Deep Semantic Hashing of 3D Geometric Features for Efficient 3D Model Retrieval

Takahiko Furuya
University of Yamanashi
4-3-11 Takeda, Kofu-shi
Yamanashi-ken, 400-8511, Japan
takahikof AT yamanashi.ac.jp

Ryutarou Ohbuchi
University of Yamanashi
4-3-11 Takeda, Kofu-shi
Yamanashi-ken, 400-8511, Japan
ohbuchi AT yamanashi.ac.jp

ABSTRACT

As the scale of 3D model databases increase, speed, in addition to accuracy, of its search becomes very important. One way to achieve fast search is to use a compact 3D shape feature whose cost of comparison is very small. Binarization of a real-valued feature vector, via hashing, is a way to obtain such compact feature vector. Previous algorithms for producing binarized 3D model features via hashing consisted of two disconnected stages; handcrafted real-valued 3D shape feature extraction followed by hashing into binary code. This compartmentalized approach, however, leads to less-than optimal binary codes, as these two stages are optimized independently. This paper proposes a deep semantic hashing algorithm called *Binarized Deep Local feature Aggregation Network* (*BDLAN*) which jointly optimizes real-valued feature extraction per 3D model and its banarization via hashing. BDLAN training minimizes quantization error caused by binarization. However, this constraint alone often maps real-valued features to their nearest binary codes in Hamming space, which are nonoptimal local minima. To alleviate the issue, we add a simple regularization called *Probabilistic Bit Inversion* (*PBI*) of binary codes. Experimental evaluation of the proposed algorithms demonstrates superior efficiency and competitive accuracy to the existing 3D model retrieval algorithms employing real-valued features.

CCS CONCEPTS

• **Information systems** → **Information retrieval** → Search engine architectures and scalability

KEYWORDS

3D model retrieval, feature hashing, binary codes, deep learning.

CGI '17, June 27-30, 2017, Yokohama, Japan
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5228-4/17/06…$15.00
http://dx.doi.org/10.1145/3095140.3095148

1 INTRODUCTION

3D shape model has been rapidly growing in numbers, prompted in part, by proliferation of low-cost 3D range scanners and 3D printers. A large number of 3D models are stored in a database to reuse 3D CAD models for mechanical design, to analyze 3D structure of molecules for medicine, to diagnose diseases from 3D MRI scans of organs, etc. Effective management of these 3D models requires a technology of *shape-based 3D Model Retrieval* (*3DMR*) that could handle a large-scale database containing a large number, e.g., 1M or more, of 3D models.

Various approaches aiming at efficient 3DMR have been proposed (e.g., [19][22][24]). A popular approach is to "compress" 3D model features, e.g., via dimensionality reduction [13][24]. Dimensionality reduction could yield low-dimensional and salient features. However, dimensionality reduction alone is not sufficient to attain enough scalability. Resulting feature vector is not compact enough if a 32bit floating point number is used for each dimension. In addition, distance computation among features, by using L_p-norm, information divergence, etc., can be costly.

A way to obtain more compact and quick-to-compare feature is by hashing of a real-valued feature vector into a binary-valued vector (e.g., [19][31]). This approach describes each 3D model by a binary vector, or code, in which each element is either "0" or "1". As a binary code, it is memory-efficient. In addition, a distance between a pair of binary codes in Hamming space can be computed very efficiently by an Exclusive-OR and a "1" population count operations found in vector instruction set of a modern CPU. However, binary codes produced by hashing real-valued vectors tended to have lower retrieval accuracy than the original. We observe that a compartmentalized strategy that combines a handcrafted feature with an independently developed hashing algorithm (e.g., [19][31]) leads to a binary feature vector nonoptimal in terms of accuracy.

Another approach to retrieval efficiency is by reduction in number of features per 3D model. If a 3D model is described by a set of features, which are typically local and low-level, a small subset of it can be selected to describe a 3D model [22] or the set of features can be aggregated into a feature per 3D model [32]. Aggregation local features has been quite popular. A set of local features can describe diverse partial shapes of a 3D model [5], and an aggregated feature can be robust against non-rigid deformation of the 3D models [36]. Also, the aggregated feature is much less

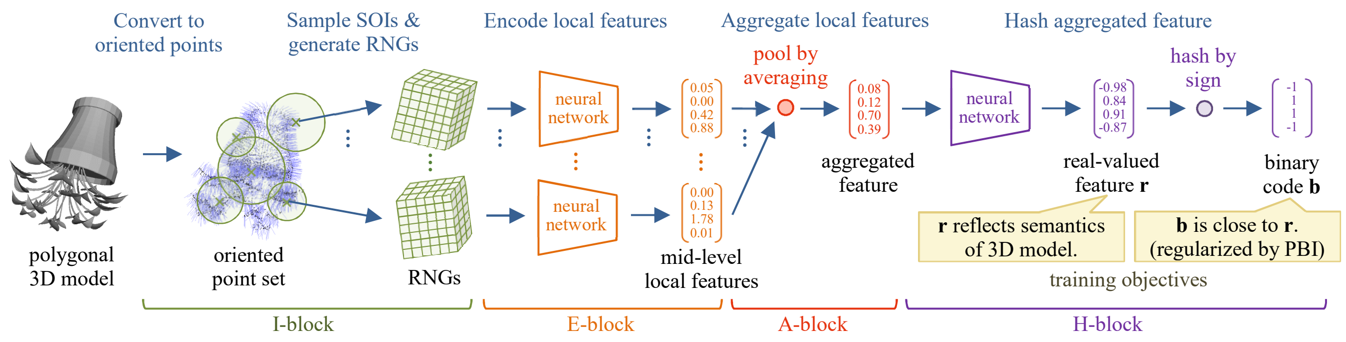

Figure 1: BDLAN extracts and then aggregates local 3D geometric features to produce a semantic binary code per 3D model. During training of BDLAN, several bits of a binary code are stochastically inverted for regularization.

costly to store and compare than the set of local features. Still, an aggregated feature is often very high-dimensional (~100k), real-valued vector (e.g., [13][28]) and is too costly for scalable 3DMR.

Our goal is to obtain compact and discriminative binary features of 3D models suitable for efficient and accurate shape-based 3D model retrieval. We approach this goal by using a processing pipeline derived by using an *integrated,* not compartmentalized, optimization approach driven by a labeled set of training 3D models.

The proposed integrated feature extraction and hashing pipeline is a Deep Neural Network (DNN) called *Binarized Deep Local feature Aggregation Network* (*BDLAN*). BDLAN is designed to produce an accurate binary code per 3D model that is robust against 3D rotation and non-rigid deformation of the 3D model. BDLAN performs the following three steps in a single deep architecture; (1) extraction of a set of rotationally invariant, local 3D geometric features, (2) aggregation of the local features to a feature per 3D model, and (3) hashing of the aggregated feature. BDLAN is built upon the recently proposed Deep Local feature Aggregation Network (DLAN) [30] that produces real-valued and semantic features for 3DMR. We inject binary constraints into the output of DLAN to obtain semantic, salient, binarized, compact feature vector, or codes. Compact binary codes (e.g., 64 bits per code) generated by BDLAN enable efficient 3DMR both in terms of memory footprint and computational time. To our knowledge, BDLAN is the first deep learning-based feature extraction and hashing algorithm to produce binary codes for scalable 3DMR.

As with DNN-based hashing algorithms for 2D images (e.g., [10][12][14]), the objective of BDLAN training is to obtain binary codes that preserve, as much as possible, distances among real-valued features in their binarized counterparts. Such binary features could be learned by minimizing error caused by binarization, or quantization, of the real-valued features ([10][12][14][16]). However, minimizing quantization error alone could result in a deep hashing neural network that produces suboptimal binary codes. In an effort to minimize quantization error, the optimization could get stuck at local minima at which real-valued vector are close to subopotimal binary codes.

To alleviate this issue, we introduce *Probabilistic Bit Inversion* (*PBI*), a simple regularization technique. During training of BDLAN, PBI randomly inverts several bits of the binary feature. PBI is expected to help the training escape from the local minima by stochastically quantizing a real-valued feature onto a certain stochastic neighborhood, not a point, in Hamming space.

Experiments using 3D model retrieval benchmark databases show that 3DMR using binary codes generated by BDLAN, trained by using PBI, retains retrieval accuracy comparable to the original, real-valued feature vector by DLAN. Using binary codes by BDLAN, however, makes the processing much less memory intensive and much faster.

Contributions of this paper can be summarized as follows;

- Proposition of a deep semantic hashing algorithm called BDLAN for scalable 3DMR.
- Proposition of a regularization algorithm called PBI designed specifically to learn salient binarized feature, or code.
- Evaluation of BDLAN and PBI by using 3DMR benchmarks.

Rest of this paper is organized as follows. We review related work in the next section. In Section 3, the proposed algorithms are described. Experiments and their results are presented in Section 4, followed by conclusion and future work in Section 5.

2 RELATED WORK

2.1 Feature Hashing for 3D Model Retrieval

Previous algorithms for hashing 3D model feature combine handcrafted, real-valued features with hashing algorithms based on shallow architectures ([19][25][29][31]). [19] employs a single layer neural network to hash handcrafted features extracted from 3D models. The single layer neural network is trained by using pairwise constraints that define two 3D models to be semantically similar or dissimilar. [31] adopts unsupervised feature hashing to efficiently perform part-based 3DMR where a query is a partial 3D shape. [31] describes each part of a 3D model by a compact binary code. For the task of 3DMR queried by a hand-drawn sketch, [25] uses simple thresholding to binarize real-valued features of 2D images, while [29] employs cross-modal manifold learning that embeds features of sketches and features of 3D models into their common Hamming space.

However, the approaches mentioned above do not necessarily yield binary codes optimal for 3DMR since they combine handcrafted features having limited descriptive power with learning models having inadequate expressive power. Recently, deep learning has been introduced to 3DMR for extracting accurate, real-valued feature per 3D model. A variety of DNNs tailored to 3D model have been proposed, accepting a variety of 3D shape representations including voxels [35], point set [1][30], manifold mesh [15], and rendered 2D views [11][26]. To our knowledge, however, no prior work has studied feature hashing by using deep architecture for scalable 3DMR.

2.2 Probabilistic Approach to Regularized Training of Deep Neural Network

Introducing randomness to training DNN could regularize the training to avoid overfitting. Dropout [20] and DropConnect [17] are widely-used algorithms that randomly prune neuronal activations and neuronal connections, respectively. [9] shows that adding Gaussian noise to connection weights improves generalization ability of neural networks for regression and classification tasks. [21] proposes a denoising autoencoder, which gains robustness against Gaussian noise or masking noise by adding such a noise to the input of the autoencoder. [4] adds Gaussian noise to gradient of objective function computed during back propagation. Noisy gradient prevents the training from overfitting and enables effective training of a very deep neural network. [6] adds noise to activation functions when training gets stuck due to saturated gradient of the activation functions.

The proposed PBI can also be classified into the probabilistic approach for regularizing DNN training. While we applied PBI to learn binary codes for 3D models, it is general enough to be applied to other deep hashing algorithms, e.g., ones for 2D image features.

3 PROPOSED ALGORITHM

3.1 Overview of the Proposed Algorithm

The proposed algorithm consists of two parts, the *Binarized Deep Local feature Aggregation Network* (*BDLAN*) architecture for extracting salient binary codes of 3D models, and a novel regularization strategy designed specifically for binary-valued hashing called *Probabilistic Bit Inversion* (*PBI*). Figure 1 shows the processing pipeline of the proposed algorithm. BDLAN takes as its input a 3D model represented as an oriented point set and produces a compact binary code of the 3D model. By using Hamming distance, which can be computed very efficiently, computing retrieval ranks of 3D models can be very quick.

Like its predecessor DLAN [30], BDLAN generates 3D model features that have invariance against similarity transformation, i.e., a combination of translation, uniform scaling, and rotation in 3D space. A BDLAN feature is also robust against non-rigid deformation as it uses a set of local features, each of which has invariance to rotation and translation, to describe a 3D model.

3.2 Extracting Binary Codes using BDLAN

BDLAN comprises four sub-blocks, that are, I-block to compute low-level local geometrical feature to be input to BDLAN, E-block to encode the local features, A-block to aggregate the encoded local features, and H-block to hash the aggregated features into binary codes (Figure 1). Structures of the first three blocks of BDLAN, I-block, E-block, and A-block, are identical those of DLAN. The last, H-block, is newly designed for feature hashing.

I-block: Given a 3D model represented as a 3D polygonal mesh, it is first converted into an oriented point set by using the algorithm by Osada et al. [23]. We randomly and uniformly sample 3k points on the surface of the 3D model. Orientation of each point corresponds to the normal vector of the triangle at which the point is sampled. We then sample 100 Spheres-Of-Interest (SOIs) with random position and random scale from the oriented point set. Each SOI is rotationally normalized by applying Principal Component Analysis (PCA) to the points enclosed by the SOI. The rotationally normalized SOI is spatially divided into 3D cells by using a 3D grid. Oriented points within each cell are described by a 10-dim. POD feature [27] to form a Rotation Normalized Grid (RNG) for the SOI.

E-block: Each of the 100 RNGs representing the 3D model is independently fed into the E-block to encode the RNG into a mid-level local 3D geometric feature. E-block has two 3D convolution layers and subsequent three fully-connected layers with ReLU activation function [2]. The E-block generates a set of 512-dim., real-valued local geometrical features of the 3D model.

A-block: The set of 100 mid-level local features from E-block is aggregated to a single feature having 512 dim. per 3D model via average pooling.

H-block: H-block first dimension reduce the aggregated feature above to a 256-dim. real-valued feature via a fully-connected layer. Then, a subsequent hash layer binarizes the dimension reduced feature. The hash layer is also fully-connected but uses hyperbolic tangent activation function to limit the activation to a range (−1, +1). The activated real-valued feature (denoted as **r**) is binarized by its sign to generate a binary code (denoted as **b**) of the 3D model. When computing Hamming distance, -1 and +1 of the hashing result are mapped to binary digits 0 and 1, respectively. The number of bits N_b for binary codes corresponds to the number of neurons (e.g., 64) in the hash layer.

3.3 Training BDLAN with PBI

To learn accurate semantic binary codes, a BDLAN is trained so that it could correctly predict object category labels of training 3D models while minimizing quantization error due to binarization of the real-valued 3D model features produced at the output end of the BDLAN. Such an objective L can be formalized as follows;

$$L = L_c + \alpha L_q \tag{1}$$

where L_c is cross-entropy loss that defines classification error of category labels and $L_q = \|\mathbf{r} - \mathbf{b}\|^2$ indicates quantization error between real-valued feature **r** and binary code **b**. To compute cross-entropy loss, we append a classification layer with softmax function behind the hash layer during training of a BDLAN. The hyper-parameter α balances L_c and L_q. We fix α to 1.0 in the experiments.

PBI regularization tries to prevent the training of binary codes to fall into poor local minima. Specifically, for each time BDLAN quantizes the real vector **r** to the binary code **b** during training, PBI randomly inverts up to P % of N_b bits of **b**. For example, when $N_b = 128$ and $P = 10\%$, PBI inverts the maximum of 12 bits randomly chosen from the 128 bit of the code. This operation means that **r** could be quantized not only into its nearest binary code but also to binary codes within its stochastic proximity. The bit-inverted binary code **b** is used to compute the loss described in equation 1. Note that PBI is performed only in the training phase. After training, the binary code of the 3D model is computed by using BDLAN without PBI as described in Section 3.2.

To effectively train BDLAN by using small number (~10k) of labelled 3D models, we adopt two-stage training of BDLAN. That is, the first stage trains only E-block by using a large number of (300k in this paper) labelled RNGs. Each RNG is sampled from either of labelled 3D models in the training dataset and category label for the RNG is identical to that for 3D model from which the RNG is sampled. We append a classification layer at the output end of E-block and train the E-block so that cross-entropy loss could be minimized. The second stage trains the whole BDLAN by using the labelled 3D models to minimize the entire loss (equation 1). EE-block inherits the result of the first stage while weights of H-block are randomly initialized. PBI is used at the second stage training. For both the first and the second stage training, we use Adam [7] with mini-batch size = 32. Training is iterated for 200 epochs and 20 epochs at the first and second stage training, respectively.

4 EXPERIMENTS AND RESULTS

4.1 Experimental Setup

Benchmark datasets: Efficiency and accuracy of the proposed algorithm are evaluated by using two standard benchmark datasets for 3DMR, i.e., ModelNet40 and ModelNet10 [35]. Figure 2 shows examples of 3D models contained in the datasets. ModelNet40 has 9,843 3D models for training and 2,468 3D models for testing. These 3D models are classified into 40 semantic object categories such as airplane, car, plant, sofa, etc. On the other hand, ModelNet10, which is a subset of ModelNet40, contains 3,991 3D models for training and 908 3D models for testing. The 3D models in ModelNet10 are categorized into 10 indoor objects, e.g., chair, table, bed, etc. For both datasets, BDLAN is trained by using 3D models in the training set and retrieval accuracy is evaluated by using those in the test set. We use Mean Average Precision (MAP) [%] as accuracy index. Since the proposed algorithm is influenced by randomness caused by initialization of DNN parameters, PBI, etc., every experiment is performed three times and their average MAP score will be reported.

Competitors: We compare performance of BDLAN with the following four algorithms;

- DLAN [30] extracts real-valued, non-negative feature per 3D model. This is the algorithm on which BDLAN is based..
- DLAN+LSH hashes the real feature vectors obtained from DLAN by using Locality Sensitive Hashing [3].
- DLAN+ITQ uses Iterative Quantization [34] for hashing.
- DLAN+AQBC uses Angular Quantization-based Binary Codes [33] which is designed to hash non-negative feature vectors.

Figure 2: Examples of 3D models in benchmark datasets.

Hash functions for ITQ and AQBC are learned by using DLAN features extracted from 3D models in the training set. LSH doesn't require learning as it is a data-independent hashing algorithm. To generate ranking results, DLAN uses Cosine distance while the other three algorithms use Hamming distance. We also compare accuracy of BDLAN against the existing 3DMR algorithms using handcrafted [8][28] or deep learning-based [11][26][35], real-valued feature of 3D model.

We use $N_b = 64$ and $P = 30\%$ for BDLAN unless otherwise stated. We used TensorFlow library [18] to implement BDLAN. All the experiments were conducted on a PC with an *Intel Core i7 6700* CPU, a *NVIDIA GeForce GTX 1080* GPU, and 64GB DRAM.

4.2 Experimental Results

Accuracy of BDLAN: Figure 3 plots retrieval accuracies of the five algorithms against the number of feature dimensions (i.e., N_b for BDLAN). For the four algorithms except for "DLAN+LSH",

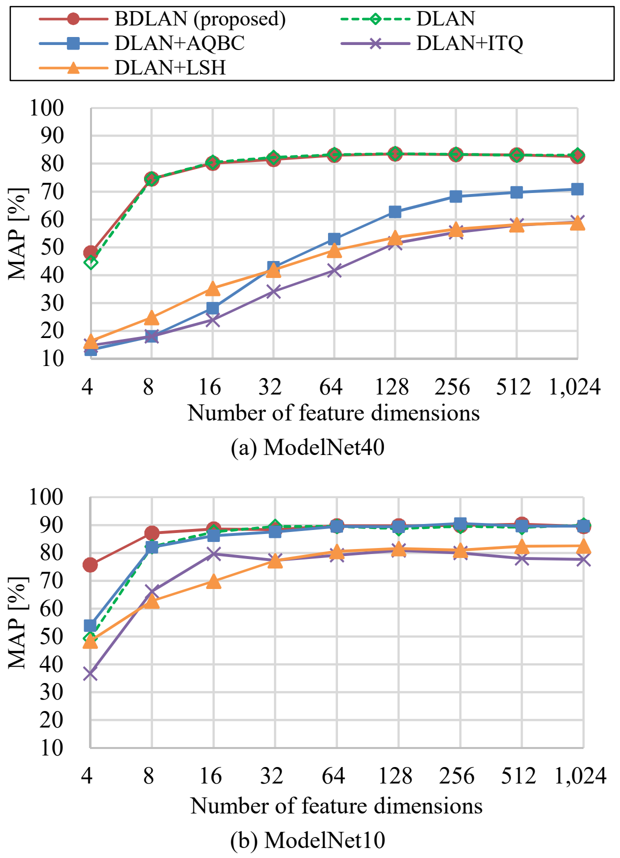

Figure 3: Feature dimensions and retrieval accuracy.

the number of feature dimensions corresponds to the number of neurons at the output layer of the DNN. On the other hand, for "DLAN+LSH", 128-dim. real-valued features are extracted by using DLAN and *N* hyperplanes are randomly generated to yield *N*-bit binary codes. Note that only "DLAN", denoted by dashed lines in Figure 3, uses real vectors while the others use binary codes for retrieval.

For both ModelNet40 (Figure3a) and ModelNet10 (Figure3b), we can observe that "BDLAN" and "DLAN" yield almost the same MAP scores if binary codes with more than 8~16 bits are used. This result indicates that BDLAN generates accurate semantic binary codes of 3D models with low quantization error due to feature hashing. Also, BDLAN shows retrieval accuracies higher than the three compartmentalized hashing algorithms, i.e., "DLAN+ITQ", "DLAN+AQBC", and "DLAN+LSH". Joint optimization of extracting semantic real features and their hashing would have a positive effect to obtain binary codes for accurate 3DMR. "DLAN+AQBC" performs well for the simple ModelNet10 dataset but suffers for ModelNet40 dataset having more object categories.

Table 1 compares retrieval accuracies of the six state-of-the-art 3DMR algorithms including the proposed BDLAN. All the existing algorithms in Table 1 employs real-valued features for retrieval. Note that accuracies of DLAN in Table 1 differs from the original paper [30] since we used our own implementation of DLAN for the experiment. BDLAN yields competitive MAP scores to the other real feature-based algorithms. Accurate 3DMR can be performed also by using binary codes obtained from DNN. Efficiency of BDLAN will be evaluated in the later subsection.

Table 1: Comparison of retrieval accuracy.

Algorithms	MAP [%]	
	ModelNet40	ModelNet10
LFD [8]	40.9	49.8
SV-DSIFT [28]	58.1	68.0
3D ShapeNets [35]	49.2	68.3
MVCNN [11]	79.5	—
GIFT [26]	81.9	**91.1**
DLAN [30]	**83.3**	89.5
BDLAN	**83.1**	**89.8**

Effectiveness of PBI: We investigate the efficacy of PBI for learning accurate binary codes of 3D models. Figure 4 shows retrieval accuracies of BDLAN with PBI (*P*=30%) or without PBI. For both ModelNet40 and ModelNet10 datasets, PBI slightly but consistently improves MAP scores of BDLAN. The gain of retrieval accuracy would be the result from regularization of binary code learning. PBI would help the training of BDLAN find a solution having better generalization capability.

Figure 5 shows influence of the hyper-parameter *P* for PBI on retrieval accuracy. For the experiments, we fixed the number of bits for binary codes N_b to 64. In the figure, *P* = 0% means PBI is not performed during the training of BDLAN. Interestingly, in Figure 5, we can observe that *P* doesn't have a large impact on retrieval accuracy. Very slight peaks can be found at around P = 30%. When P = 30%, at most 19 bits of a 64-bit code could be inverted.

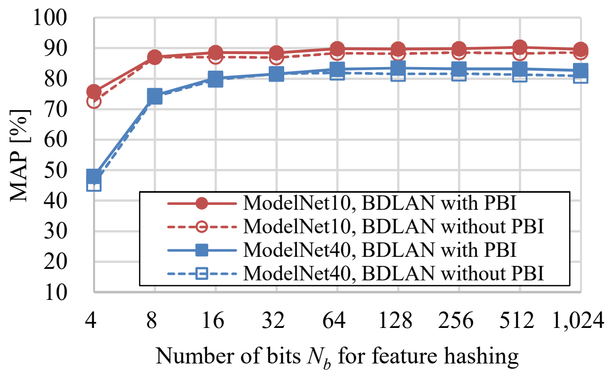

Figure 4: Effectiveness of Probabilistic Bit Inversion (PBI).

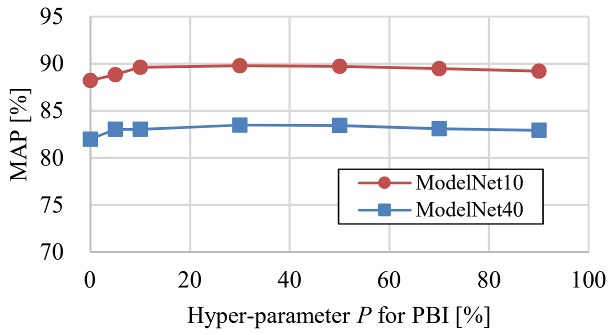

Figure 5: Probability of bit inversion and retrieval accuracy.

Efficiency of BDLAN: This subsection evaluates efficiency of BDLAN both for training and retrieval. As we described in Section 3.3, BDLAN adopts two-step training. The first step (i.e., training of E-block) took about 2~3 days and the second step (i.e., training of whole BDLAN) took about 2 hours.

We wanted to evaluate retrieval scalability of BDLAN. Unfortunately, however, the number of 3D models in the test set of ModelNet40 (i.e., 2,468) is too small to evaluate scalability. We therefore duplicated the 2,468 features of the 3D models up to 100M features to simulate a large-scale 3D model database. Table 2 summarizes retrieval efficiency of BDLAN and DLAN for the simulated large-scale database. In the table, "Feat." indicates computation time for extracting a feature from a query 3D model, "Dist." means time for computing distances among the feature of the query and the 100M features in the database, and "Memory footprint" is spatial cost to store 100M features of the 3D models. We used 64-dim. real-valued features for DLAN and 64 bit binary codes for BDLAN.

Table 2 clearly shows that BDLAN is quite a bit faster than DLAN. Computing Hamming distances among binary codes drastically reduces computational time per retrieval.

BDLAN is memory efficient as well; it requires less than 1 GByte for as many as 100M 3D models. DLAN, on the other hand, requires 24 GBytes as it uses a 32 bit floating-point number per feature dimension.

Table 2: Comparison of retrieval efficiency.

Algorithms	Per-query time [s]			Memory footprint [GB]
	Feat.	Dist.	Total	
DLAN [30]	0.08	11.50	11.58	24.41
BDLAN	0.08	**0.37**	**0.45**	**0.76**

4 CONCLUSION AND FUTURE WORK

Explosive increase in number of 3D models in recent years requires scalable algorithms for shape-based 3D model retrieval (3DMR). Nevertheless, few prior work has been studied 3D model feature that satisfies both high efficiency and high accuracy. In this paper, we proposed a deep semantic hashing algorithm intended for scalable 3DMR. The proposed algorithm called Binarized Deep Local feature Aggregation Network (BDLAN) extracts compact semantic binary code per 3D model. Compared to real-valued feature, binary code is much efficient; binary codes can be compared more quickly by using Hamming distance and they require much less memory. To obtain salient binary codes, BDLAN embeds features of 3D models into the Hamming space that well reflects semantics of the 3D models by regularizing the training by using a novel, binary-code specific regularization algorithm called Probabilistic Bit Inversion (PBI). The experiments demonstrated superior efficiency and competitive accuracy of BDLAN compared to the state-of-the-art 3DMR algorithms.

Future work includes evaluation of BDLAN on larger-scale databases having more 3D models classified into more diverse object categories. Also, efficacy of PBI regularization should be evaluated under different settings and scenarios, e.g., by deep hashing of 3D model features using base architectures other than DLAN (e.g., [11] [26][35]), or by deep hashing of 2D image features (e.g., [10][12][14]).

ACKNOWLEDGMENTS

This research is supported by *JSPS Grant-in-Aid for Young Scientists* (*B*) #16K16055.

REFERENCES

[1] A. Garcia-Garcia, F. Gomez-Donoso†, J. Garcia-Rodriguez, S. Orts-Escolano, M. Cazorla, and J. Azorin-Lopez. 2016. PointNet: A 3D Convolutional Neural Network for real-time object class recognition. *Proc. IJCNN 2016*.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. *Proc. NIPS 2012*.

[3] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity Search in High Dimensions via Hashing. *Proc. VLDB 1999*, 518–529.

[4] Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. 2016. Adding Gradient Noise Improves Learning for Very Deep Networks. *Proc. ICLR workshop 2016*.

[5] Bo Li, Yijuan Lu, Chunyuan Li, Afzal Godil, Tobias Schreck, et al. 2014. SHREC' 14 Track: Large Scale Comprehensive 3D Shape Retrieval. *Proc. EG3DOR 2014*, 131–140.

[6] Caglar Gulcehre, Marcin Moczulski, Misha Denil, and Yoshua Bengio. 2016. Noisy Activation Functions. *Proc. ICML 2016*, **48**, 3059–3068.

[7] Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. *Proc. ICLR 2015*.

[8] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. 2003. On Visual Similarity Based 3D Model Retrieval. Computer Graphics Forum, **22**(3), 223–232.

[9] Guozhong An. 1996. The effects of adding noise during backpropagation training on a generalization performance. *Neural Computation*, **8**(3), 643–674.

[10] Han Zhu, Mingsheng Long, Jianmin Wang and Yue Cao. 2016. Deep Hashing Network for Efficient Similarity Retrieval. *Proc. AAAI 2016*, 2415–2421.

[11] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. 2015. Multi-view Convolutional Neural Networks for 3D Shape Recognition. *Proc. ICCV 2015*, 945–953.

[12] Haomiao Liu, Ruiping Wang, and Shiguang Shan. 2016. Deep Supervised Hashing for Fast Image Retrieval. *Proc. CVPR 2016*, 2064–2072.

[13] Hedi Tabia, David Picard, Hamid Laga, and Philippe-Henri Gosselin. 2013. Compact Vectors of Locally Aggregated Tensors for 3D shape retrieval. *Proc. EG 3DOR 2013*, 17–24.

[14] Huei-Fang Yang, Kevin Lin, and Chu-Song Chen. 2017. Supervised Learning of Semantics-Preserving Hash via Deep Convolutional Neural Networks. TPAMI, DOI: 10.1109/TPAMI.2017.2666812.

[15] Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. 2015. Geodesic convolutional neural networks on Riemannian manifolds. *Proc. ICCV Workshop 2015*, 832–840.

[16] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. 2016. Learning Compact Binary Descriptors with Unsupervised Deep Neural Networks. *Proc. CVPR 2016*, 1183–1192.

[17] Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. 2013. Regularization of Neural Networks using DropConnect. *JMLR: Workshop and Conference Proceedings*, **28**(3), 1058–1066.

[18] Martín Abadi et al. 2016. TensorFlow: a system for large-scale machine learning. *Proc. OSDI 2016*, 265-283.

[19] Michael M. Bronstein, Alexander M. Bronstein, Fabrice Michel, Nikos Paragios. 2010. Data Fusion through Cross-modality Metric Learning using Similarity-Sensitive Hashing. *Proc. CVPR 2010*, 3594–3601.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *JMLR*, **15** (2014), 1929–1958.

[21] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *JMLR*, **11** (2010), 3371–3408.

[22] Philip Shilane and Thomas Funkhouser. 2006. Selecting Distinctive 3D Shape Descriptors for Similarity Retrieval. *Proc. SMI 2006*, DOI: 10.1109/SMI.2006.34.

[23] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. 2002. Shape distributions. *TOG* **21**(4), 807–832.

[24] Ryutarou Ohbuchi, Akihiro Yamamoto, and Jun Kobayashi. 2007. Learning Semantic Categories for 3D Model Retrieval. *Proc. MIR 2007*, 31–40.

[25] Shuang Liang, Long Zhao, Yichen Wei, and Jinyuan Jia. 2014. Sketch-Based Retrieval Using Content-Aware Hashing, *Proc.PCM 2014*, **8879**, 133–142.

[26] Song Bai, Xiang Bai, Zhichao Zhou, Zhaoxiang Zhang, and Longin Jan Latecki. 2016. GIFT: A Real-time and Scalable 3D Shape Search Engine. *Proc. CVPR 2016*, 5023–5032.

[27] Takahiko Furuya and Ryutarou Ohbuchi. 2015. Diffusion-on-Manifold Aggregation of Local Features for Shape-based 3D Model Retrieval. *Proc. ICMR 2015*, 171–178.

[28] Takahiko Furuya and Ryutarou Ohbuchi. 2014. Fusing Multiple Features for Shape-based 3D Model Retrieval. *Proc. BMVC 2014*, http://dx.doi.org/10.5244/C.28.16.

[29] Takahiko Furuya and Ryutarou Ohbuchi. 2014. Hashing Cross-Modal Manifold for Scalable Sketch-based 3D Model Retrieval. *Proc. 3DV 2014*.

[30] Takahiko Furuya and Ryutarou Ohbuchi. 2016. Deep Aggregation of Local 3D Geometric Features for 3D Model Retrieval. *Proc. BMVC 2016*.

[31] Takahiko Furuya, Seiya Kurabe, and Ryutarou Ohbuchi. 2015. Randomized Sub-Volume Partitioning for Part-Based 3D Model Retrieval. *Proc. EG 3DOR 2015*, 15–22.

[32] Yi Liu, Hongbin Zha, and Hong Qin. 2006. Shape Topics: A Compact Representation and New Algorithms for 3D Partial Shape Retrieval. *Proc. CVPR 2006*, **2**, 2025–2032.

[33] Yunchao Gong, Sanjiv Kumar, Vishal Verma, and Svetlana Lazebnik. 2012. Angular quantization-based binary codes for fast similarity search. *Proc. NIPS 2012*, 1196–1204.

[34] Yunchao Gong, Svetlana Lazebnik, and Albert Gordo. 2013. Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval. TPAMI, **35**(12), 2916–2929.

[35] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3D ShapeNets: A Deep Representation for Volumetric Shapes. *Proc. CVPR 2015*.

[36] Zhouhui Lian, Jun Zhang, et al. 2015. SHREC'15 Track: Non-rigid 3D shape retrieval. *Proc. EG3DOR 2015*, 107–120.