputs in an end-to-end framework. First, we apply a visual feature learning model to facilitate the PQ encoding in an appropriate feature space. To facilitate the encoding and train high-quality codewords, we then design a trainable layer NetPQ, which is readily pluggable into the deep neural networks and the PQ codewords are amendable in the training process via back-propagation. Our learning framework is based on Generative Adversarial Nets (GANs)[7]. GANs are a class of deep learning models for generating synthetic samples similar to the original data instances. By restricting the input noise variable of GANs, the system is able to produce *fake* images from the learned PQ codewords that are plausibly similar to the original ones. The generative adversarial model can assist the encoding functions to better fit the potential data distribution at a proper semantic level, thus the PQ codes

MM '18, October 22–26, 2018, Seoul, Republic of Korea

ACM ISBN 978-1-4503-5665-7/18/10…$15.00
https://doi.org/10.1145/3240508.3240590

have a better discriminant power. We refer to our proposed architecture as **Generative Adversarial Product Quantisation** (GA-PQ). The proposed learning framework can be either unsupervised or supervised, depending on the available label information.

We show that the proposed GA-PQ can outperform the binary embedding counterparts and the state-of-the-art PQ encoding methods, by conducting extensive image retrieval experiments on three public datasets.

The rest of the paper proceeds as follows. Section 2 provides a short overview of PQ encoding and search schemes. Section 3 describes the technical details of the proposed GA-PQ framework. Experimental results and analysis are presented in Section 4, with Section 5 concludes the paper.

2 BACKGROUND OF PRODUCT QUANTISATION

In vector quantisation (VQ), a quantiser is to map a feature vector $\mathbf{x} \in \mathbb{R}^D$ to a codeword $\mathbf{c}$ in a codebook C, i.e., $\mathbf{x} \mapsto q(\mathbf{x})$, where $q(\cdot)$ is the encoding function. Given a codebook C, an encoder satisfies the first Lloyd's condition: the encoder $q(\cdot)$ always maps the vector $\mathbf{x}$ to its nearest codeword $\mathbf{c} \in C$. Usually k-means is used to generate the codebook C, and a codeword $\mathbf{c}$ is the mean of the vectors in each cluster when $q(\cdot)$ is fixed, which is the second Lloyd's condition.

When the codeword $\mathbf{c}$ is taken from the Cartesian product of several sub-codebooks, VQ becomes PQ, and a feature vector $\mathbf{x}$ is the concatenation of M sub-vectors with equal lengths, i.e., $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_m, \ldots, \mathbf{x}_M]$, $1 \leq m \leq M$. The dimension of each sub-vector is D/M. $\mathbf{x}$'s nearest codeword $\mathbf{c} \in C$ is the concatenation of the M nearest codewords $C = [\mathbf{c}_1, \ldots, \mathbf{c}_m, \ldots, \mathbf{c}_M]$, where the Cartesian product $C = C_1 \times \ldots, \times C_M$ is a codebook computed by k-means in M subspaces. In the off-line database encoding phase, the m-th sub-vector $\mathbf{x}_m$ is quantised by:

$$q_m(\mathbf{x}_m) = \arg\min_{\mathbf{c}_m^{(k)}} \|\mathbf{x}_m - \mathbf{c}_m^{(k)}\|^2, \quad k = 1, \ldots, K, \tag{1}$$

where $\mathbf{c}_m^{(k)}$ is the k-th centroid in the m-th subspace, and K is the number of centroids in each subspace, $1 \leq k \leq K$. Thus, a feature vector $\mathbf{x}$ can be approximated by M sub-quantisers as $q(\mathbf{x}) = [q_1(\mathbf{x}_1), \ldots, q_m(\mathbf{x}_m), \ldots, q_M(\mathbf{x}_M)]$.

In the off-line encoding phase, each image is encoded into M codewords. Thus, the database only needs to store the indices of the codewords to achieve data compression. The distances among all codewords are pre-computed and stored in a lookup table. In the online retrieval phase, VQ provides two ways to compute the approximate distance between two vectors: symmetric and asymmetric calculations. For symmetric distance calculation (SDC), the distance d between any two codewords in a subspace are pre-computed and stored in a K-by-K lookup table as follows:

$$d(\mathbf{c}_m^{(i)}, \mathbf{c}_m^{(j)}) = \|\mathbf{c}_m^{(i)} - \mathbf{c}_m^{(j)}\|^2. \tag{2}$$

Given a query image $\mathbf{q}$, it can be first encoded to M codewords $[\mathbf{q}_1, \ldots, \mathbf{q}_m]$, and its distance to an image $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_M]$ in the reference set is estimated by:

$$d(\mathbf{q}, \mathbf{x}) = \sum_{m=1}^{M} d(\mathbf{q}_m, \mathbf{x}_m). \tag{3}$$

For asymmetric distance calculation (ADC), the distance in each subspace is calculated online and stored in a 1-by-K lookup table. In either SDC or ADC, the distance between two vectors is approximated by the summation of distances in M subspaces.

In the retrieval phase, the distances are sorted in ascending order, with the top samples considered as the most relevant instances. To accelerate the computation, the inverted index can be applied for non-exhaustive search[9].

There are several optimised PQ approaches, which seek to achieve a better accuracy in content-based visual retrieval systems. For example, Ge et al. proposed the Optimised Product Quantisation (OPQ) to compute an orthogonal matrix to minimise the data distortion in the rotated feature space[4], which is similar to Iterative Quantisation (ITQ) for binary embedding[6]. Similarly, we proposed the Bilinear Optimised Product Quantisation (BOPQ) for high-dimensional vector encoding[24] at a much less memory cost. In [10], Yannis et al. extended OPQ for ANN search by computing the local rotation matrices and quantising the residuals within the coarse clusters (cells). The optimised PQ methods generally have a better performance for ANN tasks. However, we found that minimising the data distortion does not improve the accuracy at the semantic level, because it does not contribute to enhancing the semantic correlations of the PQ codewords in some cases. In our proposed GA-PQ framework, the codewords are fine-tuned in the optimisation procedure with the consideration of higher-level constraints, so the encoding can be conducted at the semantic level to improve the retrieval performance.

3 GENERATIVE ADVERSARIAL PRODUCT QUANTISATION

In this section, we introduce our GA-PQ learning framework in detail. First, we design a trainable PQ layer with triplet constraints, which can be plugged into the neural networks. Then we elaborate on how to integrate the PQ layer into a Generative Adversarial Net, and how to optimise the model.

3.1 System Overview

The overall framework of the proposed GA-PQ network is illustrated in Fig. 1. Our framework is comprised of three components: (1) a convolutional neural network (CNN) based encoder, with a NetPQ stacked on top of the last layer; (2) a convolutional generator begins with the codewords as the input, which aims to procude new images from the learned PQ codewords; and (3) a CNN based discriminator to distinguish the original images and generated images. The reason for applying the GAN based architecture is straightforward: we aim to restrict the noise input to seek a "better" latent representation, so the encoding applied on such representation can generate *fake* images that look similar to the original ones. In the NetPQ layer, the encoding functions and codebook can be directly optimised by back-propagation. In the next subsection, we describe the NetPQ layer in detail.

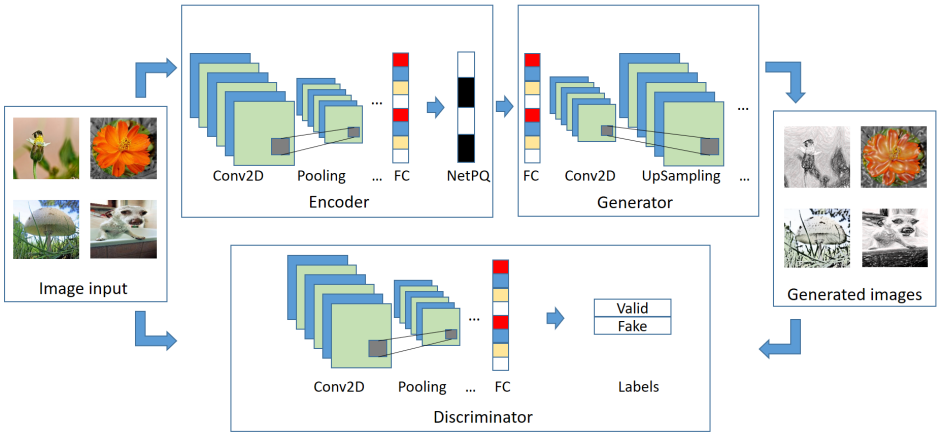


Figure 1: The general learning framework of GA-PQ

3.2 NetPQ: A Trainable PQ Model with Triplet Constraints

We design a NetPQ layer that allows the end-to-end encoding from raw image inputs to compact PQ codes. The streamline of passing a feature vector through NetPQ layer is illustrated in Fig. 2.

The codebook of PQ is computed on the feature representation by k-means, and discontinuities in PQ encoding is the hard assignment of a sub-vector to the nearest codeword in its subspace as shown in Eq.(1). Such property makes the standard back-propagation infeasible in network training. To make the assignment differentiable, we apply both soft and hard assignment of vector partitions on multiple codewords $\{\mathbf{c}_m^{(k)}|k=1,\ldots,K\}$. This idea of applying soft assignment of codewords is inspired by NetVLAD proposed by Relja et al. [1], which implements a trainable VLAD layer to aggregate convolutional descriptors into feature vectors for visual place recognition. The soft assignment function for PQ codes is:

$$p_m^{(k)}(\mathbf{x}_m) = \frac{e^{-\gamma\|\mathbf{x}_m-\mathbf{c}_m^{(k)}\|^2}}{\sum_{k'=1}^{K} e^{-\gamma\|\mathbf{x}_m-\mathbf{c}_m^{(k')}\|^2}}, \quad k=1,\ldots,K, \tag{4}$$

where $p_m = [p_m^{(1)},\ldots,p_m^{(K)}]$ is the assignment probability set and γ is a positive scaling parameter. When $\gamma \to \infty$, the soft-assignment function becomes a hard one. This function can assign a sub-vector $\mathbf{x}_m$ to the codewords $\{\mathbf{c}_m^{(k)}|k=1,\ldots,K\}$ proportional to their proximity, and ensures $0 < p_m^{(k)}(\mathbf{x}_m) < 1$, with the highest value assigned to the closest codeword.

If we expand the squares in Eq.(4), and omit the term $e^{-\gamma\|\mathbf{x}_m\|^2}$, the soft-assign function becomes a soft-max regression function:

$$p_m^{(k)}(\mathbf{x}_m) = \frac{e^{\mathbf{w}_m^{(k)\top}\mathbf{x}_m+b_m^{(k)}}}{\sum_{k'=1}^{K} e^{\mathbf{w}_m^{(k')\top}\mathbf{x}_m+b_m^{(k')}}}, \quad k=1,\ldots,K, \tag{5}$$

where the weight vector $\mathbf{w}_m^{(k)} = 2\alpha\mathbf{c}_m^{(k)}$ and bias scalar $b_m^{(k)} = -\gamma\|\mathbf{c}_m^{(k)}\|^2$.

Thus, a soft sub-encoder is actually a linear combination of K codewords with a soft-max activation in a subspace. The soft presentation of the codewords can be computed by:

$$\tilde{q_m}(\mathbf{x}_m) = \sum_{k=1}^{K} p_m^{(k)}\mathbf{c}_m^{(k)}. \tag{6}$$

Let $k^* = \arg\max_k(\mathbf{w}_m^{(k)\top}\mathbf{x}_m + b_m^{(k)})$ be the index of the highest probability in p_m and let $e_m = [e_m^{(1)},\ldots,e_m^{(K)}]$ be a one-hot encoding, such that $e_m^{(k^*)} = 1$ and $e_m^{(k)} = 0$ for $k \neq k^*$. The hard representation of the codewords is:

$$\hat{q_m}(\mathbf{x}_m) = \sum_{k=1}^{K} e_m^{(k)}\mathbf{c}_m^{(k)}. \tag{7}$$

Given a vector $\mathbf{x} = [\mathbf{x}_1,\ldots,\mathbf{x}_M] \in \mathbb{R}^D$ as the input, the outputs of NetPQ are two D-dimensional vectors with both soft and hard representations, and each of them is concatenated by M sub-vectors, respectively:

$$\begin{aligned}\tilde{q}(\mathbf{x}) &= [\tilde{q_1}(\mathbf{x}_1),\ldots,\tilde{q_M}(\mathbf{x}_M)],\\ \hat{q}(\mathbf{x}) &= [\hat{q_1}(\mathbf{x}_1),\ldots,\hat{q_M}(\mathbf{x}_M)].\end{aligned} \tag{8}$$

NetPQ tries to minimise the central loss L_c, i.e., the Euclidean distance between $\tilde{q}(\mathbf{x})$ and $\hat{q}(\mathbf{x})$ to encourage features from the same class to be clustered together:

$$L_c = \sum_{\mathbf{x}} \|\tilde{q}(\mathbf{x}) - \hat{q}(\mathbf{x})\|^2. \tag{9}$$

The parameter sets $\{\mathbf{w}_m^{(k)}\}$, $\{b_m^{(k)}\}$, and $\{\mathbf{c}_m^{(k)}\}$ are all differentiable, so our proposed NetPQ is pluggable as a meta layer in neural networks. Note that the one-hot vector e_m is not differentiable in the back-forward optimisation. During the training process, e_m forces $\{\mathbf{w}_m^{(k)}\}$, $\{b_m^{(k)}\}$, and $\{\mathbf{c}_m^{(k)}\}$ to generate similar hard and soft representations, respectively.

To increase the discriminative power of NetPQ, in our system we add the triplet loss L_t introduced by [16] in the optimisation

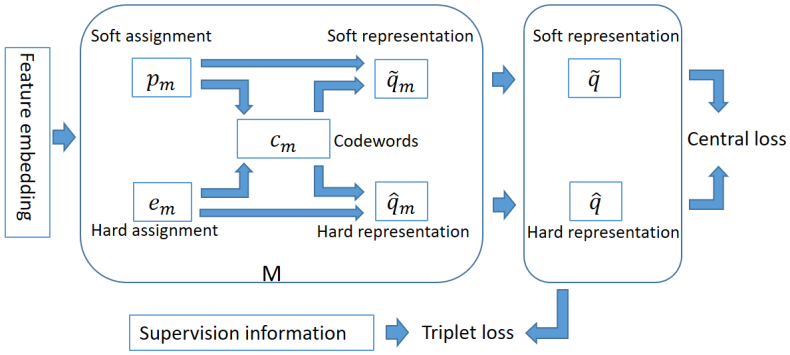


Figure 2: The computation streamline of NetPQ

when there is supervision information available. In the unsupervised case, we can use a pre-trained deep model to calculate the flatten visual feature map as the weak supervision information. Such feature representation may not fit the target image retrieval system, but can generally distinguish the similar and dissimilar data instances. In the supervised case, label information is directly used to generate triplets. Assume there is an anchor image $\mathbf{x}^a$ similar to a positive image $\mathbf{x}^p$ but dissimilar to a negative image $\mathbf{x}^n$, where their Euclidean distances satisfy:

$$\|\mathbf{x}^a - \mathbf{x}^p\|^2 + \alpha < \|\mathbf{x}^a - \mathbf{x}^n\|^2, \tag{10}$$

where α is a margin that is enforced between positive and negative pairs. In the encoding system, we want the approximated distance computed by Eq. (3) also follow the constraint:

$$d(\hat{q}(\mathbf{x}^a), \hat{q}(\mathbf{x}^p)) + \alpha < d(\hat{q}(\mathbf{x}^a), \hat{q}(\mathbf{x}^n)). \tag{11}$$

So the triplet loss becomes:

$$L_t = \sum_{\mathbf{x}^a} [d(\hat{q}(\mathbf{x}^a), \hat{q}(\mathbf{x}^p)) - d(\hat{q}(\mathbf{x}^a), \hat{q}(\mathbf{x}^n)) + \alpha]_+. \tag{12}$$

When there is label information available in the training process, a positive sample is selected in the same class with the anchor image, while a negative sample is selected from other classes that is *close* to the anchor image. By minimising the triplet loss L_t, the codewords are no longer constrained within the original data distribution, but with a more discriminative power at the semantic level in the relaxed feature space.

Since it is infeasible to iterate all possible triplets within a large training dataset, it is crucial to select the hard triplets that contribute to the PQ codewords. In the batch optimisation process, the gradients are only generated when the triplet violates Eq. (11). Otherwise, the gradients are zero. Thus, the triplets are dynamically selected in each mini-batch.

3.3 Integrating NetPQ into Adversarial Generative Nets

Our proposed GA-PQ learning framework is based on Adversarial Generative Nets (GANs) [7]. GANs are unsupervised or semi-supervised deep learning models that are implemented by two independent networks, a generator **G** and a discriminator **D**, contesting with each other in a zero-sum game framework. The generator **G** learns to map the visual features from a latent space to a particular data distribution of interest, while the discriminator **D** learns to distinguish between instances from the true data distribution and the generated candidates. In the inference procedure, the generator's objective is to increase the error of the discriminator by producing synthesised data instances that appear to be from the true data distribution, while the discriminator is optimised to increase its discriminative ability. The two networks simultaneously evolve until they reach an equilibrium.

Let I be the original image set, and I^R be the generated images by **G**, respectively, the adversarial loss of the min-max game for GANs is formulated as follows:

$$\min_{\Theta_G} \max_{\Theta_D} \log(\mathbf{D}(I)) + \log(1 - \mathbf{D}(I^R)), \tag{13}$$

where Θ_G and Θ_D are parameters for generator **G** and discriminator **D**, respectively.

Recently, many variations of GANs have been proposed, such as Auxiliary Classifier GANs (AC-GAN)[13], deep convolutional GANs (DC-GAN)[14], Semi-supervised GANs (SS-GAN)[20], Least-square GAN (LS-GAN)[12] and Wasserstein GANs (WGAN)[2, 8]. In image processing, GANs have many applications such image enhancement[15], 3D object modelling[23], and cross-model retrieval [21]. In the proposed GA-PQ framework, we use the LS-GAN[12] as the generative adversarial for data enhancement to produce better PQ codewords.

The GA-PQ framework starts with an encoder **E**, which is basically a CNN model followed by a NetPQ layer. It first extracts the visual feature maps from the raw image input then encodes the flatten visual feature vectors into PQ codewords. In unsupervised GANs, the generator **G** is seeded with the random input sampled from a pre-defined feature space. However, in the GA-PQ framework, we use the concatenated PQ codewords from the output of the encoder **E** as the input of **G**, which is not random noise variables, to generate images that are visually similar to the original ones. Applying generative adversarial models for PQ training is straight-forward: in content-based image retrieval systems, if the PQ codes of a query image can be recovered to a *fake* image, the visually related images can also be encoded to the similar PQ codes in the reference set. The generator **G** is able to produce plausibly

valid images but actually *fake* ones when given the PQ codes, so these generated images are also complementary to the training dataset.

The generator **G** begins with a dense layer, then reshapes the feature output to a tensor. Then we use the de-convolutional blocks comprised of transposed convolutional layers, upsampling layers and batch normalisation layers to restore a feature vector to a *fake* image. The tanh activation is applied to the last convolutional layer, corresponding to three channels for RGB images.

The discriminator **D** is defined in a similar way as the encoder, which consists of the convolutional blocks and dense layers. In the learning process, **D** tries to distinguish if an input image is original or generated from the PQ codewords, which is adversarial to the generator **G**.

3.4 Training of GA-PQ

The training of GA-PQ consists of the following steps:

1. Learning a CNN-based model for visual feature extraction. We use VGG16 model[18] pre-trained on ImageNet for the preparation of PQ encoding. For general purpose, we use the output of first fully-connected layer with L2 norm for visual feature extraction. The feature model is frozen and all weight parameters will not change in the later stages.

2. Initialising the NetPQ layer. To accelerate the training process and ensure the codewords uniquely map the data instances in the PQ encoding space, we adopte k-means to pre-compute $\{\mathbf{c}_m^{(k)}\}$, and train a soft-max regression model to initialise $\{\mathbf{w}_m^{(k)}\}$ and $\{b_m^{(k)}\}$ the NetPQ layer introduced in Section 3.2.

3. Global optimisation of GA-PQ. In this step, the initialised encoder **E** is integrated into the GAN model. Both the PQ encoding functions and codewords are fine-tuned, together with the parameters of GAN. In each iteration, the system randomly selects a batch of training images and optimises the encoder **E** by minimising the central loss L_c in Eq. (9) and triplet loss L_t in Eq. (12) respectively. Then the system feeds the codewords representation into the generator **G** by forward-propagation, to generate a batch of *fake* image. The input image batch is marked with a *valid* label, and the generated batch is marked with a *fake* label, respectively. Both *valid* and *fake* images are fed into the discriminator **D**, and the parameters Θ_D are optimised by gradient descent in back-propagation. After that, we combine the encoder **E**, the generator **G** and the discriminator **D**, and freeze Θ_D and only update Θ_E and Θ_G. To do this, we feed another batch of training images into the combined model, set the *valid* labels, and conduct a batch-update. The parameters Θ_E, Θ_D and Θ_G are iteratively updated until convergence.

4 EXPERIMENTS

In this section, we describe the experiments on three image datasets, then give both quantitative and qualitative results to validate our proposed GA-PQ learning framework.

4.1 Datasets

Our experiments were conducted on three public image datasets: (1) CIFAR-10 dataset[1], (2) Oxford Buildings dataset[2], and (3) Paris Buildings dataset[3].

The CIFAR-10 dataset consists of 60,000 low-resolution colour images in 10 classes, and all images are well labelled. The retrieval system aims to search the semantically similar images when given query images. For each class, 100 samples were randomly selected from the test set as queries, and 3,000 images from the training set were used for learning encoding parameters. The whole training set was used as the reference set.

In the Oxford Buildings dataset, 5,062 images were collected and annotated to generate comprehensive ground truth for 11 different landmarks, each represented by 5 possible queries. To evaluate the retrieval accuracy, we used the images with *Good* labels provided as the ground-truth, i.e., the images are nice and clear to recognise the object/building. The Paris Buildings dataset consists of 6,412 images describing 12 particular landmarks of Paris city. In this dataset, there are no *Good* labels corresponding to the queries, so we used the *OK* labels as ground-truth, i.e., more than 25% of the objects in these images are clearly visible. We injected the above three reference image sets to Flickr250K dataset[4], where all images are irrelevant to the two datasets, to form a large-scale reference set to test the retrieval performance. In the off-line training process, we randomly sampled 50,000 images from the reference set to learn the encoding functions and PQ codewords.

4.2 Baselines

We compare GA-PQ with other state-of-the-art encoding methods for image retrieval tasks, including both binary embedding and product quantisation as follows:

(1) **Iterative Quantisation (ITQ)**[6]: Applying the PCA and iterative quantisation to generate binary codes;
(2) **PQ**[9]: The standard PQ algorithm without any re-order of the dimensions or optimisation. The codebook is computed via k-means in multiple subspaces;
(3) **Optimised PQ (OPQ)**[4]: Applying a global rotation matrix to minimise the quantisation distortion. Specifically, we use the parametric algorithm (balanced partition) to compute the orthogonal rotation matrix;
(4) **Locally Optimised PQ (LOPQ)**[10]: Using multiple rotation matrices to locally minimise the data distortion and quantise the residuals in different cells;
(5) **Binary Generative Adversarial Networks (BGAN)**[19]: Applying GANs to generate binary codes for image retrieval.

Among the above baseline algorithms, ITQ and BGAN are the state-of-the-art non-deep and deep hashing algorithms, respectively. The mechanisms behind BGAN and GA-PQ are similar, and both of them use adversarial models to improve the quality of compact codes. However, they have distinct encoding and search strategies. Besides of these, BGAN adopts the neighbourhood structure loss while our GA-PQ applies the triplet loss.

[1] https://www.cs.toronto.edu/~kriz/cifar.html
[2] http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/
[3] http://www.robots.ox.ac.uk/~vgg/data/parisbuildings/
[4] http://press.liacs.nl/mirflickr/

To show the benefits of the end-to-end encoding framework, we compare our GA-PQ method against the "off-the-shelf" networks pre-trained on other tasks for image retrieval. For non-deep models ITQ, PQ, OPQ and LOPQ, we use the output of the first dense layer in VGG16 trained on ImageNet as the visual feature. In these methods, the visual feature vectors cannot be updated in the learning process, which is the basic difference to our proposed GA-PQ. For PQ based encoding, each data instance is represented by an M-dimensional vector with 8-bit unsigned integer values. To keep the same storage cost for fair comparison, $M \times 8$-bit binary codes were used to encode an image for binary embedding.

4.3 Evaluation Metrics

In the performance evaluation of a retrieval system, Average Precision (AP) is a single value that evaluates the retrieval performance by averaging the precision values of top-retrieved data instances given a single query. We use Mean Average Precision (MAP) to evaluate the retrieval accuracy, which is the average AP values of all queries. Also, we use the recall curves to visualise the retrieval performance.

4.4 Implementation Details

We implemented the proposed NetPQ layer with TensorFlow. On top of each subspace of the feature output, we simply put a small dense layer with soft-max activation and an assignment layer. Each assignment layer contains 256 vectors that represent the PQ codewords. The overall GA-PQ learning framework is also implemented by stacking the layers defined in the Keras module.

On CIFAR-10 dataset, we set the code length M to 4, 8 and 16, respectively, since the images have a smaller size and lower-dimensional visual features. The generator **G** contains a dense layer and four convolutional blocks. The first three convolutional blocks comprised of convolutional layers (with 128, 64, 32 filters, respectively), upsampling layers and batch normalisation layers. To avoid the sparse gradients which cause the instability issue, we choose the Leaky Rectified Linear Unit (LeakyReLU)[11] as the activation of the first three convolutional layers. The last block contains one convolutional layer with 3 filters and tanh activation. The discriminator **D** outputs the estimated probability that the PQ codewords are drawn from the distribution of the generator **G**, which is stacked by four convolutional blocks (the convolutional layers have 32, 64, 128 and 256 filters, respectively) and two dense layers, and the last dense layer is with the sigmoid activation for the binary *valid/fake* label.

On Oxford Buildings and Paris Buildings datasets, we resized all images to 224×224, which is consistent with the default setting for training VGG16 model on ImageNet, and doubled M for all encoding methods. Similar to the settings on CIFAR-10 dataset, the generator **G** contains a dense layer and six convolutional blocks (with 512, 256, 128, 64, 32 and 3 filters, respectively). The discriminator **D** is defined in a similar way as the encoder, which is stacked by five convolutional blocks and two dense layers.

Our experiment was conducted on a desktop equipped with a Nvidia Titan Xp GPU card.

Table 1: The MAP results using different loss functions on CIFAR-10 dataset

Components	$M = 4$	$M = 8$	$M = 16$
L_c	0.376	0.391	0.382
$+ L_t$	0.724	0.737	0.716
$+ L_a$	0.764	0.740	0.733

4.5 Results and Analysis

4.5.1 Component Analysis of Loss functions. In our proposed GA-PQ, there are three loss functions: central loss L_c, triplet loss L_t and adversarial loss L_a. In the training procedure, the above objectives are iteratively optimised. We study the effect of each loss function on the retrieval performance by reporting the results on CIFAR-10 dataset as in Table 1. If we only apply the joint central loss, the NetPQ layer is nearly equivalent to original PQ. However, solely optimising the NetPQ layer yields an unsatisfactory retrieval performance because the data distribution of the encoding space is totally ignored, so the accuracy is even worse than the original PQ method. If adding the triplet loss, the codewords become much more discriminative, which leads to around 30% improvement of the MAP. When integrating the NetPQ with DC-GAN, the system achieves the best performance. The reason is that the discriminator **D** can force the PQ codewords to generate identical images through the generator **G**, thus the feature map can be learned in a better latent space.

4.5.2 Image Reconstruction Results. We use the PQ codewords as the constrained input for the generative adversarial model, which is the same with BGAN[19] that uses the binary codes to generate *fake* images, to improve the quantised data representation. To visualise the image reconstruction, we show some qualitative results on CIFAR-10 dataset in Fig. 3. The first row shows the original images, the middle row gives the reconstructed images using BGAN, and the bottom row displays the reconstruction results from our proposed GA-PQ.

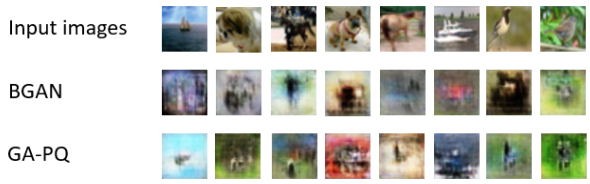


Figure 3: Image reconstruction samples using BGAN and GA-PQ on CIFAR-10 dataset

4.5.3 Comparison with different encoding methods. Building the index for large-scale image databases to facilitate online search is usually a time-consuming work. We tested the image processing efficiencies on the Flickr250K by recording the times required for feature extraction and encoding, which are shown in Table 2. Since the compact PQ codes are computed from the output of the first dense layer in VGG16 model, the time complexities of feature extraction are the same. However, in the encoding phase, they have

Table 2: Encoding time comparison of different PQ methods

Method	Flickr250K		
	$M = 8$	$M = 16$	$M = 32$
PQ	8,042 ms	8,238 ms	9,397 ms
OPQ	9,787 ms	10,973 ms	11,273 ms
LOPQ	145,206 ms	146,104 ms	146,879 ms
GA-PQ	8,185 ms	8,213 ms	8,432 ms

different time complexities. Our GA-PQ is computed with the GPU acceleration, and the general encoding time is similar to PQ. LOPQ has the highest time complexities.

Figure 7: Top retrieved image samples using different encoding methods on Paris Building dataset

The retrieval MAPs of all encoding methods introduced in subsection 4.2 with respect to different code lengths are illustrated in Table 3. From this table, we have the following observations: (1) PQ based models generally achieve better results than hashing based models, and the reason is the information loss for PQ models is lower. Such results are consistent with [4, 9, 24]. (2) On all of the three datasets, the retrieval accuracies of deep models are generally higher than that of shallow models. (3) The standard PQ performs suprisingly better than OPQ on CIFAR-10 dataset. Such phenomenen signifies the semantic correlation might have an even higher importance than the geometric property in the feature space. (4) Our proposed GA-PQ generally outperforms all other encoding approaches. Specifically, the GA-PQ outperforms the best PQ counterpart by 3%, 2% and 4%, respectively, and outperforms the state-of-the-art hashing (BGAN) counterpart by 2%, 1% and 4%, on the three datasets, respectively. (5) Except the performance on CIFAR-10 dataset, MAP generally increases for all methods when code length M varies. It is understandable that when code length is short, the compact codes have less discriminative power. For the "shallow" models ITQ, PQ, OPQ and LOPQ, their retrieval performance varies on three datasets, and the pre-trained features without fine-tuning procedure cannot guarantee that the encoding is conducted in the optimal feature space.

From Fig. 4 to Fig. 6, we show the recall curves of different models on these datasets, which also supports the superior retrieval performance of the GA-PQ in most cases. The underlying principle is that our GA-PQ framework uses the trainable PQ encoding layer, which can be further optimised when some prior information is given. Such information is provided by both triplet constraints and the adversarial generative model. The triplets are generated in the batch optimisation, reflecting the data distribution of all possible data samples in the database. The adversarial generative network, from another perspective, implements the image data augmentation, which also brings benefits to produce better PQ codewords.

In Fig. 7, we show a demonstration of the image retrieval examples using different encoding methods on Paris Building dataset, where top 10 retrieved images are displayed. The images with red bounding boxes indicate the failure results. From the partial list, we can see PQ, LOPQ and our proposed GA-PQ return the largest number of correctly retrieved samples in the reference set.

5 CONCLUSION

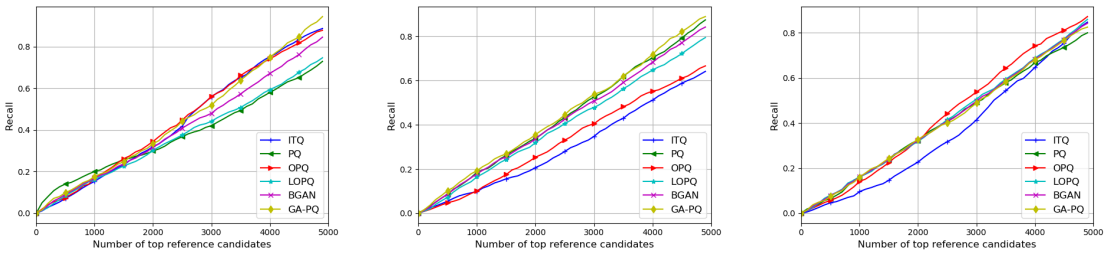
In this paper, we address two important problems in building effective PQ encoding functions for large-scale image retrieval. The first problem is how to embed a trainable PQ encoding layer in the neural network. In our work, we formulated the NetPQ layer to enable the end-to-end encoding from raw images to compact PQ codes. The second problem is how to obtain high-quality PQ codes for accurate image retrieval when training source is limited. For this issue, we used PQ codewords as the constrained inputs of the generative adversarial model. The proposed GA-PQ learning framework can be either supervised or unsupervised, depending on the available label information. We tested our proposed method for scalable image retrieval task on three public datasets. The experimental results demonstrate that our proposed GA-PQ outperforms the state-of-the-art encoding techniques, including both VQ and hashing based methods.

ACKNOWLEDGMENTS

This work was supported by the Australian Research Council under Project DP180100958. The Titan Xp GPU card was donated by NVIDIA corporation.

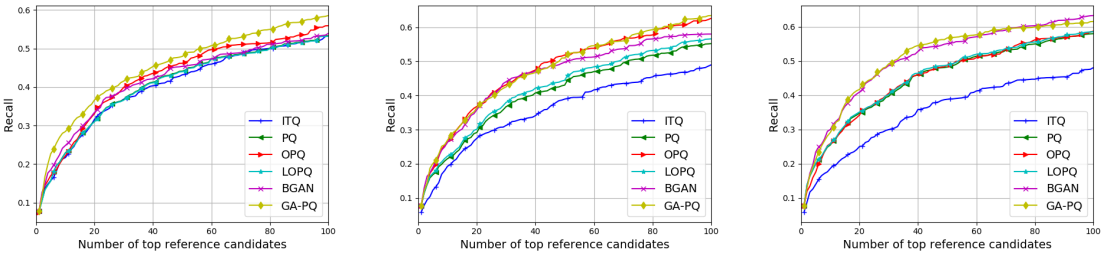
Table 3: MAP comparison for different encoding methods

Method	CIFAR-10			Oxford Buildings			Paris Buildings		
	$M = 4$	$M = 8$	$M = 16$	$M = 8$	$M = 16$	$M = 32$	$M = 8$	$M = 16$	$M = 32$
ITQ	0.548	0.523	0.469	0.304	0.318	0.338	0.397	0.429	0.458
PQ	**0.803**	0.694	0.647	0.332	0.349	0.355	0.460	0.495	0.523
OPQ	0.371	0.327	0.357	0.361	0.365	0.364	0.545	0.567	0.585
LOPQ	0.679	0.692	0.727	0.351	0.353	0.364	0.524	0.535	0.578
BGAN	0.729	0.721	0.724	0.361	0.358	0.383	0.548	0.545	0.590
GA-PQ	0.764	**0.740**	**0.733**	**0.362**	**0.388**	**0.399**	**0.588**	**0.607**	**0.618**



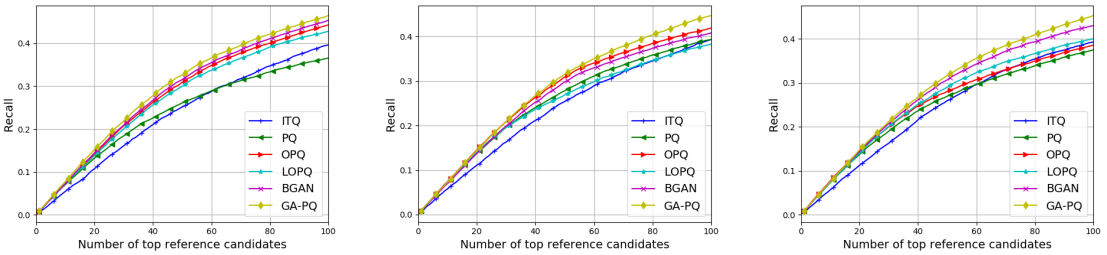
(a) $M = 4$ **(b)** $M = 8$ **(c)** $M = 16$

Figure 4: Recall comparisons of GA-PQ versus other encoding methods on CIFAR-10 dataset



(a) $M = 8$ **(b)** $M = 16$ **(c)** $M = 32$

Figure 5: Recall comparisons of GA-PQ versus other encoding methods on Oxford Buildings dataset



(a) $M = 8$ **(b)** $M = 16$ **(c)** $M = 32$

Figure 6: Recall comparisons of GA-PQ versus other encoding methods on Paris Buildings dataset

REFERENCES

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. 2016. NetVLAD: CNN architecture for weakly supervised place recognition. In *IEEE CVPR*. 5297–5307.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *ICML*. 214–223.

[3] Yue Cao, Mingsheng Long, Jianmin Wang, and Shichen Liu. 2017. Deep visual-semantic quantization for efficient image retrieval. In *IEEE CVPR*, Vol. 2. 6.

[4] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2014. Optimized Product Quantization. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 36, 4 (2014), 744–755.

[5] Allen Gersho and Robert M Gray. 2012. *Vector quantization and signal compression*. Vol. 159. Springer Science & Business Media.

[6] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 35, 12 (2013), 2916–2929.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*. 2672–2680.

[8] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *NIPS*. 5769–5779.

[9] Herve Jegou, Matthijs Douze, and Cordeli Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 33, 1 (2011), 117–128.

[10] Yannis Kalantidis and Yannis Avrithis. 2014. Locally optimized product quantization for approximate nearest neighbor search. In *IEEE CVPR*. 2321–2328.

[11] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML*. 3.

[12] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. 2017. Least squares generative adversarial networks. In *IEEE ICCV*. 2813–2821.

[13] Augustus Odena, Christopher Olah, and Jonathon Shlens. 2016. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585* (2016).

[14] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).

[15] Mehdi SM Sajjadi, Bernhard Scholkopf, and Michael Hirsch. 2017. EnhanceNet: Single image super-resolution through automated texture synthesis. In *IEEE CVPR*. 4491–4500.

[16] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *IEEE CVPR*. 815–823.

[17] Fumin Shen, Yan Xu, Li Liu, Yang Yang, Zi Huang, and Heng Tao Shen. 2018. Unsupervised deep hashing with similarity-adaptive and discrete optimization. *IEEE Trans. on Pattern Analysis and Machine Intelligence* (2018).

[18] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[19] Jingkuan Song, Tao He, Lianli Gao, Xing Xu, Alan Hanjalic, and Heng Tao Shen. 2018. Binary Generative Adversarial Networks for Image Retrieval. In *AAAI*. 394–401.

[20] Jost Tobias Springenberg. 2015. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390* (2015).

[21] Bokun Wang, Yang Yang, Xing Xu, Alan Hanjalic, and Heng Tao Shen. 2017. Adversarial Cross-Modal Retrieval. In *ACM MM*. 154–162.

[22] Jingdong Wang, Ting Zhang, Nicu Sebe, and Heng Tao Shen. 2017. A survey on learning to hash. *IEEE trans. on pattern analysis and machine intelligence* (2017).

[23] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. 2016. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NIPS*. 82–90.

[24] Litao Yu, Zi Huang, Fumin Shen, Jingkuan Song, Tao Shen, Heng, and Xiaofang Zhou. 2017. Bilinear Optimized Product Quantization for Scalable Visual Content Analysis. *IEEE Trans. on Image Processing* 26, 10 (2017), 5057 – 5069.

[25] Haofeng Zhang, Li Liu, Yang Long, and Ling Shao. 2018. Unsupervised Deep Hashing With Pseudo Labels for Scalable Image Retrieval. *IEEE Trans. on Image Processing* 27, 4 (2018), 1626–1638.