Multimedia Tools and Applications (2019) 78:35157–35178
https://doi.org/10.1007/s11042-019-08078-y

Feature set aggregator: unsupervised representation learning of sets for their comparison

Takahiko Furuya [1] · Ryutarou Ohbuchi [1]

Received: 10 August 2018 / Revised: 5 June 2019 / Accepted: 2 August 2019 /
Published online: 20 August 2019

Abstract
Unsupervised representation learning of unlabeled multimedia data is important yet challenging problem for their indexing, clustering, and retrieval. There have been many attempts to learn representation from a collection of unlabeled 2D images. In contrast, however, less attention has been paid to unsupervised representation learning for *unordered sets of high-dimensional feature vectors*, which are often used to describe multimedia data. One such example is set of local visual features to describe a 2D image. This paper proposes a novel algorithm called Feature Set Aggregator (FSA) for accurate and efficient comparison among sets of high-dimensional features. FSA learns representation, or embedding, of unordered feature sets via optimization using a combination of two training objectives, that are, set reconstruction and set embedding, carefully designed for set-to-set comparison. Experimental evaluation under three multimedia information retrieval scenarios using 3D shapes, 2D images, and text documents demonstrates efficacy as well as generality of the proposed algorithm.

Keywords Unsupervised learning · Feature aggregation · Set-to-set comparison · Multimedia information retrieval · Autoencoder · Neural network

1 Introduction

Recent advances in deep learning have achieved great successes in many computer vision tasks. The successes, especially for 2D images, are attributed in large part to availability of labels required to effectively train deep neural networks. However, in practice, many multimedia data collections are often left unlabeled due to high cost of manually annotating large

✉ Takahiko Furuya
takahikof@yamanashi.ac.jp

Ryutarou Ohbuchi
ohbuchi@yamanashi.ac.jp

[1] University of Yamanashi, 4-3-11 Takeda, Kofu-shi, Yamanashi-ken 400-8511, Japan

number of data items. Motivated by this situation, unsupervised representation learning, which attempts to obtain meaningful features from unlabeled dataset, has been gaining much attention for indexing, clustering, or retrieval of multimedia data without labels.

A multimedia datum, that is, a 2D image, a 3D shape, or a text document, can often be represented by *an unordered set of high-dimensional feature vectors*. For example, a 2D image is often represented as a set of local visual features (e.g., SIFT [40]). A 3D shape is represented as a set of local geometric features (e.g., [12]) or 3D point set (e.g., [6]). Also, Bag-of-Words model treats a document as a set of words where each word is described by a high-dimensional vector (e.g., [62]). Such unordered sets that ignore spatial relations among parts of the datum are robust against variation of context, for example, varied poses of articulated objects or swapped phrases in a sentence.

This paper addresses the problem of Set-to-Set Comparison (SSC) that measures distances among unlabeled data, each of which is described by a set of high-dimensional feature vectors. Achieving accuracy and efficiency for SSC under unsupervised setting is quite challenging. Earth Mover's distance [56] or Hausdorff distance [22] have been conventionally employed for SSC. However, these distances are sensitive to outlier features and suffer from high cost of finding element-to-element correspondences between two sets.

To improve accuracy and efficiency of SSC, feature aggregation algorithms including Bag-of-Features [8] and its variants (e.g., [54, 66]) have been proposed. Feature aggregation is performed by encoding each feature within the set and then pooling the encoded features to form a single feature vector that succinctly describes the set. Most of the feature aggregation algorithms optimize feature encoding via codebook learning. However, the aggregated features, typically generated by simple sum of the encoded features, are not necessarily optimized for SSC. In addition, aggregated features are often very high-dimensional (e.g., more than 10 k dimensions [54]) and thus are not suitable for efficient multimedia indexing, clustering, and retrieval.

In this paper, towards more accurate and efficient SSC, we propose a novel unsupervised representation learning algorithm dedicated to multimedia data described by unordered feature sets. A neural network, called *Feature Set Aggregator* (*FSA*), aggregates an input feature set to a compact representation for effective SSC. Training of the FSA is more data-driven than the previous SSC algorithms ([8, 54, 66]). To effectively train the FSA, we design two training objectives for SSC. The first objective is *set reconstruction using angular chamfer distance* which aims to learn accurate representations of "whitened" input feature sets. The set reconstruction loss differs from vector reconstruction loss used by standard autoencoders [16]. The second objective is *set embedding using triplets on family of sets* to learn "smooth" embedded feature space appropriate for SSC. Different from the common label-based triplet sampling (e.g., [23]), our approach creates diverse triplets from an unlabeled collection of feature sets.

Efficacy and generality of the FSA are demonstrated via experiments under three multimedia information retrieval scenarios, that are, 3D shape retrieval, 2D image retrieval, and text document retrieval. In majority of the retrieval experiments, the FSA significantly outperforms the conventional SSC algorithms including state-of-the-art feature aggregation algorithms.

Contribution of this paper can be summarized as follows;

- We propose a novel unsupervised representation learning algorithm for comparison among sets, where each set consists of unordered features vectors. The algorithm, called Feature

Set Aggregator, uses a pair of training objectives designed for accurate reconstruction of set and accurate embedding of aggregated features.
- We extensively evaluate the proposed algorithm under three multimedia information retrieval scenarios to demonstrate accuracy, efficiency, and generality of the algorithm.

2 Related work

2.1 Supervised deep learning for unordered Sets

Motivated in part by successes in 2D image recognition, a variety of Deep Neural Network (DNN) architectures for diverse data representation, including those for unordered sets, have been proposed. Zaheer et al. [74] proposed a DNN called DeepSets that processes sets of high-dimensional features. To obtain invariance against permutation of elements in a set, DeepSets performs max-pooling over mid-level features that appear at an intermediate layer of the DNN to form a global feature of the set. Application of DeepSets includes 3D point set classification, set anomaly detection, and others. The DNN proposed by Furuya and Ohbuchi [14] aggregates a set of local geometric features to describe a 3D shape for 3D model retrieval. PointNet [6] and its variant PointNet++ [55], both by Qi et al., enables end-to-end learning of 3D point sets for their classification and semantic segmentation. For the task of person re-identification, Liu et al. [39] proposed a DNN that handles sets comprising 2D images of identical person. In the field of 2D image recognition, Arandjelović et al. [3] and Lin et al. [36] proposed feature aggregation modules called NetVLAD and NeXtVLAD, respectively. These modules treat a feature map activated in a hidden layer of Convolutional Neural Network (CNN) as an unordered set of visual features. The set of visual features is aggregated to a single feature vector for subsequent classification.

Note, however, that all the DNNs described above are trained by using a large collection of labeled data. On the other hand, our approach is unsupervised. That is, representations of unordered feature sets are learned from multimedia data without having labels.

2.2 Unsupervised representation learning for unordered Sets

Learning meaningful feature representation from unlabeled data is challenging since direct supervision can't be used. Thus, surrogate tasks are generally adopted to train DNNs without supervision by labels. Autoencoder [16] and its variants (e.g., [46, 63]) learn feature representation via reconstruction task. An autoencoder consists of a pair of an encoder and a decoder. The encoder transforms an input to its latent representation, or *code*, and the decoder reconstructs the input from the code. Other surrogate tasks to learn visual representation of 2D images include, for example, solving jigsaw puzzle [49], sorting video sequence [30], tracking objects in video [65], inpainting images [53], and generating images [18].

In contrast to unsupervised representation learning for 2D images, there has been little study on representation learning using unlabeled sets of high-dimensional features. Recently, Achlioptas et al. [2] and Yang et al. [73] proposed autoencoders for unordered sets of 3D points to learn 3D shape feature representation. Compared to the autoencoders for 3D point set, our approach is more general. That is, the FSA can handle unordered sets of feature vectors having arbitrary number of dimensions including but not limited to 3.

Meanwhile, unsupervised feature aggregation is widely adopted to obtain a representation for inter-set comparison of an unordered set of features. Bag-of-Features [8], Locality-constrained Linear coding [66], Fisher vector coding [54], and Vector of Locally Aggregated Descriptors [25] are popular feature aggregation algorithms. They learn a codebook containing representative features, or codewords. By using the codebook, each input feature in an input set is encoded into one of the codewords. The encoding is performed by using first-order statistics [8, 66] or higher-order statistics [25, 54] in the input feature space. The encoded features are pooled by summing to form an aggregated feature per set for classification or SSC. More recently, feature aggregation algorithms aiming at more accurate feature encoding [13, 38] or codebook-free aggregation [37, 67] have been proposed. However, these feature aggregation algorithms are not necessarily optimized for SSC. In addition, these methods often yield very high-dimensional aggregated features. In this study, we introduce a neural network to learn representations of feature sets for accurate and efficient SSC than the previous approaches.

In addition to multimedia indexing and retrieval, use cases of aggregated features that describe unordered sets include biometric identification. For example, matching fingerprint images or palmprint images [43] involves comparison among unordered sets of keypoints detected on skin images. In order to secure biometric verification system and protect privacy of users, skin images are often encoded to their feature representations. To achieve accurate and fast verification, previous studies adopt techniques such as feature selection [33], feature compression [41], or feature fusion [32, 34, 35, 42]. Although we don't conduct experiments on fingerprint/palmprint recognition in this paper, our proposed learning algorithm for unordered sets would be applicable to such biometric identification problems that require comparison of unordered sets.

3 Proposed algorithm

3.1 Overview of feature set aggregator

A Feature Set Aggregator (FSA) is a shallow neural network for learning accurate and compact representations of unordered sets of high-dimensional features. Figure 1 shows a processing pipeline of the FSA. We use 3D shapes as the datatype for the explanation here, but the arguments hold for other multimedia data as 2D images and text documents. The FSA is an asymmetric tandem of an encoder and a decoder. The encoder encodes and aggregates a set of (pre-whitened) features via three processing steps. Each input feature is first transformed into a sparse code. The set of sparse-code is then pooled by sum to form a single feature vector per input set. Finally, the pooled feature is embedded, via metric learning, into a feature space suitable for SSC.

The FSA is trained by using a combination of two training objectives. The first objective, set reconstruction, attempts to learn accurate representations of sets. We propose to use angular chamfer distance as the loss function for reconstruction of sets of whitened input features. The second objective, set embedding, is designed to learn accurate distance metric among sets. To train FSA using these objectives, the algorithm generates, from unlabeled training data, a large number of triplets by generating family of sets from input sets.

During the inference, the encoder part only of the FSA is utilized to obtain an embedded feature of an input set. That is, the decoder part of the FSA is not used for feature aggregation

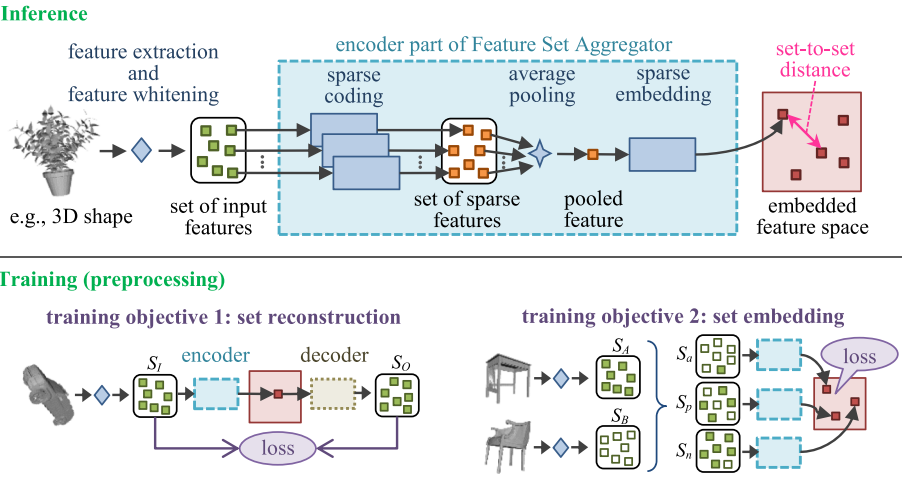


Fig. 1 Proposed Feature Set Aggregator (FSA) effectively and efficiently compares sets of high-dimensional features in the embedded feature space. Embedding is optimized by using the two training objectives, i.e., set reconstruction and set embedding, designed for set-to-set comparison

after the training. SSC is performed by computing Euclidean distances among the embedded features produced by the encoder. Note that, although we evaluate the FSA using multimedia information retrieval scenario, the FSA would be applicable to other tasks such as clustering that requires inter-set distance/ similarity computation.

3.2 Architecture of feature set aggregator

Encoder. The encoder function f transforms a set $S_I = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \in \mathbb{R}^{df}$ of n feature vectors having d_f dimensions into a d_e-dimensional feature vector. For example, these numbers would be $n = 512$, $d_f = 64$ and $d_e = 256$. For simplicity, we assume the cardinality n of each set is equal among all the sets contained in a dataset. As shown in Fig. 1, the encoder comprises three blocks, that are, the sparse coding block, the average pooling block, and the sparse embedding block. The sparse coding block has an architecture similar to an encoder part of k-sparse autoencoder [46]. That is, each input feature is transformed into a d_c-dimensional vector via a single fully-connected (FC) layer activated by ReLU [17] function. Then, the vector of activation values is sparsified by keeping the k_c largest activations while setting the rest to zero. Formally, each input feature $\mathbf{x}_i$ is sparsely encoded to $\mathbf{h}_i$ by using the following equation.

$$\mathbf{h}_i = \text{top\_k}(\text{ReLU}(\mathbf{x}_i \mathbf{W}_{\text{sc}} + \mathbf{b}_{\text{sc}})) \tag{1}$$

In Eq. 1, $\mathbf{W}_{\text{sc}}$ is a linear projection matrix having size $d_f \times d_c$ and $\mathbf{b}_{\text{sc}}$ is a d_c-dimensional bias vector. $\mathbf{W}_{\text{sc}}$ and $\mathbf{b}_{\text{sc}}$ are tuned during the training. The function "top_k" chooses the k_c largest values from the vector activated by ReLU. We use $d_c = 2048$ and $k_c = 5$ throughout the experiments unless otherwise stated.

The average pooling block aggregates the n sparse vectors to a single d_c-dimensional vector per set by using the following equation.

$$\mathbf{h}_{\text{pooled}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{h}_i \tag{2}$$

Pooling operation provides for invariance against permutation of elements in the set and reduces both spatial and temporal costs for comparison among sets. We found that, in preliminary experiments, average-pooling performed better than max-pooling.

The sparse embedding block refines the pooled feature $\mathbf{h}_{\text{pooled}}$ to a more compact and accurate feature. Similar to the sparse coding block, we use the following equation to obtain d_e-dimensional vector $\mathbf{f}$.

$$\mathbf{f} = \text{top\_k}\left(\text{ReLU}\left(\mathbf{h}_{\text{pooled}}\mathbf{W}_{\text{se}} + \mathbf{b}_{\text{se}}\right)\right) \tag{3}$$

In Eq. 3, $\mathbf{W}_{\text{se}}$ is a matrix having size of $d_c \times d_e$ and $\mathbf{b}_{\text{se}}$ is a d_e-dimensional vector. The function "top_k" in Eq. 3 extracts the largest k_e activations from the vector produced by ReLU. We set the number of non-zero elements k_e to roughly one tenth of d_e. For example, when we use d_e = 256, k_e is set to 25. Finally, the sparse vector $\mathbf{f}$ is normalized by its Euclidean norm to form an embedded feature of the input set.

Injecting sparseness into the encoder is motivated by the success of feature aggregation algorithms based on sparse coding (e.g., [13, 66]). Sparse coding is expected to enhance saliency of features. Positive impact of sparseness adopted by FSA on SSC accuracy will be demonstrated in the experimental section.

Decoder. The decoder reconstructs the input feature set from its embedding $\mathbf{f}$ produced by the encoder. We use an FC layer without activation function for decoding. As shown in Eq. 4, the d_e-dimensional embedded feature $\mathbf{f}$ is linearly transformed to an nd_f-dimensional vector $\mathbf{r}$. $\mathbf{W}_{\text{d}}$ is a matrix having size of $d_e \times nd_f$ and $\mathbf{b}_{\text{d}}$ is a nd_f-dimensional bias vector. The vector $\mathbf{r}$ is then reshaped to a matrix having size $n \times d_f$ that represents the reconstructed set of n features having d_f dimensions.

$$\mathbf{r} = \mathbf{f}\,\mathbf{W}_{\text{d}} + \mathbf{b}_{\text{d}} \tag{4}$$

Preprocessing. Prior to be passed onto the FSA, each input feature is normalized by using PCA-whitening [24] for fast and effective training. PCA-whitening reduces dimensionality of features and spheres their distribution to have zero-mean, unit-variance, and no correlation among the dimensions. Equation 5 computes the normalized input $\mathbf{x}_{normalized}$ from the input feature $\mathbf{x}$, where $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are mean and standard deviation of $\mathbf{x}$, respectively. $\mathbf{P}$ is a PCA projection matrix learned by 250,000 input features randomly collected from the training dataset. Matrix $\mathbf{P}$ of size $d_{orig} \times d_f$ reduces dimensionality of input feature $\mathbf{x}$ from d_{orig} to d_f. The value d_{orig} depends on the input feature while d_f is set manually. In this study, d_f is set at 64 for most cases in the experiments.

$$\mathbf{x}_{normalized} = \frac{\mathbf{x} - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\mathbf{P} \tag{5}$$

3.3 Effective training of feature set aggregator

The FSA is trained by using a combination of two objective functions, that are, set reconstruction loss L_R and set embedding loss L_E described below. Overall loss for the FSA is a

weighted sum of the two losses; $L = L_R + \alpha L_E$. The hyper-parameter α balances the two terms. Impact of α on retrieval accuracy will be evaluated in the experiments.

Set reconstruction using angular chamfer distance We require a reconstruction loss that satisfies the following two criteria; (1) suitable for comparing sets of whitened features that have sphered distribution, and (2) invariant against permutation of elements both in input sets and reconstructed output sets. To meet these criteria, we propose to use angular chamfer distance shown in Eq. 6 as the set reconstruction loss L_R.

$$L_R = \sum_{i=1}^{N} \left(\frac{1}{n} \sum_{\mathbf{x} \in S_I} \min_{\mathbf{y} \in S_O} \left(1 - \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} \right) + \frac{1}{n} \sum_{\mathbf{y} \in S_O} \min_{\mathbf{x} \in S_I} \left(1 - \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} \right) \right) \tag{6}$$

In Eq. 6, S_I and S_O indicate the input and the output feature set of the FSA, respectively. n is cardinality of the sets and N is the number of sets used for training. While standard chamfer distance usually adopts Euclidean distance to find a pair of elements that are closest across two sets (e.g., [2, 73]), we employ Cosine distance (i.e., angle) as it is a better metric for comparing whitened high-dimensional features having sphered distribution. The experiments will demonstrate validity of choosing the angular chamfer distance.

Set embedding using triplets on family of sets For accurate SSC, sets consisting of similar features should be close to each other, while sets having dissimilar features should be distant from each other, in the embedded feature space. Such embedded feature space is obtained via training by using triplets. We use a triplet (S_a, S_p, S_n), in which the anchor set S_a is considered to be more similar to the positive set S_p than the negative set S_n.

To collect diverse and a large number of such triplets from unlabeled training dataset, we propose a novel triplet generation algorithm based on family of sets (see Fig. 2). Specifically, we first randomly pick two different sets, i.e., S_A and S_B, from the training dataset. Assume that both S_A and S_B have n elements, and importantly, the pair of sets is at least partially distinct. We then generate a family of sets out of the union $S_A \cup S_B$ to be used as either S_a, S_p, or S_n. That is, a generated set $S \subset S_A \cup S_B$ consists of rn elements randomly sampled from S_A and $(1-r)n$ elements randomly sampled from S_B where the ratio r is randomly determined from a uniform distribution in the range [0, 1]. From $S_A \cup S_B$, three member sets of a triplet, S_a, S_p, and S_n, are created by using three different ratios r_a, r_p, and r_n such that $r_a < r_p < r_n$. We form numerous such triplets (S_a, S_p, S_n) as training samples for set embedding.

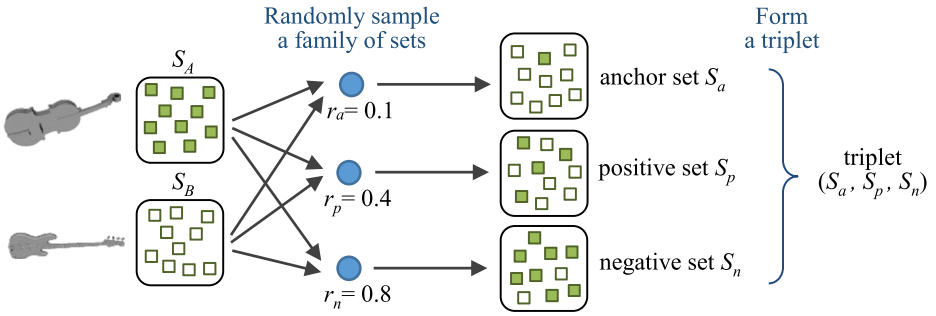


Fig. 2 Low-cost triplet generation using family of sets. We use $n = 10$, $r_a = 0.1$, $r_p = 0.4$, $r_n = 0.8$ for description in the figure

Intuitively, the relation among the sets in the triplet, i.e., "S_a is more similar to S_p than S_n." would hold since S_a is likely to share more elements with S_p than S_n. Our triplet generation algorithm enables us to collect numerous and diverse triplets at very low cost from the unlabeled training dataset. We use the triplet hinge loss [57] shown in Eq. 7 as the embedding loss L_E, where $f(\cdot)$ denotes the encoder function of the FSA. Note that output of the function f corresponds to the embedded feature vector $\mathbf{f}$ described in Section 3.2. Minimizing L_E leads FSA to form the desired embedded feature space where S_a lies closer to S_p than S_n.

$$L_E = \sum_{i=1}^{N} \max\left(0, \left\|f(S_a) - f(S_p)\right\|_2^2 - \left\|f(S_a) - f(S_n)\right\|_2^2 + 1\right) \tag{7}$$

Optimization We employ stochastic gradient descent algorithm with mini batch to train the FSA. Specifically, each parameter of the FSA is iteratively updated by using the following equation.

$$w_i \leftarrow w_i - \eta \frac{\partial L}{\partial w_i} \tag{8}$$

In Eq. 8, $L = L_R + \alpha L_E$ is the overall objective function and w_i indicates each element of projection matrices and bias vectors of FSA, i.e., $\mathbf{W}_{\mathrm{sc}}$, $\mathbf{b}_{\mathrm{sc}}$, $\mathbf{W}_{\mathrm{se}}$, $\mathbf{b}_{\mathrm{se}}$, $\mathbf{W}_{\mathrm{d}}$, and $\mathbf{b}_{\mathrm{d}}$. η is a learning rate. Details on optimization will be described in Section 4.1.

4 Experiments and results

4.1 Experimental setup

We verify effectiveness of the proposed FSA under 3D shape retrieval, 2D image retrieval, and document retrieval scenarios where each datum is described by an unlabeled feature set.

Benchmark datasets We use nine datasets. Table 1, Table 2, and Table 3 summarize statistics of the datasets. For 3D shape retrieval, we use ModelNet10, ModelNet40 [69] and ShapeNetCore55 [5] that consist of polygonal 3D models. We use the training/testing data split provided by [5, 69]. For 2D image retrieval, we use Caltech101 [10], Flowers102 [48], and Animals with Attributes 2 (AwA2) [71] datasets. Although the images of AwA2 have animal class labels and attribute labels, we use only animal class labels as groundtruth for evaluation. For document retrieval, we use News20 [29], Amazon4 [4], and DBpedia14 [31] datasets. News20 includes text messages posted to newsgroups. Amazon4 contains product reviews. DBpedia14 comprises documents that summarize Wikipedia articles. We use only the testing set of DBpedia having 70,000 documents for our experiments. For Caltech101, Flowers102, AwA2, News20, Amazon4, and DBpedia14, we randomly split the whole dataset so that each category contains nearly equal number of samples between the training set and the testing set.

For each dataset, the training set is used to train the FSA, while the testing set is used for evaluation. We use Mean Average Precision (MAP) [%] as a quantitative measure of retrieval accuracy. For each query of category c, retrieval targets of the same category c are treated as

Table 1 Benchmark datasets for 3D shape retrieval

datasets	ModelNet10	ModelNet40	ShapeNetCore55
# of training data	3991	9843	35,764
# of testing data	908	2468	10,265
# of categories	10	40	55
examples of category	bathtub, bed, chair, table, monitor	airplane, chair, plant, car, piano	knife, guitar, table, lamp, sofa, clock

correct while those belonging to categories other than c are treated as incorrect. Since training of the FSA is affected by randomness in its initialization and triplet generation steps, we run every experiment three times and their averaged accuracies are reported. We do not report deviations of accuracy since they are within 1% in MAP for most cases.

Features We use multiple (i.e., two to four) features per retrieval scenario to make the evaluation convincing. Most of the features described below have dimensionality of about 100 to 500. The preprocessing by using PCA-whitening compresses them down to $d_f = 64$ dimensions.

For 3D shapes, we employ RoPS [20], POD [12], and raw oriented points (OP) as local geometric features. A 3D polygon model is first converted into a set of 2048 oriented points by using the algorithm by Ohbuchi et al. [50]. We then extract a set of either 512 RoPS or 512 POD features from the oriented point set to describe shape of the 3D model by the set of features. We also use a set of randomly subsampled 1024 OPs to describe the 3D model where each OP is represented by a 6-dimensional vector (i.e., 3 dimensions for point position and the other 3 dimensions for point orientation). As an exception, we don't reduce the dimensionality of OPs at the preprocessing stage for it already has low dimensionality.

For 2D images, we use Dense SIFT (DSIFT) and convolutional neural network activation (CNNA) as local visual features. From each image, we extract a set of 1024 SIFT [40] features sampled at random positions and at random scales. For CNNA, we use three CNNs, that are, AlexNet [27], VGG19 [60], and ResNet50 [21] trained by using ImageNet corpus [9]. Each CNNA feature is denoted by "CNNA-Alex", "CNNA-Vgg19", and "CNNA-Res50" depending on the CNN used for local feature extraction. CNNA-Alex is extracted as follows. We first randomly crop an image to obtain 20 patches having size 224 × 224 pixels each. Each patch is fed into AlexNet to obtain the *pool5* feature map having 6 × 6 spatial resolution. We partition the feature map into 36 feature vectors. As a result, we obtain a set of 20 × 36 = 720 CNNA-Alex features per image. Similarly, for CNNA-Vgg19 (or CNNA-Res50), we feed 5 patches randomly cropped per image into the CNN. We then obtain the *block4_pool* feature map of VGG19 (or the *activation_48* feature map of ResNet50) having 14 × 14 spatial resolution.

Table 2 Benchmark datasets for 2D image retrieval

datasets	Caltech101	Flowers102	AwA2
# of training data	4367	4119	18,670
# of testing data	4310	4070	18,652
# of categories	101	102	50
examples of category	ant, ferry, crocodile, umbrella, mandolin	lotus, rose, marigold, sunflower, hibiscus	antelope, dalmatian, hippopotamus, wolf

Table 3 Benchmark datasets for document retrieval

datasets	News20	Amazon4	DBpedia14
# of training data	5652	4000	35,000
# of testing data	5662	4000	35,000
# of categories	20	4	14
examples of category	baseball, motorcycle, gun, religion, space	book, DVD, electronics, kitchen	company, animal, film, artist, building

Thus, we obtain $5 \times 14 \times 14 = 980$ CNNA-Vgg19 features (or 980 CNNA-Res50 features) per image.

For text documents, we use word vectors learned by using Latent Semantic Analysis (LSA) [28] or Word2Vec (W2V) [47] algorithms. After removing stop words, each document is described by a set of 200 words randomly sampled from the document. Each word is transformed into a word vector by using the trained model either of LSA [19] or W2V [68].

Competitors We compare the FSA against 13 unsupervised SSC algorithms roughly grouped into four approaches described below.

- **No aggregation.** Chamfer distance (CD) [15] performs SSC by all-pair matching of features between two sets. We also tried SSC using Earth Mover's distance, but we found it was infeasible since computing bijections of feature vectors was temporally inefficient.
- **Aggregation using first-order statistics.** Each feature in a set is encoded based on similarities, or weights, to one or more codewords. The encoded features are pooled, by summing, to form an aggregated feature per set. We adopt Bag-of-Features (BF) [8], Locality-constrained Linear coding (LL) [66], and Database-adaptive k-Sparse Autoencoder (DkSA) [13] as competitors. Additionally, we evaluate sum pooling of raw input features, denoted by "sum". Without elaborate encoding, this probably is the simplest of feature aggregation methods.
- **Aggregation using higher-order statistics.** A set of features is aggregated by using higher-order statistics, such as covariance, of features distributed around codewords. We use Vector of Locally Aggregated Descriptors (VLAD) [25], Fisher Vector coding (FV) [54], Super Vector coding (SV) [70], Fine-residual VLAD (FVLAD) [38], Second-order aggregation with Matrix Power Normalization (SMPN) [37], and Grassmann Pooling (GP) [67] as competitors.
- **Aggregation using neural network.** NetVLAD [3] and NeXtVLAD [36] are feature aggregation modules which can be incorporated in neural networks having various architectures. To verify effectiveness of feature aggregation by FSA, we replace the encoder part of FSA described in Section 3.2 with NetVLAD or NeXtVLAD modules. Note that NetVLAD and NeXtVLAD do not explicitly constrain embedded features to be sparse during their training. This way, we can evaluate the effect sparseness has on accuracy. For a fair comparison, we use the same decoder and loss function for NetVLAD, NeXtVLAD, as well as for FSA.

Implementation details of FSA To train the FSA, we use Adam optimizer [26] with initial learning rate $\eta = 0.001$ to minimize the overall objective function L described in Section 3.3.

Table 4 Values of hyper-parameters for FSA used in the experiments

hyper-parameters	3D shape retrieval	2D image retrievaland document retrieval
(d_c, k_c)	(2048, 5)	(2048, 5)
(d_e, k_e)	(2048, 200) or (256, 25)	(2048, 200) or (256, 25)
α	5 for RoPS and POD feature0.5 for OP feature	1

Each mini-batch contains 8 triplets (i.e., 24 sets) that are created on-the-fly during training. Training is iterated for 100 epochs. We used TensorFlow r1.7 [1] to implement the FSA.

Table 4 summarizes values of the hyper-parameters for the FSA. We used the values in Table 4 unless otherwise noted. As we will show in Section 4.4, the balancing parameter α has an impact on retrieval accuracy. Therefore, in the case of the 3D shape retrieval scenario, we searched for an optimal value of α for each input feature. On the other hand, for 2D image and document retrieval, we fixed α to 1. This is because searching for optimal values of α for all the pairs of input features and datasets in these scenarios is computationally too expensive.

4.2 Comparison with existing unsupervised SSC algorithms

Accuracy Table 5, Table 6, and Table 7 compare retrieval accuracies of 3D shapes, 2D images, and documents, respectively. Each table shows MAP scores for every combination of the input features and the 14 SSC algorithms including the proposed FSA. BF, LL, DkSA, NetVLAD, and NeXtVLAD produce aggregated features having 2048 dimensions, while VLAD, FV, SV, and FVLAD produce aggregated features with nearly 8000 dimensions. For the FSA, we use two embedded feature dimensions, i.e., $d_e = 2048$ and 256, denoted by "FSA-2048" and "FSA-256", respectively, in the tables. We do not report accuracies of CD for the AwA2 dataset and the DBpedia14 dataset. This is because chamfer distance when applied to such a large dataset had excessively high temporal cost.

Table 5 3D shape retrieval accuracy (MAP [%])

Algorithms	ModelNet10			ModelNet40			ShapeNetCore55		
	RoPS	POD	OP	RoPS	POD	OP	RoPS	POD	OP
CD [15]	50.4	50.3	51.5	39.0	38.4	40.8	35.5	35.7	41.6
sum	38.0	42.6	19.4	27.1	30.5	10.4	26.5	31.3	10.9
BF [8]	54.1	57.0	48.6	38.0	40.6	38.1	36.6	40.3	41.1
LL [66]	54.0	55.6	50.8	40.7	42.7	43.4	37.8	41.7	45.4
DkSA [13]	59.6	61.2	53.6	44.3	48.4	38.4	42.8	47.9	38.1
VLAD [25]	50.4	53.7	50.1	37.4	41.8	36.7	34.7	40.2	39.0
FVLAD [38]	50.2	53.8	47.9	38.5	41.5	35.0	35.3	40.0	35.6
FV [54]	51.9	54.1	47.9	39.8	42.6	37.3	36.7	43.7	39.2
SV [70]	52.7	54.1	47.0	40.1	42.5	34.5	37.4	41.6	33.3
SMPN [37]	55.4	58.6	39.2	42.7	46.9	23.1	41.2	46.7	34.5
GP [67]	55.2	57.3	31.5	42.1	45.9	15.8	41.0	45.2	25.8
NetVLAD [3]	58.1	60.4	51.3	41.8	45.1	34.7	42.0	43.3	40.9
NeXtVLAD [36]	60.3	61.4	53.2	42.3	44.8	36.7	40.9	43.6	44.2
FSA-2048 (proposed)	**70.0**	**71.6**	**68.1**	**52.3**	**56.6**	**44.5**	**51.5**	**55.0**	**53.9**
FSA-256 (proposed)	**69.4**	**71.4**	**67.9**	**51.6**	**55.5**	**43.8**	**50.6**	**53.8**	**53.5**

Table 6 2D image retrieval accuracy (MAP [%])

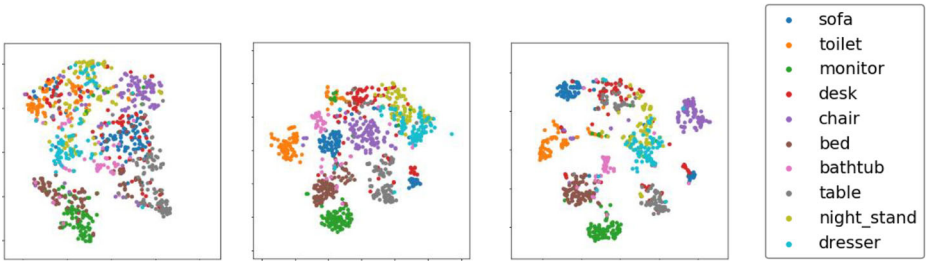
algorithms	Caltech101				Flowers102				AwA2			
	DSIFT	CNNA-Alex	CNNA-Vgg19	CNNA-Res50	DSIFT	CNNA-Alex	CNNA-Vgg19	CNNA-Res50	DSIFT	CNNA-Alex	CNNA-Vgg19	CNNA-Res50
CD [15]	11.5	13.8	8.4	27.0	4.5	5.7	3.6	22.2	–	–	–	–
sum	12.9	41.7	35.9	**65.0**	5.2	27.1	35.1	47.7	3.6	11.0	13.9	54.3
BF [8]	15.3	37.8	33.2	42.4	9.2	28.6	28.4	41.0	4.2	10.8	12.6	26.1
LL [66]	18.0	38.8	35.9	43.6	8.8	32.9	37.7	47.0	4.3	10.7	12.2	27.8
DkSA [13]	20.5	43.1	40.3	**62.0**	10.0	35.1	43.2	53.1	4.4	11.6	14.8	45.2
VLAD [25]	22.4	40.7	42.1	57.3	10.8	**37.9**	50.5	**55.4**	**5.4**	**12.4**	17.2	44.0
FVLAD [38]	23.1	36.6	**43.1**	56.5	12.0	37.5	51.0	54.5	**5.4**	11.8	**18.0**	39.1
FV [54]	22.4	39.6	40.1	55.8	11.1	36.1	47.9	49.6	**5.4**	10.7	16.8	36.5
SV [70]	21.7	40.9	42.8	51.9	11.2	36.3	51.0	50.3	5.3	10.8	17.2	35.3
SMPN [37]	22.5	44.3	40.3	51.2	11.8	37.2	46.4	48.3	4.9	**12.3**	16.0	37.1
GP [67]	20.4	42.1	42.6	49.4	9.1	33.6	45.1	46.4	4.4	11.2	15.5	37.3
NetVLAD [3]	21.3	42.9	40.9	49.6	**13.3**	32.3	47.7	47.7	4.4	10.6	12.6	41.5
NeXtVLAD [36]	21.7	44.2	41.0	47.4	12.8	32.9	45.6	51.1	4.2	11.3	11.3	40.5
FSA-2048 (proposed)	**27.9**	**47.4**	**43.8**	60.0	**15.2**	**40.6**	**58.7**	**59.4**	5.1	10.7	**18.3**	**58.4**
FSA-256 (proposed)	**26.5**	**45.0**	38.5	53.9	11.8	37.5	**52.5**	46.6	5.1	10.4	17.3	**58.3**

Table 7 Document retrieval accuracy (MAP [%])

algorithms	News20		Amazon4		DBpedia14	
	LSA	W2V	LSA	W2V	LSA	W2V
CD [15]	8.1	9.7	42.0	44.7	–	–
sum	12.6	16.4	51.6	55.9	57.0	60.6
BF [8]	13.0	11.1	47.6	45.5	43.5	42.3
LL [66]	12.9	11.9	48.3	46.5	50.0	46.9
DkSA [13]	14.3	12.6	47.0	42.9	37.9	34.6
VLAD [25]	14.0	12.9	37.4	36.3	28.9	41.3
FVLAD [38]	8.7	8.6	35.1	36.4	32.2	39.0
FV [54]	11.4	11.1	44.8	37.3	36.1	39.4
SV [70]	12.8	12.7	48.7	37.2	35.2	40.9
SMPN [37]	**21.7**	16.8	61.2	55.8	56.7	56.4
GP [67]	17.1	14.3	59.8	55.8	41.5	35.7
NetVLAD [3]	10.7	20.1	50.0	53.2	58.9	65.8
NeXtVLAD [36]	12.9	20.0	52.4	55.1	57.2	56.7
FSA-2048 (proposed)	**23.5**	**29.0**	**61.4**	**64.8**	**66.3**	**69.9**
FSA-256 (proposed)	18.9	**28.0**	**61.5**	**65.3**	**64.2**	**66.3**

The results in Table 5, Table 6, and Table 7 demonstrate high accuracy as well as high generality of the FSA. While not true in all the cases, FSA-2048 significantly outperforms the other SSC algorithms for diverse combination of benchmark datasets and input features. The proposed neural network architecture and its training objectives of the FSA appears to learn better representation of unordered feature sets for SSC. In many cases, more compact FSA-256 still yields accuracies superior or comparable to its competitors. On the other hand, the existing feature aggregation algorithms show limited generality compared to the FSA. For example, VLAD and FVLAD perform well in 2D image retrieval, but their accuracies are lower than those of the simple sum aggregation in the most cases of text document retrieval.

Comparison of FSA-2048 with NetVLAD and NeXtVLAD verifies effectiveness of the embedding function of FSA. FSA-2048 consistently outperforms NetVLAD and NeXtVLAD for all the datasets in Table 5, Table 6, and Table 7. The dimensionality of these three features are identical at 2048. As we will show in Section 4.4, introducing sparseness to feature coding and feature embedding of FSA has positive impact on retrieval accuracy.

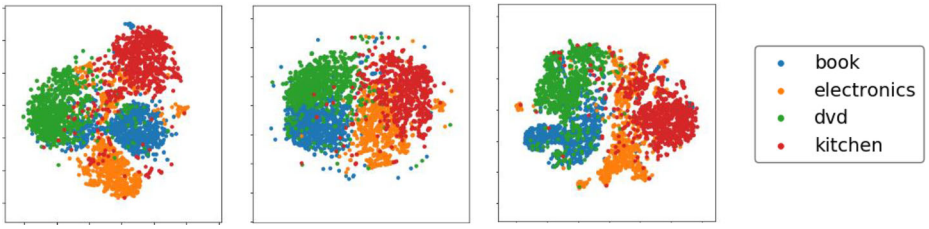


(a) sum, MAP = 38.0 %

(b) DkSA, MAP = 59.6 %

(c) FSA (proposed), MAP = 69.4 %

Fig. 3 Visualization of feature spaces formed by aggregated RoPS features. Each colored dot represents an embedded feature of a 3D shape belonging to one of the ten object categories



(a) sum, MAP = 55.9 % (b) DkSA, MAP = 42.9 % (c) FSA (proposed), MAP = 65.3 %

Fig. 4 Visualization of feature spaces formed by aggregated W2V features. Each colored dot represents an embedded feature of a text document belonging to one of the four topics

Visualization We visualize feature spaces produced by three feature aggregation algorithms, i.e., sum of feature vectors, DkSA, and the proposed FSA. We use t-SNE algorithm [45] for visualization. Figure 3 visualizes feature spaces formed by aggregated RoPS features of test-set 3D shapes contained in the ModelNet10 dataset. Similarly, Fig. 4 compares feature spaces formed by aggregated W2V features of test-set text documents in the Amazon4 dataset. We can observe that, especially in the case of 3D shape shown in Fig. 3, the FSA yields an embedded feature space where categories are better separated than feature spaces produced by the sum aggregation and the DkSA aggregation.

Temporal complexity Table 8 compares temporal complexity of SSC algorithms to compute a distance between two feature sets. Recall that each feature set contains n vectors having d_f dimensions. Chamfer distance (CD) requires high computational cost for distance computation since it does not aggregate features. Sum aggregation is temporally very efficient since it aggregates features by simple summing. VLAD requires $O(m{\cdot}n{\cdot}d_f)$ for aggregation, whose computation is dominated by searching codewords closest to each of n input features. m is number of the codewords for VLAD. FSA requires $O(d_c{\cdot}(n{\cdot}d_f + d_e))$ for aggregation since it needs $O(n{\cdot}d_f{\cdot}d_c)$, $O(n{\cdot}d_c)$, and $O(d_c{\cdot}d_e)$ for sparse coding, average pooling, and sparse embedding, respectively.

For feature aggregation, FSA has the highest complexity in Table 5 since d_c is typically larger than m (e.g., $d_c = 2048$ and $m = 128$). On the other hand, for distance computation, FSA is more efficient than CD and VLAD. Dimensionality of aggregated feature d_e of FSA is significantly smaller than that of VLAD.

Table 9 shows computation time per query on the ModelNet40 test set including 2468 3D models. We used a PC having an *Intel Core i7 6700* CPU, a *GeForce GTX 1080 Ti* GPU, and 64GB DRAM for the timing. In Table 9, the columns named "feature extraction" and "feature aggregation" are processing times for extracting 512 RoPS features from the query 3D model and aggregating these RoPS features of the

Table 8 Temporal complexity for set-to-set comparison

SSC algorithms	feature aggregation	distance computation
CD	–	$O\ (n^2{\cdot}d_f)$
sum	$O\ (n{\cdot}d_f)$	$O\ (d_f)$
VLAD	$O\ (m{\cdot}n{\cdot}d_f)$	$O\ (m{\cdot}d_f)$
FSA	$O\ (d_c{\cdot}(n{\cdot}d_f + d_e))$	$O\ (d_e)$

Table 9 Computation time [s] per query

SSC algorithms	feature extraction	feature aggregation	distance computation	total
CD	0.053	–	3.216	3.269
sum	0.053	0.0006	0.002	0.056
VLAD	0.053	0.0035	0.006	0.063
FSA	0.053	0.0059	0.001	0.060

We used $n = 512$, $d_f = 64$, $m = 128$, $d_c = 2048$, $d_e = 64$

query, respectively. The column "distance computation" means time for comparing the aggregated feature of the query against aggregated features of the 2467 retrieval targets. Note that codes of the SSC algorithms used for the timing were accelerated by different hardware. For example, our implementation of FSA employs the GPU for acceleration while VLAD utilizes multiple cores of the CPU.

Although direct comparison is difficult for the reason described above, Table 9 indicates that the three algorithms that employ feature aggregation based approach, that are, sum, VLAD, and FSA, are more efficient than the algorithm without aggregation, that is, CD. Retrieval time of FSA is comparable to that of sum aggregation, which pools features very quickly. In the feature aggregation based approaches, computation time are dominated by RoPS feature extraction. In the case of CD, the cost of distance computation is dominant, in which element-to-element comparison of all the features between pairs of sets must be performed.

Since the proposed FSA is based on neural network, its training cost is much higher than the existing feature aggregation algorithms. VLAD took about 20 s for codebook learning with $m = 128$ while FSA took about 4 h of training for the ModelNet40 dataset. Note, however, that training of FSA is preprocessing. It is required only once per dataset before querying the dataset.

Table 10 3D shape retrieval accuracy (MAP [%])

algorithms	supervision bylabels?	ModelNet10	ModelNet40
D2 [51]	No	28.6	19.1
SPRH [64]	No	41.2	32.9
LFD [7]	No	49.8	40.9
SPH [52]	No	44.1	33.3
PANORAMA [58]	No	60.3	46.1
SV-DSIFT [11]	No	60.2	49.5
LL-MO1SIFT [11]	No	63.1	48.0
DkSA-POD [13]	No	61.2	48.4
FSA-DSIFT (proposed)	No	**73.3**	**58.8**
FSA-MO1SIFT (proposed)	No	**73.1**	**60.3**
FSA-POD (proposed)	No	**71.6**	**56.6**
3D ShapeNets [69]	Yes	68.3	49.2
MVCNN [61]	Yes	–	79.5
DeepPano [59]	Yes	84.2	76.8
DLAN [14]	Yes	90.6	85.0
PANORAMA-ENN [58]	Yes	93.3	86.3

Table 11 Comparison of set reconstruction loss (MAP [%] for ModelNet10)

set-to-set distance	element-to-element distance		
	Euclidean distance	Manhattan distance	Cosine distance
Earth Mover's distance	61.1	61.3	63.8
Hausdorff distance	52.4	52.4	58.5
chamfer distance	60.2	60.0	**66.6**

4.3 Comparison with existing 3D shape retrieval algorithms

The experiments under the 3D shape retrieval scenario using ModelNet10 and ModelNet40 follow the evaluation protocol provided by Wu et al. [69]. Therefore, we can compare accuracy of our 3D shape matching algorithm using the FSA against accuracies of existing algorithms for 3D shape retrieval.

Table 10 compares MAP scores for the ModelNet10 and the ModelNet40 datasets. As competitors, we use eight algorithms that compare unlabeled 3D shapes without any supervision. Among the eight algorithms, SV-DSIFT [11], LL-MO1SIFT [11], and DkSA-POD [13] are state-of-the-art algorithms for comparing unlabeled 3D shapes. They extract a set of handcrafted feature vectors called DSIFT, MO1SIFT, or POD, respectively. The set of features is aggregated per 3D shape by using feature aggregation algorithms, i.e., SV, LL, or DkSA. We replace these feature aggregation algorithms used by these algorithms with the proposed FSA. In addition, we also list, in Table 10, retrieval accuracies of five deep neural network-based 3D shape matching algorithms that rely on supervised training using object category labels.

In Table 10, those algorithms using FSA outperform the other eight unsupervised algorithms. Feature aggregation using the FSA significantly boosts MAP scores, by about 10% for most cases, compared to such state of the art feature aggregation algorithms as SV, LL, and DkSA. As expected, deep neural networks trained with supervision yield higher retrieval accuracies than the algorithms without supervision. Interestingly, however, the FSA-based algorithms outperform the 3D ShapeNets. This result would partially supports our argument, that is, the FSA learns accurate representation of feature sets without relying on labels.

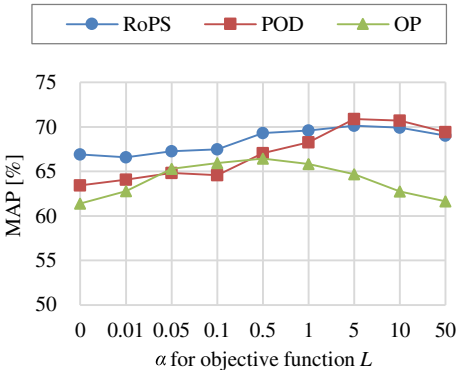


Fig. 5 Retrieval accuracy plotted against blending parameter α of the loss function (ModelNet10)

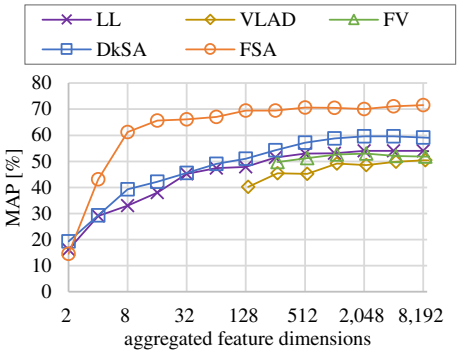


Fig. 6 Retrieval accuracy plotted against aggregated feature dimension (ModelNet10)

4.4 In-depth evaluation of feature set aggregator

This section explores effect design parameter has on the proposed algorithm. We use the ModelNet10 dataset along with the RoPS feature for the experiments described below.

Set reconstruction loss We compare nine set reconstruction losses. We employ three set-to-set distances, i.e., Earth Mover's distance, Hausdorff distance, and chamfer distance for L_R. Besides, for each set-to-set distance, we vary distance metric among set elements. That is, we use Euclidean, Manhattan, and Cosine distances as element-to-element distances. We disable set embedding loss L_E by fixing the hyper-parameter α to 0 in the experiment. The results shown in Table 11 indicates that choice of set reconstruction loss has a significant impact on retrieval accuracy. Chamfer distance combined with Cosine distance, in other words, angular chamfer distance, performs the best among the nine loss functions. Angle would be an appropriate metric to compare the whitened input features having sphered distribution.

Set embedding loss We investigate effectiveness of the set embedding loss L_E by varying the hyper-parameter α from 0 to 50. Figure 5 plots MAP scores against the different values of α. In the figure, $\alpha = 0$ means the FSA is trained solely by set reconstruction loss L_R. Mixing L_R and L_E by using values of $\alpha > 0$, for example, $\alpha = 0.5$, improves retrieval accuracy for all the three 3D shape features we experimented with. The proposed triplet generation algorithm using family of sets has a positive impact on learning better embedding feature space for SSC. In Fig. 5, peaks of retrieval accuracy appear at different values of α depending on the feature

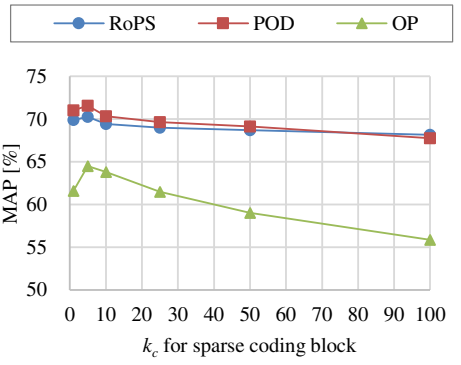


Fig. 7 Retrieval accuracy plotted against sparseness k_c of the sparse coding block (ModelNet10)

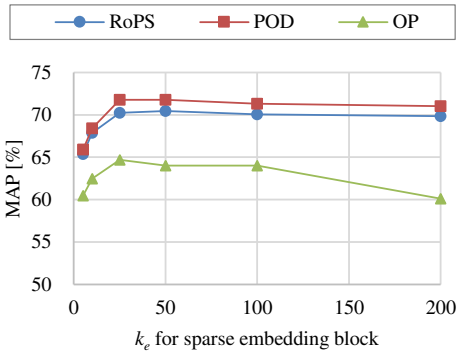


Fig. 8 Retrieval accuracy plotted against sparseness k_e of the sparse embedding block (ModelNet10)

(and also probably on the dataset). This implies that manually searching an appropriate value of α is required to obtain the best result when using the proposed FSA algorithm. Note, however, that the FSA is not very sensitive to the value of α. The FSA consistently produced good MAP scores (over 60%) for the wide range of α we experimented with.

Embedded feature dimension Figure 6 plots MAP scores against the number of dimensions of aggregated features. For the FSA, aggregated feature dimension corresponds to dimensionality d_e of the embedded feature space. Accuracies of the conventional feature aggregation algorithms, i.e., LL and DkSA, drop significantly if dimensionality of features becomes low. In contrast, the FSA keeps high MAP value (nearly 60%) even at dimensionality as small as 8.

Sparseness in encoder Figure 7 and Fig. 8 show an impact of sparsity in the encoder part of the FSA on retrieval accuracy. Figure 7 plots accuracies against the number of non-zero activations k_c of the sparse coding block. We fixed the other hyper-parameters d_c, d_e, and k_e to 2048, 256, and 25, respectively. For all the features in Fig. 7, accuracies have peaks at around $k_c = 5$. Increasing k_c, i.e., having the coding block produce denser activations, lowers retrieval accuracy. Sparseness in the coding block enhances saliency of each encoded input features. Meanwhile, Fig. 8 plots accuracy against k_e, i.e., the number of non-zero activations in the sparse embedding block, while keeping the number of non-zero activations k_c at 5. The plots of accuracy have weak peaks around $k_e = 25$ or $k_e = 50$. Although not so evident as in the case of sparse coding block, sparseness of the embedding block also helps the FSA learn better feature representation of sets.

Ablation study Table 12 demonstrates importance of whitening input features, sparseness in encoder, and objective function for training. In the table, "No" for the "whitening input features" column indicates that input sets of RoPS features are fed into the FSA as-is without

Table 12 Ablation study of FSA (MAP [%] for ModelNet10)

whitening input features	sparseness in encoder	objective function	MAP [%]
Yes	Yes	$L_R + \alpha L_E$	**69.4**
Yes	Yes	L_E only	66.7
Yes	Yes	L_R only	66.9
Yes	No	L_R only	59.7
No	No	L_R only	55.4

whitening. "No" for the "sparseness in encoder" column means sparsification using top-k activations is omitted at the sparse encoding layer and the sparse embedding layer of the FSA. The results validate the design of FSA algorithm that aims at accurate SSC.

5 Conclusion and future work

In indexing, clustering, or retrieval of multimedia data, each datum is often described by an unordered set of high-dimensional feature vectors. To perform these operations effectively and efficiently, Set-to-Set Comparison (SSC) among such unordered sets is very important. Aiming at accurate and efficient SSC, this paper proposed an unsupervised representation learning algorithm called Feature Set Aggregator (FSA). To learn effective embedding of unlabeled sets in an unsupervised manner, we designed two training objectives for FSA, that are, set reconstruction using angular chamfer distance and set embedding using triplets generated on family of sets. Extensive evaluation under three multimedia information retrieval scenarios using 3D shapes, 2D images, and text documents demonstrated efficacy and generality of the FSA.

As future work, we will seek better neural network architecture and better training objective than the current FSA. For example, the current FSA with only a few layers is much shallower than recent neural networks used, for example, for image recognition. We need to evaluate influence of increasing the number of layers of FSA on SSC accuracy. Also, we plan to evaluate the FSA under scenarios other than multimedia information retrieval, e.g., clustering, to further verify the quality and generality of feature representation of sets learned by the FSA. Possible evaluation scenarios include more specific tasks such as fingerprint recognition or palmprint recognition (e.g., [43]). Recognizing fingerprints or palmprints often requires set-to-set comparison where each set consists of keypoints extracted from a skin image. Furthermore, we envision extending the FSA to the problem of cross-modal set-to-set comparison. Learning common representation of sets describing different types of multimedia data would be valuable for cross-modal information retrieval (e.g., [72]) or more challenging cross-modal multimedia understanding problem [44].

References

1. Abadi M et al (2016) TensorFlow: a system for large-scale machine learning. Proc. OSDI 2016:265–283
2. Achlioptas P, Diamanti O, Mitliagkas I, Guibas L (2017) Learning Representations and Generative Models for 3D Point Clouds, arXiv preprint, arXiv:1707.02392
3. Arandjelović R, Gronat P, Torii A, Pajdla T, Sivic J (2018) NetVLAD: CNN architecture for weakly supervised place recognition. TPAMI 40(6):1437–1451
4. Blitzer J, Dredze M, Pereira F (2007) Biographies, Bollywood, boom-boxes and blenders: domain adaptation for sentiment classification. Proc. ACL 2007:440–447
5. Chang AX et al. (2015) ShapeNet: An Information-Rich 3D Model Repository, arXiv:1512.03012
6. Charles RQ, Su H, Kaichun M, Guibas LJ (2017) PointNet: deep learning on point Sets for 3D classification and segmentation. Proc. CVPR 2017:77–85
7. Chen DY, Tian XP, Te Shen Y, Ouhyoung M (2003) On visual similarity based 3D model retrieval. Comput Graph Forum 22(3):223–232
8. Csurka G, Dance CR, Fan L, Willamowski J, Bray C (2004) Visual categorization with bags of Keypoints. Proc. ECCV 2004 workshop on statistical learning in computer vision: 59–74

9. Deng J, Dong W, Socher R, Li L-J, Li K, Li F-F (2009) ImageNet: a large-scale hierarchical image database. Proc CVPR 2009:248–255
10. Fei-Fei L, Fergus R, Perona P (2004) Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. Proc CVPR workshop 2004:59–70
11. Furuya T, Ohbuchi R (2014) Fusing multiple features for shape-based 3D model retrieval, Proc British Machine Vision Conference (BMVC)
12. Furuya T, Ohbuchi R (2015, 2015) Diffusion-on-manifold aggregation of local features for shape-based 3D model retrieval. Proc. ICMR:171–178
13. Furuya T, Ohbuchi R (2016) Accurate aggregation of local features by using K-sparse autoencoder for 3D model retrieval. Proc. ICMR 2016:293–297
14. Furuya T, Ohbuchi R (2016) Deep aggregation of local 3D geometric features for 3D model retrieval. Proc BMVC 2016:121.1–121.12
15. Gavrila DM, Philomin V (1999) Real-time object detection for "smart" vehicles. Proc. ICCV 1999:87–93
16. Geoffrey E, Hinton RRS (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507
17. Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. Proc AISTATS 2011:315–323
18. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2016) Generative adversarial nets. Proc NIPS 2016:2672–2680
19. Günther F English LSA space, https://sites.google.com/site/fritzgntr/home
20. Guo Y, Sohel F, Bennamoun M, Lu M, Wan J (2013) Rotational projection statistics for 3D local surface description and object recognition. IJCV 105(1):63–86
21. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. Proc CVPR 2016:770–778
22. Henrikson J (1999) Completeness and total boundedness of the Hausdorff metric, MIT Undergraduate Journal of Mathematics: 69–80
23. Hoffer E, Ailon N (2015) Deep metric learning using triplet network. Proc. ICLR 2015 workshop
24. Hyvärinen A, Hurri J, Hoyer PO (2009) Natural image statistics: a probabilistic approach to early computational vision. Springer, Verlag
25. Jégou H, Douze M, Schmid C, Pérez P (2010) Aggregating local descriptors into a compact image representation. Proc CVPR 2010:3304–3311
26. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. Proc. ICLR 2015
27. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. Proc. NIPS 2012: 1097–1105.
28. Landauer TK, Foltz PW, Laham D (1998) An introduction to latent semantic analysis. Discourse Process 25(2–3):259–284
29. Lang K (1995) Newsweeder: Learning to filter netnews. Proc. ICML 1995:331–339
30. Hsin-Ying Lee, Jia-Bin Huang, Maneesh Singh, Ming-Hsuan Yang, Unsupervised representation learning by sorting sequences, Proc. ICCV 2017, pp. 667–676, 2017.
31. Lehmann J et al (2015) DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web 6(2):167–195
32. Leng L, Zhang J (2013) Palmhash code vs. palmphasor code. Neurocomputing 108:1–12
33. Leng L, Zhang J, Xu J, Khan MK, Alghathbar K (2010) Dynamic weighted discrimination power analysis in DCT domain for face and palmprint recognition. Proc ICTC 2010:467–471
34. Leng L, Li M, Leng L, Teoh ABJ (2013) Conjugate 2DPalmHash code for secure palm-print-vein verification. Proc CISP 3:1705–1710
35. Leng L, Li M, Kim C, Bi X (2017) Dual-source discrimination power analysis for multi-instance contactless palmprint recognition. MTAP 76(1):333–354
36. Lin R, Xiao J, Fan J (2018) NeXtVLAD: An efficient neural network to aggregate frame-level features for large-scale video classification, Proc. ECCV 2018 workshops: 206–218
37. Lin T-Y, Maji S, Koniusz P (2018) Second-order democratic aggregation. Proc. ECCV 2018:639–656
38. Liu Z, Wang S, Tian Q (2016) Fine-residual VLAD for image retrieval. Neurocomputing 173(3):1183–1191
39. Liu Y, Yan J, Ouyang W (2017) Quality aware network for set to set recognition. Proc. CVPR 2017:4694–4703
40. Lowe DG (2004) Distinctive image features from scale-invariant Keypoints. IJCV 60(2):91–110
41. Lu L, Zhang J, Gao C (2011) Muhammad Khurram khan, Khaled Alghathbar, two-directional two-dimensional random projection and its variations for face and palmprint recognition. Proc ICCSA 2011: 458–458
42. Lu L, Zhang J, Gao C (2011) Muhammad Khurram khan, ping Bai, two dimensional PalmPhasor enhanced by multi-orientation score level fusion. Proc STA 2011:122–129

43. Lu L, Beng A, Teoh J (2015) Alignment-free row-co-occurrence cancelable palmprint. Fuzzy Vault Pattern Recogn 48(7):2290–2303
44. Lu H, Li Y, Chen M, Kim H, Serikawa S (2018) Brain intelligence: go beyond artificial intelligence. Mobile Netw Appl 23(2):368–375
45. van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579–2605
46. Makhzani A, Frey B (2014) k-sparse autoencoders, Proc. ICLR 2014
47. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. Proc. NIPS 2013:3111–3119
48. Nilsback M-E, Zisserman A (2008) Automated flower classification over a large number of classes. Proc ICVGIP 2008:722–729
49. Noroozi M, Favaro P (2016) Unsupervised learning of visual representations by solving jigsaw puzzles. Proc. ECCV 2016:69–84
50. Ohbuchi R, Minamitani T, Takei T (2005) Shape-similarity search of 3D models by using enhanced shape functions. IJCAT 23(2):70–85
51. Osada R, Funkhouser T, Chazelle B, Dobkin D (2002) Shape distributions. ACM Trans Graph (TOG) 21(4): 807–832
52. Papadakis P, Pratikakis I, Perantonis S, Theoharis T (2007) Efficient 3D shape matching and retrieval using a concrete radialized spherical projection representation. Pattern Recogn 40(9):2437–2452
53. Pathak D, Krähenbühl P, Donahue J, Darrell T, Efros AA (2016) Context encoders: feature learning by Inpainting. Proc CVPR 2016:2536–2544
54. Perronnin F, Sánchez J, Mensink T (2010) Improving the fisher kernel for large-scale image classification. Proc. ECCV 2010, part IV: 143–156
55. Qi CR, Yi L, Su H, Guibas LJ (2017) PointNet++: deep hierarchical feature learning on point Sets in a metric space, Proc. NIPS 2017: 5105–5114
56. Rubner Y, Tomasi C, Guibas LJ (1998) A metric for distributions with applications to image databases. Proc ICCV 1998:59–66
57. Schroff F, Kalenichenko D, Philbin J (2015) Facenet: a unified embedding for face recognition and clustering. Proc CVPR 2015:815–823
58. Sfikas K (2018) Ioannis Pratikakis, Theoharis Theoharisa, ensemble of PANORAMA-based convolutional neural networks for 3D model classification and retrieval. Comput Graph 71:208–218
59. Shi B, Bai S, Zhou Z, Bai X (2015) DeepPano: deep panoramic representation for 3-D shape recognition. Signal Process Lett 22(12):2339–2343
60. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. Proc. ICLR 2015:1–14
61. Su H, Maji S, Kalogerakis E, Learned-Miller E (2015) Multi-view convolutional neural networks for 3D shape recognition. Proc. ICCV
62. Thorsten Joachims A (1997) Probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. Proc ICML 1997:143–151
63. Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol P-A (2010) Stacked Denoising autoencoders: learning useful representations in a deep network with a local Denoising criterion. J Mach Learn Res 11: 3371–3408
64. Wahl E, Hillenbrand U, Hirzinger G (2003) Surflet-pair-relation histograms: a statistical 3D-shape representation for rapid classification. Proc Fourth Int Conf 3D Digit Imag Model (3DIM) 2003:474–481
65. Wang X, Gupta A (2015) Unsupervised learning of visual representations using videos. Proc. ICCV 2015: 2794–2802
66. Wang J, Yang J, Yu K, Lv F, Huang T, Gong Y (2010) Locality-constrained linear coding for image classification. Proc. CVPR 2010:3360–3367
67. Wei X, Zhang Y, Gong Y, Zhang J, Zheng N (2018) Grassmann pooling as compact homogeneous bilinear pooling for fine-grained visual classification. Proc ECCV 2018:365–380
68. Word2vec. https://code.google.com/archive/p/word2vec
69. Wu Z, Song S, Khosla A, Yu F, Zhang L, Tang X, Xiao J (2015) 3D ShapeNets: a deep representation for volumetric shapes. Proc. CVPR 2015:1912–1920
70. Xi Z, Kai Y, Zhang T, Huang TS (2010) Image classification using super-vector coding of local image descriptors. Proc ECCV 2010:141–154
71. Xian Y, Lampert CH, Schiele B, Akata Z (2018) Zero-shot learning - a comprehensive evaluation of the good, the bad and the ugly. TPAMI 40(8)
72. Xu X, Song J, Lu H, Yang Y, Shen F, Huang Z (2018) Modal-adversarial semantic learning network for extendable cross-modal retrieval. Proc ICMR 2018:46–54
73. Yang Y, Feng C, Shen Y, Tian D (2017) FoldingNet: Interpretable Unsupervised Learning on 3D Point Clouds, arXiv preprint, arXiv:1712.07262

74. Zaheer M, Kottur S, Ravanbakhsh S, Poczos B, Salakhutdinov RR, Smola AJ (2017) Deep sets, Proc. NIPS 2017: 3394–3404.

Takahiko Furuya received his B. Eng., M. Eng., and Ph. D. in engineering degrees from University of Yamanashi in 2008, 2010, and 2015, respectively. He worked for Nisca corporation from 2010 to 2013. He was granted JSPS Research Fellowships for Young Scientists (DC2) in 2015. In 2015, he joined, as an assistant professor, the Department of Computer Science, University of Yamanashi. His research interests include multimedia information retrieval and machine learning.

Ryutarou Ohbuchi received the B. Eng from the Department of Electrical and Electronic Engineering, Sophia University, Tokyo, Japan. He received M. Eng. from the Department of Computer Science, University of Electro-communications, Tokyo, Japan. He joined IBM Japan Science Institute in 1984. He received his Ph. D in Computer Science from the University of North Carolina at Chapel Hill in 1994. From 1994 to 1999, he worked as a researcher at IBM Research, Tokyo. In 1999, he joined, as an associate professor, the Department of Computer Science, University of Yamanashi, Yamanashi. Japan. He is a full professor since 2007. In the past, he has worked on graphics display hardware design, augmented reality display for medical applications, photorealistic rendering, digital watermarking of 3D shape models, and others. His recent research interests include multimedia data analysis, retrieval, and mining, especially of 3D shape models. Prof. Ohbuchi is a member of ACM, IEEE Computer Society, IPSJ, and IIEEJ.