SPACE SHUTTLE MODEL: A PHYSICS INSPIRED METHOD FOR LEARNING QUANTIZABLE DEEP REPRESENTATIONS

*Shicong Liu, Hongtao Lu**
{artheru,htlu}@sjtu.edu.cn
Key Laboratory of Shanghai Education Commission for
Intelligent Interaction and Cognitive Engineering,
Department of Computer Science and Engineering,
Shanghai Jiao Tong University, P.R.China
Phone: +86-21-34204879, Fax: +86-21-34204728

ABSTRACT

Recent advance of large scale similarity search involves using deeply learned representations to improve the search accuracy and use vector quantization methods to increase the search speed. However, how to learn deep representations that both strongly preserve similarities between data pairs and can be accurately quantized via vector quantization remains a challenging task. In this paper, we propose a novel physics based method named *space shuttle model* (SSM) to learn effective deep representations that can be accurately quantized. It consider network output as a roaming space shuttle "propelled" by similarity loss and subject to "gravitational forces" from quantization codewords. SSM is related to momentum methods commonly used in deep learning but is applied on network outputs instead of network parameters. Experimental results on large scale similarity search demonstrate that the proposed framework outperforms the state-of-the-art.

Index Terms— Large Scale Search, Deep Learning, Vector Quantization

1. INTRODUCTION

Approaches to large scale similarity search typically involves compressing a high-dimensional representation into a short code of a few bytes by hashing [1, 2] or vector quantization [3]. Short code embedding greatly save space and accelerate the search speed. A trending research topic is to generate short codes that preserves semantic similarities, e.g., [4, 5]. Traditional short code generation methods require that the input data is firstly represented by some hand-crafted representations (e.g., GIST [6] for images). Recently, learned representations with a deep neural network and a similarity loss is more preferable due to its stronger ability to distinguish semantic similarities[7, 8].

*Corresponding author

In this paper we explore learning effective deep representations and short codes for fast and accurate similarity search. A key problem is that we don't know what exactly the representations should be, since we're only given "similar" and "dissimilar" pairs. [9, 10] propose to learn deep representations and mappings to binary hash codes with a deep convolutional neural network. However these deeply learned representations must exhibit certain clustering structures for effective short codes generation or severe performance loss could induce. How to learn representations that can be converted into short codes with minimal performance loss still remains a challenging task. Existing methods [11, 12, 13] add a quantization loss term to the similarity loss to regularize the output to be *quantizable*. However this methodology leads to difficulties in optimization, since they actually introduce a constant bias in the gradients computed from similarity loss for each sample. A joint representation and vector quantization codebook learning framework is highly desirable.

To this end, we put forward a novel physics inspired methodology named *Space Shuttle Model* (SSM) to learn representations exhibiting required structures for vector quantization: We treat network outputs as a roaming space shuttle, which are propelled by "similarity loss" and are subject to "gravity" from neighboring quantization codewords. Then we can use the velocity of the space shuttle as gradients. This method can be seen as applying momentum on the network outputs, which is usually applied on network parameters. Experimental results on ILSVRC 2012 [14], CIFAR [15], MNIST [16] and NUS-WIDE [17] show that our joint optimization framework significantly outperforms state-of-the-art methods by a large margin in term of search accuracy.

978-1-5090-6067-2/17/$31.00 ©2017 IEEE

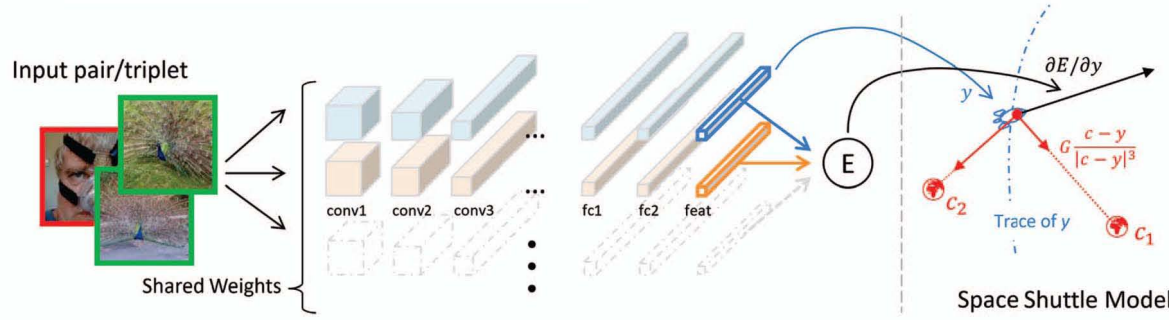

Fig. 1: *Illustrative Explanation of Space Shuttle Model*: The network output $\mathbf{y}$ has a momentum and is subject to two "forces", one is similarity loss, and the other is gravity from quantization codewords. Then we can compute the "velocity", aka, $\Delta\mathbf{y}$ and perform the back-propagation.

2. RELATED WORKS

2.1. Deep Representations for Similarity Search

Convolutional Neural Networks have shown strong performances in image classification tasks. Qualitative [18] and quantitative [19] evidences of their feasibility for image retrieval tasks have also been observed in the former studies. Convolutional networks can generate effective deep representations within the *siamese architectures*[7], and are successful in fine-grained classification [8], unsupervised learning[20], and image retrieval[9, 10, 11, 13] tasks, etc. A Siamese architecture consists of several streams of layers sharing the same weights and uses a multi-wise loss function to train the network. A commonly used loss function for retrieval tasks is triplet loss[10], since it characterizes a relative ranking order and achieves satisfying performances[21]. Formally, given a sample $\mathbf{x}$, which is more similar to $\mathbf{x}^+$ rather than $\mathbf{x}^-$, the triplet loss is defined as :

$$E = max\{0, g + \|\hat{\mathbf{y}}_{\mathbf{x}} - \hat{\mathbf{y}}_{\mathbf{x}^+}\|_2 - \|\hat{\mathbf{y}}_{\mathbf{x}} - \hat{\mathbf{y}}_{\mathbf{x}_-}\|_2\} \quad (1)$$

, where $\hat{\mathbf{y}}_{\mathbf{x}}$, $\hat{\mathbf{y}}_{\mathbf{x}_-}$, $\hat{\mathbf{y}}_{\mathbf{x}_+}$ are the deep representations of $\mathbf{x}, \mathbf{x}_-, \mathbf{x}_+$, respectively. g is a gap parameter that regularizes the distance difference between "more similar" pair and "less similar" pair, it also prevents network from collapsing. Pairwise loss [13] and classification loss are also used in recent literatures [5] due to their simplicity for optimization.

Compared to traditional hand-crafted visual features, deeply learned representations significantly improves the performances of image retrieval. However the deep representations obtained with convolutional neural networks are high dimensional (e.g., 4096-d for AlexNet [18]), thus the representations must be effectively compressed for efficient retrieval tasks.

2.2. Vector Quantization for Nearest Neighbor Search

Vector quantization methods are promising for nearest neighbor retrieval tasks. They outperform the hashing competitors like ITQ[2], Spectral Hashing[1], etc, by a large margin[3, 22] on metrics like l_2. Vector quantization techniques involve quantizing a vector as a sum of certain pre-trained vectors and then compress this vector into a short encoding representation. Taking Product Quantization (PQ) [3] as an example, denote a *database* $\mathcal{X}$ as a set of N d-dimensional vectors to compress, PQ learns M *codebooks* $\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_M$ for disjoint M subspaces, each of which is a list of K *codewords*: $\mathbf{c}_m(1)$, $\mathbf{c}_m(2)$, ..., $\mathbf{c}_m(K) \in \mathbb{R}^{d/M}$. Then PQ uses a *mapping function* $i_m(\cdot) : \mathbb{R}^{d/M} \rightarrow [K]$[1] to encode a vector: $\mathbf{x} \mapsto (i_1(\mathbf{x}_1), \cdots, i_M(\mathbf{x}_M))$, where $\mathbf{x} = \text{concat}(\mathbf{x}_1, \cdots, \mathbf{x}_M)$, and $\mathbf{x}_{[M]} \in \mathbb{R}^{d/M}$. A *Quantizer* q_m is defined as $q_m(\mathbf{x}) = \mathbf{c}_m(i(\mathbf{x}))$, meaning the m-th subspace of $\mathbf{x}$ is approximated as $q_m(\mathbf{x})$ for latter use. Product quantization minimizes the *quantization error*, which is defined as

$$Q = \frac{1}{N} \sum_{\mathbf{x} \in \mathcal{X}} ||\mathbf{x} - [q_1(\mathbf{x}), \cdots, q_M(\mathbf{x})]^T||^2. \quad (2)$$

Minimizing Eqn.2 involves performing classical K-means clustering algorithm on each of the M subspaces. Afterwards, one can use *asymmetric distance computation* (ADC) [3] to greatly accelerate distance computation: Given a query vector $\mathbf{q}$, we first split it to M subspaces: $\mathbf{q} = \text{concat}(\mathbf{q}_1 \cdots \mathbf{q}_M)$. We then compute and store $\|\mathbf{q}_m - \mathbf{c}_m(k)\|^2$ for $k \in [K]$ and $m \in [M]$ in a look-up table. Finally, for each $\mathbf{x} \in \mathcal{X}$, the distance between $\mathbf{q}$ and $\mathbf{x}$ can be efficiently approximated using only $(M-1)$ floating-point additions and M table-lookups with the following equation:

$$\|\mathbf{q} - \mathbf{x}\|^2 \approx \sum_{m=1}^{M} \|\mathbf{q}_m - \mathbf{c}_m(i_m(\mathbf{x}))\|^2 \quad (3)$$

A number of improvements for PQ have been put forward, e.g., *Optimized product quantization* (OPQ) [22] *additive quantization* (AQ) [23], and *generalized residual vector quantizaiton* (GRVQ) [24], etc. Vector quantization methods also allow a number of efficient non-exhaustive search methods including Inverted File System[3], Inverted Multi-index[25].

Recent studies on vector quantization focus on vector quantization for semantic similarities, for example, supervised

[1] $[K]$ denotes $\{1, 2, 3, \ldots, K\}$