VSE-fs: Fast Full-Sample Visual Semantic Embedding

Songlin Zhai, Guibing Guo*, Fajie Yuan, Yuan Liu, and Xingwei Wang

Abstract—The visual semantic embedding (VSE) aims to construct a joint embedding space between visual features and semantic information, whereby classes can be well retrieved for a given image. However, VSE faces the computational challenge due to the large scale image-class data and the constrained system processing power. To speed up model training, many researchers resort to different sampling strategies by involving only a small portion of the classes at each training step. However, these methods are greatly biased especially when the sampling distribution deviates from the true data distribution. In order to retain VSE models fidelity, we adopt the regular *full-sample* in our algorithm. We also devise two *separate* optimization strategies to reduce time complexity, and derive more effective updating rules. The experimental results on four real datasets demonstrate that our approach not only converges much faster than the state-of-the-art sampling models, but also generates more accurate class retrieval.

Index Terms—Visual Semantic Embedding, Full Sample, Negative Sampling

1 Introduction

Class retrieval for a given image becomes a research hot spot to address the problem of automatically annotating images with relevant classes, which are often used as keywords by search engines. However, there is a *semantic gap* between the low-level image pixels (features) and related high-level class content (semantics). Many research works have attempted to solve the problem, such as Visual Semantic Embedding (VSE) models [1], [2] to build a joint embedding space between images and classes, and Neural Network models (NN) [3] to automatically learn image features and then classify images into multiple classes. In practice, NN models have a large number of network parameters, and thus are usually computationally expensive to train. Training a complicated model may take up to several weeks even if being equipped with many power machines (e.g., expensive GPUs). Besides, the information of image raw pixels may not be available in some situations or datasets, causing the class retrieval task more challenging. Whereas VSE models exactly aim to explore the latent relations between images and classes, bypassing the weaknesses of traditional NNs. Furthermore, VSE model is also be easily extended by appending the image features (see Section 3.4). Thus, our work follows the line of VSE models, and try to build a better joint embedding space with full sample without the consideration of image pixels or textual information.

However, VSE models face the computational challenge due to the large scale image-class data and the constrained system processing power, which limit its application capacity. In detail, the bottleneck of training VSE models lies in two parts: (1) the scoring inner-product operation and (2) the huge number of images and classes. In order to reduce time complexity, some previous research works suggest to sample from class space, which later has become a common practice by creating a sample of $m < M$ classes at each training step. Various sampling methods have been proposed to alleviate the performance defect caused by the imperfect sampler, including uniform sampling [4], static sampling (word2vec) [5], dynamic sampling (WARP) [2], and a state-of-the-art fast dynamic & adaptive sampling (VSE-ens) [1]. However, sampling-based methods are biased [6], i.e., it cannot converge to the same loss as full examples - regardless how many update steps are taken. Generally, there are two ways to mitigate this issue: (1) design a complicated sampling distribution closer to the true data distribution - which is either suboptimal or inefficient, or (2) increase the sampling size, m – which is trivial but costly. Note that there is little research work to study a more efficient method to compute the relevance scores.

In this paper, we focus on the full-sample based learning to retain VSE model's fidelity which resolves the performance defect of sampling-based methods, and meanwhile adopt a least square loss function to preserve a determinate analytic solution. To reduce the huge time complexity, we propose two efficient transformation strategies to accommodate the whole data loss. Specifically, we propose a f-separate transformation to reduce the time complexity of computing score when update parameters. Furthermore, we also conduct a positive separate (p-separate) transformation to resolve the huge time complexity of full sample learning. The experimental results on four real-world datasets, namely OpenImages[1] [7], NUS-WIDE[2] [8], IAPR-TC12[3] [9] and Flickr[4] demonstrate that our VSE-fs model trains 4.27 times faster than the start-of-art model (VSE-ens) on OpenImages, 12.59 times on NUS-WIDE, 9.68 times on IAPR-

* *Corresponding author*

- *Songlin Zhai, Guibing Guo, Yuan Liu and Xingwei Wang are with the Software College, Northeastern University, China.*
- *Fajie Yuan is with Tencent computer system Co. Ltd, Shenzhen City, Guangdong Province, China.*

1. https://github.com/openimages/dataset
2. lms.comp.nus.edu.sg/research/NUS-WIDE.htm
3. http://www.imageclef.org/photodata
4. https://www.flickr.com/

TC12 and 19.19 times on Flickr, and produces significant improvements on class retrieval accuracy in the meanwhile.

2 Related Work

In this section we briefly review two negative sampling strategies as well as three representative algorithms, which will be used for performance comparison in Section 4.3.

2.1 Uniform Sampling

Pairwise learning to rank with a uniform sampling, also referred to as Opt-AUC [4] criterion, is a popular way to optimize VSE models. The aim of Opt-AUC is to rank any positive $(image, class^+)$ pair from ground-truth higher than a randomly sampled negative $(image, class^-)$ one:

$$score(image, class^+) > score(image, class^-)$$

where $class^+$ and $class^-$ denote the positive and negative class, respectively, also applied to the following discussion.

Although the uniform sampling of negative classes is very efficient and only takes $O(1)$ time, it converges much slower and performs worse than other well-designed samplers. This is because many sampled examples are not informative and thus less effective for model training as explained by [1].

2.2 Dynamic Sampling

On account of the accuracy defect of uniform sampler, some dynamic samplers are proposed to better optimize VSE models. Two state-of-the-art algorithms are shortly reviewed as follows.

WARP [2]: (Weston et al. 2011) introduce a Weighted Approximate-Rank Pairwise Loss, where the weighted rank is implemented by a rejection sampler. That is, it will repeatedly draw negative classes until the score of a drawn negative class meets the requirement:

$$score(image, class^-) \geqslant score(image, class^+).$$

However, WARP sampling is expensive to find the violated negative examples due to the time-consuming traveling operation on the whole non-positive class set which takes $O(TK)$ time (T and K are the average sampling trials and the scoring computation steps respectively). What's worse, T will become much larger after several training iterations, as most positive pairs are likely to have higher scores than negative ones.

VSE-ens [1]: To solve the aforementioned issue of WARP sampling, *VSE-ens* [1] is proposed to approximately and efficiently estimate the rank of negative classes without executing the scoring inner-product operation before each stochastic gradient descent (SGD) update. That is, VSE-ens aims to sample the item pairs that are most likely (maybe not exactly) to meet the rejection requirement of WARP sampling.

In summary, uniform sampler i.e. Opt-AUC [4] trains VSE models with low time complexity but only reaches poor performance. Dynamic sampler i.e. WARP [2] is capable of solving the issue of poor performance (in uniform sampler) but at the cost of high time complexity. Although the improved dynamic sampler i.e. VSE-ens [1] can help relieve the efficiency problem of WARP, sampling-based approaches inherently cannot take full advantage of all information in the dataset, leading to performance loss to some extent. Thus, in this paper we will introduce our VSE model (VSE-fs) with efficient full-sample training in the following sections.

3 Scalable VSE Model

For the sake of discussion, we will first introduce a number of notations in this paper. Given a set of image-class pairs represented by $\mathcal{D} = \{(i, c)\}_{N \times M}$, where i and c denote an image and a corresponding class, respectively; N and M are the number of images and classes, respectively. A VSE model is to map images and classes into image embedding space denoted by $\mathbb{R}_I$ and class embedding space denoted by $\mathbb{R}_C$, respectively. For an image i, it can be denoted by an embedding vector $\mathbf{v}_i \in \mathbb{R}_I$, and similarly a class c can be represented by $\mathbf{v}_c \in \mathbb{R}_C$. Hence, we use ${V_I}^{N \times K} \subseteq \mathbb{R}_I$ to denote the image embedding matrix and take ${V_C}^{M \times K^{'}} \subseteq \mathbb{R}_C$ for the class embedding matrix, where K and $K^{'}$ are the dimension of image and class embedding matrix respectively. Moreover, we set $K = K^{'}$ to force the mapping of images and classes into a joint embedding space, whereby classes can be quickly retrieved for a given image. Lastly, we use C_i as the set of classes associated with image i, and I_c as the set of images labeled with class c, where C denotes the ground-truth class set of all images and I denotes the set of all images. Thus, the objective of VSE is to learn proper embedding representations (V_I and V_C) of images and classes.

3.1 VSE-fs Model

Different from previous methods to sample from non-positive classes, we propose a full-sample based model to retain the fidelity of VSE models. The problem is that the number of images for each class is extremely imbalanced in full-sample learning. To resolve this issue, we propose a weighting scheme (in Section 3.1.1) to take into account the importance of different classes. Furthermore, to represent images more accurately, we model an image (e.g. i) by the combination of image embedding vector $\mathbf{v}_i$ and the summation of related positive class embedding vectors $\sum_{c \in C_i} \mathbf{v}_c$. In summary, we derive a weighted least square loss function to discriminate positive classes from non-positive ones. To be specific, it is a summation of weighted residual error[5], defined as follows:

$$\underbrace{\min_{\forall\, i:N} loss}_{} = \underbrace{\sum_{c \in C_i} w_{ic} (\xi_{ic}^{+} - \zeta_{ic})^2}_{\text{positive part}} + \underbrace{\sum_{c \notin C_i} \beta_c \zeta_{ic}^2}_{\text{non-positive part}} + reg \quad (1)$$

where ξ_{ic}^{+} and ξ_{ic}^{-} denote the ground truth score of positive and non-positive (i, c) pair respectively, where $\xi_{ic}^{-} = 0$ as the image is not related with the non-positive class. ζ_{ic} is the relevant score of (i, c) pairs predicted by our VSE-fs model which will be defined in Section 3.2. $w_{ic} \in W^{N \times M}$ is the positive weight for positive (i, c) pairs, accounting

5. Our optimization target pertains to unconstrained *Quadratic Program* (QP), and is a least square approximation which has the determinate analytic solution.

TABLE 1: The mean number of positive and non-positive classes of images across four datasets.

Feature	OpenImages	NUS-WIDE	IAPR-TC12	Flickr
Positive	8	8	4	5
Non-Positive	7,475	5,010	271	595
Proportion[♭]	1 / 934	1 / 626	1 / 68	1 / 119

[♭] The number of positive ones VS the number of non-positives ones.

for the co-occurrence frequency of image i and class c. β_c controls the contribution of non-positive classes to the loss function. It is a class-specific weighting strategy elaborated in Section 3.1.1. $reg = \lambda\left(\sum_{i=1}^{N}||\mathbf{v}_i||^2 + \sum_{c=1}^{M}||\mathbf{v}_c||^2\right)$ is the regularization term to avoid overfitting.

3.1.1 Weighting Scheme of Non-positive Classes

In four datasets, the number of images for each class is extremely imbalanced, that is, some classes are '*common*' or '*popular*' while the others are '*infrequent*' or '*unpopular*'. It is probable that a 'popular' class is a true-negative one for an image, if the image is not related with the class. Thus, we should assign a higher weight for these *true negative* classes and lower weight for those that may be positive. Inspired by the weighting strategies of [10], we propose an adaptive weighting strategy[6], given by:

$$\beta_c = \beta_0 \frac{\chi_c^{\alpha}}{\sum_{c'=1}^{M}\chi_{c'}^{\alpha}}$$

where χ_c denotes the '*popularity*' of this class across the whole dataset and is defined by $|I_c|/\sum_{c'=1}^{M}|I_{c'}|$; β_0 controls the overall strength of the weights. α is a nonlinear scale factor to control the importance of non-positive weights relative to the positive weights.

3.2 Scoring Strategy

The positive classes of a given image are valuable in modelling this image by which we can get more accurate results (experimental results are given in Section 4.5). Different from previous research works [1], [2], we model an image (e.g., i) by the combination of image embedding vector $\mathbf{v}_i$ and the summation embedding vectors of its associated positive classes $\sum_{c\in C_i}\mathbf{v}_c$ to better capture the connections between images and classes, defined by:

$$\zeta_{ic} = \langle \mathbf{v}_i + \gamma\mathbf{p}_i, \mathbf{v}_c\rangle \; - \; excl_c \tag{2}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product operation between two vectors, and $\gamma \in [0,1]$ controls the strength of the injected class embedding vectors. We denote $\mathbf{p}_i$ as the extended embedding vector and define it as follows:

$$\mathbf{p}_i = |C_i|^{-\frac{1}{2}} \sum_{c'\in C_i}\mathbf{v}_{c'}$$

where $\mathbf{p}_i \in P^{N\times K}$ is the summation of all positive class embedding vectors for a given image i, and thus P is the extended embedding matrix with the size of $N \times K$ and $|C_i|^{-\frac{1}{2}}$ is the normalization term.

6. It is trivial to prove that χ^{α} is a monotone decreasing function with respect to the scale factor α by taking the first and second derivation. Thus, the scale factor is indeed an amplification coefficient.

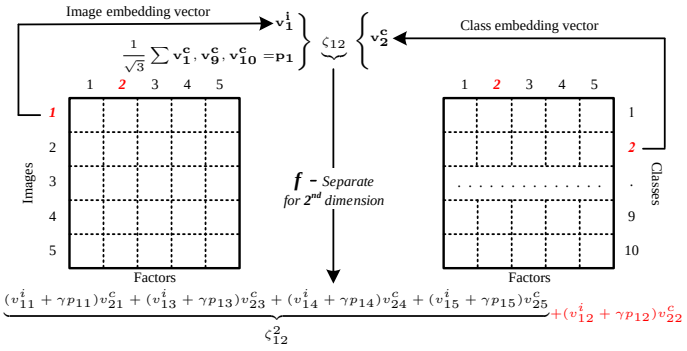

Fig. 1: f-separate Transformation where we use $v_{..}^{i}$ and $v_{..}^{c}$ to denote the entries in image and class embedding matrix respectively. When updating the 2-dimension factor of the image or class embedding vector, the score of training pair $(1,2)$ can be converted into a current dimension-invariant value ζ_{12}^2 by subtracting the element product $(v_{12}^{i}+\gamma p_{12})v_{22}^{c}$, which can be pre-computed in $O(1)$ time.

Additionally, $excl_c$ in Eq. 2 is used to exclude the class embedding vector if $\mathbf{p}_i$ contains the class in question. Formally, it can be defined as a piece-wised function as follows:

$$excl_c = \begin{cases} \gamma\,|C_i|^{-\frac{1}{2}}\langle \mathbf{v}_c^{'}, \mathbf{v}_c\rangle, & c \in C_i \\ 0, & c \notin C_i \end{cases}$$

where $\mathbf{v}_c^{'}$ is the excluded class embedding vector from $\mathbf{p}_i$ which denotes the cached embedding vector of class c after last iteration and will not be updated in this iteration.

Given that our loss function is a quadratic convex function (see Footnotes 5 and Eq. 1), we can derive the optimization rules by directly setting the derivation of our loss function to 0 with respect to v_{if}, where f denotes the f^{th} dimension in an either image or class embedding vector. Finally, the obtained image updating rule is given as follows:

$$v_{if} = \frac{\sum_{c\in C_i} w_{ic}v_{cf}(\xi_{ic}^{+} - \zeta_{ic}) - \sum_{c\notin C_i}\beta_c v_{cf}\zeta_{ic}}{\lambda} \tag{3}$$

3.3 Scalable Transformation

Eq. 3 summarizes the updating rules in element level which takes $O(K^2NM)$ time of updating the whole image embedding matrix. Specifically, the huge amount of time is cost by the scoring operation and the summation in non-positive part. In this section, we will propose two *separate* transformations to greatly alleviate the training time complexity.

3.3.1 f-separate Transformation for Scoring Operation

Our first observation regarding Eq. 3 is that one has to perform the inner product operation (i.e., ζ_{ic}) for each parameter optimization (i.e., v_{if}), which merely takes $O(K)$ time. Although the scoring operation does not sound a very time-consuming process in updating one element of an embedding vector, it will quickly reach up to $O(K^2)$ time in order to update the whole embedding vector, letting alone the operation working on all the non-positive (i,c) pairs. Thus, it is necessary to speed up the process. In fact, the reason why we re-compute the score in updating another dimension is that the element of current dimension is a part of the scoring operation, and the score will change along with the updating dimension.

Hence, the basic idea is to separate the current updating dimension i.e. f from the score and cache the summation of other f-irrelevant (dubbed f-separate) dimensions before updating f dimension. Later, we can update the cache before updating another dimension i.e. $f+1$. In this way, $O(K^2)$ time can be simply reduced to $O(K)$. Hence, it is easy to obtain the f-separate scoring function from Eq.2:

$$\zeta_{ic}^{f} = \zeta_{ic} - v_{cf}(v_{if} + \gamma p_{if}) \ + \ excl_c \tag{4}$$

where ζ_{ic}^{f} denotes the computation without f-dimension.

Thus, by pre-computing ζ_{ic}^{f}, we can compute ζ_{ic} in $O(1)$ time to accelerate the training process. The time complexity can be reduced from $O(K^2NM)$ to $O(KNM)$ for updating image or class embedding matrix. We then apply the f-separate strategy to Eq. 3, deriving the following new updating rules:

$$v_{if} = \frac{\sum_{c\in C_i} w_{ic}v_{cf}\triangle score_{ic} - \underline{\sum_{c\notin C_i} \beta_c v_{cf}(\zeta_{ic}^{f} + \gamma p_{if}v_{cf})}}{\sum_{c\in C_i} w_{ic}v_{cf}^2 + \underline{\sum_{c\notin C_i} \beta_c v_{cf}^2} + \lambda} \tag{5}$$

where $\triangle score_{ic} = \xi_{ic}^{+} - \zeta_{ic}^{f} - excl_c - \gamma p_{if}v_{cf}$ for the sake of simplicity.

Fig. 1 explains the idea of f-separate transformation. Specifically, assume that we have 5 images and 10 classes in the training dataset and set the embedding dimension as 5. Suppose image 1 is related with classes { 1, 9, 10 }, the score ζ_{12} can be transformed into a 2-separate value ζ_{12}^{2} by subtracting $(v_{12}^{i} + \sum_{c'\in\{1,9,10\}} v_{c'2}^{c})v_{22}^{c}$, which can be computed before updating the 2^{nd} element in the image embedding vector of image 1.

3.3.2 p-separate Transformation for Non-positive Part

The time complexity of Eq. 5 is smaller than the original updating rules. However, the underline parts still need to directly traverse the whole non-positive class set. Moreover, as shown in Table 1, the number of non-positive classes is very large and much larger than the number of positive ones. In other words, the computational bottleneck now lies in the summation of the non-positive classes, which almost requires to traverse the whole class set.

To ameliorate the time complexity of updating rules, the traverse learning over all non-positive classes can be transformed as the residuals between all (i, c) pairs and the whole positive (i, c^+) pairs (relatively small number of item pairs), referred to as p-separate transformation.

Firstly, we focus on $\sum_{c\notin C_i} \beta_c v_{cf}^2$, which can be reformulated as:

$$\sum_{c\notin C_i} \beta_c v_{cf}^2 = \underbrace{\sum_{c=1}^{M} \beta_c v_{cf}^2}_{\text{image independent}} - \sum_{c\in C_i} \beta_c v_{cf}^2$$

If we define $H^C = \beta V_C^\top V_C$, where $\beta = [\beta_1, ..., \beta_c]$ is the weight vector of classes, the first term $\sum_{c=1}^{M} \beta_c v_{cf}^2$ can be noted as $h_{ff}^{c} \in H^c$, which is independent of the current updating image embedding vector. Thus, we can pre-compute this term before updating elements of the image embedding vector. The time-consuming operation of summation in negative part can be transformed into a pre-computed time and the summation in positive part time.

Additionally, we can further speed up the caching process by only computing only half of entries in the cached matrix since H^C is a upper triangular matrix.

Algorithm 1: Fast Full-sample VSE-fs Model

Input: $\mathcal{D}$, K, λ, positive and non-positive weights W, β_0 and scale factor α
Output: Image and class embedding matrix V_I, V_C

1 **for** $\forall i \in N, c \in C_i$ **do**
2 | *Compute* $\zeta_{ic} \leftarrow$ Eq. 2 ▶ $O(K|\mathcal{D}|)$
3 **end**
4 **while** *Not Converged* **do**
5 | // Update the image embedding matrix ;
6 | Update cache $H^C = \beta V_C^\top V_C$ ▶ $O(K^2M)$;
7 | **for** $i \in 1 \rightarrow N$ **do**
8 | | **for** $k \in 1 \rightarrow K$ **do**
9 | | | Compute ζ_{ic}^{f} in $C_i \leftarrow$ Eq. 4;
10 | | | Update $v_{if} \leftarrow Eq.\ 6$ ▶ $O(K + |C_i|)$;
11 | | | Update score ζ_{ic} in C_i ;
12 | | **end**
13 | **end**
14 | // Update the class embedding matrix ;
15 | Update cache $H^I = (\gamma P + V_I)^\top(\gamma P + V_I)$ ▶ $O(K^2N)$;
16 | **for** $c \in 1 \rightarrow M$ **do**
17 | | **for** $k \in 1 \rightarrow K$ **do**
18 | | | Compute ζ_{ic}^{f} in $I_c \leftarrow$ Eq. 4 ;
19 | | | Update v_{cf} ▶ $O(K + |I_c|)$;
20 | | | Update score ζ_{ic} in I_c ;
21 | | **end**
22 | **end**
23 **end**
24 **return** V_I and V_C

Analogously, we can also derive pre-computed formulation of ζ_{ic}^{f} $mixed$ terms by applying the similar strategy as follows:

$$\sum_{c\notin C_i} \beta_c v_{cf}(\zeta_{ic}^{f} + \gamma v_{cf}p_{if}) = \sum_{k\neq f}(v_{ik} + \gamma p_{ik})h_{fk}^{c} + \gamma p_{if}h_{ff}^{c} - \sum_{c\in C_i} \beta_c v_{cf}(\zeta_{ic}^{f} + \gamma v_{cf}p_{if})$$

Therefore, after applying p-separate transformation strategy, the final updating rules of image embedding matrix in element level can be re-written as follows:

$$v_{if} = \frac{\sum_{c\in C_i}\{w_{ic}v_{cf}(\xi_{ic}^{+} - excl_c) - \triangle\omega_{ic}(\gamma v_{cf}^2 p_{if} + v_{cf}\zeta_{ic}^{f})\} - term_i}{\sum_{c\in C_i} \triangle\omega_{ic}v_{cf}^2 + h_{ff}^{c} + \lambda} \tag{6}$$

where $\triangle\omega_{ic} = w_{ic} - \beta_c$, $term_i = \sum_{k\neq f} v_{ik}^{'} h_{fk}^{c} - p_{if}h_{ff}^{c}$ for the sake of simplicity. Analogously, we can also derive the updating rule of the entry in class embedding matrix by applying these two separate strategies.

Algorithm 1 summarizes the accelerated full-sample VSE model (VSE-fs). According to the updating rules, calculating each ζ_{ic}^{f} requires the score of the corresponding training pair (i, c), i.e., ζ_{ic}. Thus, Lines (1-3) compute the initial score in advance, which is used in the subsequent steps. Note that although ζ_{ic} changes when updating v_{if} or v_{cf}, it can be updated synchronously by Line 11 or Line 20.

TABLE 2: Time complexity of all comparison models in each training iteration, where T in WARP denotes the average sampling trials for each SGD update. c_1 (in VSE-ens) and c_2 (in Opt-AUC) are the constant coefficient of the time complexity respectively with the relationship of $c_2 < c_1 \ll T$.

VSE-fs	VSE-ens	WARP	Opt-AUC
$O((N+M)K^2 + K\|\mathcal{D}\|)$	$O(c_1 K\|\mathcal{D}\|)$	$O(TK\|\mathcal{D}\|)$	$O(c_2 K\|\mathcal{D}\|)$

3.4 Discussion

In Algorithm 1, updating an image embedding vector takes $O(K + |C_i|)$ time (Line 10). Thus, one VSE-fs iteration takes $O(K^2N + K|\mathcal{D}|)$ ($|\mathcal{D}|$ is the number of positive (i, c) pairs) time for updating all image-related parameters (Line 7). The overall time complexity for updating parameters of both images and classes is $O(K^2(N + M) + K|\mathcal{D}|)$. For the VSE-ens model, it updates the class rank for every $M \log(M)$ steps which takes $O(const \cdot K|\mathcal{D}|)$ for every iteration. Table 2 summarizes the time complexity for all aforementioned models, where c_1 and c_2 are the constant coefficient of VSE-ens and Opt-AUC; T denotes the average sampling trials of WARP. Besides, there is a relation with $T \gg c_1 > c_2$. Note that $(N + M)K^2$ has the same order of magnitude as $K|\mathcal{D}|$ because $(N + M)K$ is not always larger than $|\mathcal{D}|$; that is, for each iteration, our model has almost the same order of time complexity as Opt-AUC and VSE-ens. Due to a smaller constant coefficient, Opt-AUC model has a much shorter training time compared with VSE-ens. Most importantly, our VSE-fs model converges much faster because it takes into account full samples rather than a training pair when updating each parameter.

Additionally, we can also safely parallel the algorithm by separating the updates of different images (or classes) — allowing different workers to update the parameters with disjoint sets of images (or classes) since the updating process is independent with each other. Note that the updating process of different dimensions (i.e. v_{if} and v_{if+1}) cannot be paralleled, since the entry of current updating dimension (i.e. f) will have an influence on the updating process of other dimensions (i.e. $f + 1$) by ζ_{ic}^{f+1} (see Eq. 6).

Although in this paper we aim to resolve the task of visual semantic embedding without the content of images, our proposed VSE-fs model can be easily applied to the situations where those information is available. A straightforward manner is to reformulate the original scoring as:

$$\zeta_{ic} = \underbrace{\langle \mathbf{v}_i + \gamma \mathbf{p}_i, \mathbf{v}_c \rangle - excl_c}_{\text{VSE part}} + \underbrace{\langle \mathbf{E}\mathbf{x}_i, \mathbf{v}_c \rangle}_{\text{Visual part}}$$

where $\mathbf{E}$ and $\mathbf{x}_i$ denote the transformation matrix and image visual features extracted by advanced CNNs [3], respectively. Besides, all of VSE models can be extended by using this method, which also demonstrates the scalability of VSE research field. However, it is beyond the discussion of this paper, and we will leave it for future exploration.

4 Experiments

4.1 Experimental Settings

4.1.1 Datasets

In our experiments, we adopt three popular image class datasets, namely OpenImages [7], NUS-WIDE [8] and IAPR-TC12 [9]. All of them can be accessed publicly (see Footnotes 1∼3). Additionally, we crawl another image datasets from Flickr using Flickr API. The original NUS-WIDE, IAPR-TC12 and clawed Flickr datasets are used for both training and testing. For OpenImages, we randomly sample a large volume (around 1.5M) of (image, class) pairs from the original OpenImages dataset in our experiments to reduce the whole training time, since the volume of images in original dataset has reached an astonishing scale (about 9M). The resulting dataset contains 1,388,832 images and 7,483 classes. For each image, we preserve one (image, class) pair for testing and the rest for training. In this way, we obtain 10,052,007 (image, class) pairs in the training set and the rest 1,388,832 in the test set.

TABLE 3: Training time comparison on the three datasets, in which the row of '*Accelerate*' denotes the improvements of our approach relative to the sampling model in question, and the time indicates the elapsed time of each training iteration.

Model	OpenImages	NUS-WIDE	Flickr	IAPR-TC12
VSE-fs	6.14 m	32.93 s	10.31s	1.73 s
VSE-ens	32.33 m	447.49 s	208.11 s	18.47 s
Accelerate	4.27 x	12.59 x	19.19 x	9.68 x
WARP	216.39 m	1047.71 s	1010.56 s	34.33 s
Accelerate	32.24 x	30.82 x	97.02 x	18.84 x
Opt-AUC	9.91 m	85.86 s	75.88 s	8.10 s
Accelerate	0.61 x	1.61 x	6.36 x	3.68 x

4.1.2 Parameter Settings

We fix the number of latent factors as $K = 128$, and initialize variables by a normal distribution $\mathcal{N}(0, 0.001)$. The other parameters suggested by the original papers are adopted in our experiments. Additionally, our method works without learning rate, bypassing the well-known difficulty in tuning stochastic gradient descent learners. Five popular ranking metrics are adopted to measure the class retrieval accuracy, including precision and recall (denoted by Pre@N, Rec@N), mean average precision (MAP), normalize discounted cumulative gain (NDCG) and area under the ROC curse (AUC), where the cutoff N is set to 5 or 10. The detailed metric definitions can be found in [1], which are omitted in this paper due to space limitations.

4.2 Training Speed

Although our model takes advantage of full sample, we stress that our model can be efficiently executed, and much faster than the models with negative sampling. Besides, the cached terms can be further optimized by the symmetry of cache matrices. Moreover, our model can be efficiently paralleled to train (different workers update different image or class embedding vectors) which will dramatically reduce the training time. Due to the indeterminacy of negative sampling results, baseline models must be performed in single thread. Thus, we just run a single thread to train VSE-fs model for fair efficiency comparison. The per iteration training time for these models is presented in Table 3, where m and s denote minutes and seconds respectively. Additionally, due to the time-consuming sampling operation

TABLE 4: The ranking accuracy of all methods, where values '(+)' indicate the percentage of improvements (symbol % is omitted) our approach achieves relative to the corresponding baseline model.

Model	Pre@5	Rec@5	Pre@10	Rec@10	MAP	NDCG	AUC
OpenImages							
VSE-fs	0.0869	0.4303	0.0558	0.5579	0.2846	0.3492	0.7788
VSE-ens	0.0837 (+3.82)	0.4183 (+2.87)	0.0557 (+0.18)	0.5572 (+0.13)	0.2744 (+3.72)	0.3410 (+2.40)	0.7783 (+0.06 x)
WARP	0.0706 (+23.09)	0.3529 (+21.93)	0.0474 (+17.72)	0.4741 (+17.68)	0.2315 (+22.94)	0.2886 (+21.00)	0.7369 (+5.69 x)
Opt-AUC	0.0471 (+84.50)	0.2355 (+82.72)	0.0346 (+61.27)	0.3465 (+61.01)	0.1461 (+94.80)	0.1928 (+81.12)	0.6730 (+15.72 x)
NUS-WIDE							
VSE-fs	0.0313	0.1567	0.0227	0.2273	0.0967	0.1272	0.6135
VSE-ens	0.0278 (+12.59)	0.1391 (+12.65)	0.0198 (+14.65)	0.1982 (+14.68)	0.0893 (+8.29)	0.1144 (+11.19)	0.5990 (+2.42 x)
WARP	0.0107 (+192.52)	0.0533 (+194.00)	0.0083 (+173.49)	0.0830 (+173.86)	0.0336 (+187.80)	0.0448 (+183.93)	0.5415 (+13.30 x)
Opt-AUC	0.0035 (+794.29)	0.0177 (+785.31)	0.0028 (+710.71)	0.0279 (+714.70)	0.0113 (+755.75)	0.0151 (+742.38)	0.5139 (+19.38 x)
Flickr							
VSE-fs	0.1267	0.6323	0.0732	0.7316	0.4456	0.5152	0.8647
VSE-ens	0.1143 (+10.85)	0.5716 (+10.62)	0.0696 (+5.17)	0.6957 (+5.16)	0.3887 (+14.64)	0.4621 (+11.49)	0.8453 (+2.30 x)
WARP	0.1251 (+1.28)	0.6256 (+1.07)	0.0726 (+0.83)	0.7262 (+0.74)	0.4375 (+1.85)	0.5070 (+1.62)	0.8609 (+0.44 x)
Opt-AUC	0.1001 (+26.57)	0.5039 (+25.48)	0.0664 (+10.24)	0.6635 (+10.26)	0.3078 (+44.77)	0.3925 (+31.26)	0.8288 (+4.33 x)
IAPR-TC12							
VSE-fs	0.0602	0.3011	0.0440	0.4413	0.1940	0.2501	0.7242
VSE-ens	0.0598 (+0.67)	0.2990 (+0.70)	0.0436 (+0.92)	0.4364 (+1.12)	0.1836 (+5.66)	0.2427 (+3.05)	0.7126 (+1.63 x)
WARP	0.0595 (+1.18)	0.2976 (+1.18)	0.0428 (+2.80)	0.4278 (+3.16)	0.1796 (+8.02)	0.2380 (+5.08)	0.7086 (+2.20 x)
Opt-AUC	0.0543 (+10.87)	0.2713 (+10.98)	0.0414 (+6.28)	0.4136 (+6.70)	0.1629 (+19.09)	0.2212 (+13.07)	0.7011 (+3.29 x)

of WARP model, we manually set the maximal number of sampling trials as 500 for OpenImages dataset in the training process. The results in Table 3 are based on this setting. Opt-AUC has an approximate per-iteration training time compared with VSE-fs because of the simple random-sampling strategy.

4.3 Retrieval Performance

Table 4 summarizes the retrieval performance of all the comparison models, where the best results of each model are reported. Generally, our model achieves the best performance among these models across different datasets and evaluation metrics. Specifically, the results reported on four datasets are very consistent and justify the correctness of our experiments. Opt-AUC produces the poorest performance, suggesting the value of advanced sampling strategies over uniform sampling. VSE-ens beats WARP and Opt-AUC to a large extent, which is consistent with the results reported by [1] on OpenImages, NUS-WIDE and Flickr. Even though, our approach with full-sample outperforms VSE-ens, which has a carefully designed negative sampler. To sum up, it is valuable to optimize VSE models with full non-positive examples, and it provides significant performance gains relative to negative sampling approaches.

4.4 Memory Usage

The majority of memory usage in our VSE-fs model lies in three parts: (1) the two derived cached matrices H^I and H^C (see Line 6 and Line 15 in Algorithm 1) which have the size of $K \times K$; (2) the positive weight matrix (see $w_{ic} \in W^{N \times M}$ in Eq. 1) which has the size of $|\mathcal{D}|$ since it is unnecessary to store the weights of all (i, c) pairs; (3) the negative weights β_c with the size of M (see Eq. 1).

Fig. 2 (a) illustrates the memory usage of the four comparison models on OpenImages. The results show that the memory usage of VSE-fs model is slightly higher than VSE-ens model while the WARP and Opt-AUC use the least memory. The rightmost bar denotes the memory usage of image and class embedding matrices which is the basic memory usage of VSE models. We omit the details in NUS-WIDE, Flickr and IAPR-TC12 since they share the similar effects. However, although VSE-fs has a high memory usage, RAM is not the bottleneck any more for training the model nowadays because of the low price of RAM, and it is a commonplace to run model on large RAM servers, in which most modern operating systems have excellent memory management strategy, and many third-party tools could be used to optimize the memory footprint. Furthermore, our model learning can be easily paralleled (discussed in Section 3.4), in which the embedding matrices can be stored in different servers and the memory usage of a single computer can be further reduced.

4.5 Effect of Parameters β_0, γ and α

Parameter β_0 controls the strength of the non-positive examples in our VSE-fs model (see Eq. 1). It has direct influence on the retrieval performance. We tune its value from 1 to 512 exponentially stepping by 2^n (n is the number of steps), and then finer-tune with a smaller interval to search the best settings. The results are shown in the left sub-figure of Fig. 2 (b) in terms of Pre@5 on OpenImages. We omit the details in other metrics and other datasets (for space saving) since they follow similar trends. It indicates that a proper setting of parameter β_0 is important to achieve the best performance, while smaller or larger values may lead to poor performance. In fact, the best settings of parameter β_0 are 16, 3, 128 and 8 for OpenImages, NUS-WIDE, IAPR-TC12 and Flickr, respectively.

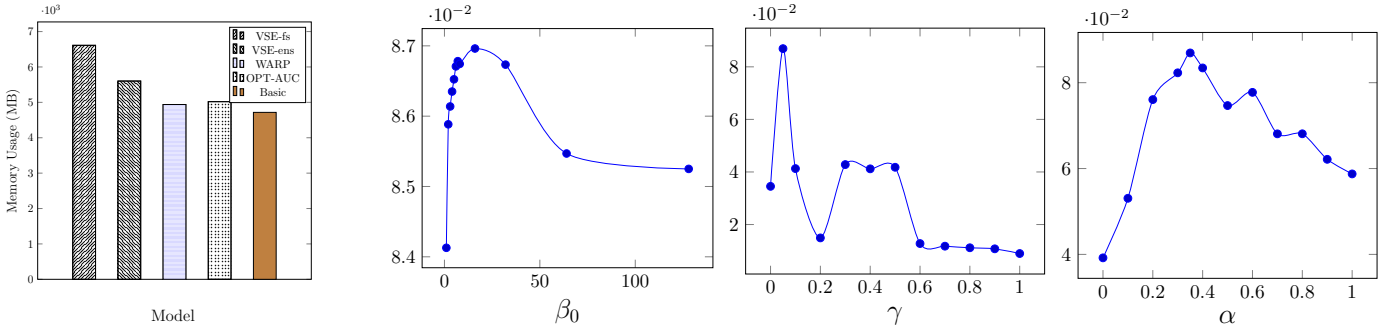

Fig. 2: (a). Memory usage of four comparison models on OpenImages. (b). Effects of the parameters β_0, γ and α on OpenImages in terms of Precision@5.

Parameter γ accounts for the strength of the extended vector $\mathbf{p}_i$ in scoring function (see Eq. 2) with the range in $[0, 1]$. Our new scoring function will turn to the ordinary scoring by setting $\gamma = 0$. We tune γ from 0 to 1 to find the best results as shown in the middle sub-figure of Fig. 2 (b) in terms of Pre@5 on OpenImages. The experimental results demonstrate that our scoring function gains improvements of 1.52 x on OpenImages compared with the ordinary scoring (i.e., $\gamma = 0$) and the optimal setting of γ is 0.01. Note that the retrieval performance dramatically decreases when setting $\gamma = 1$, which is consistently worse than the results of $\gamma = 0$ on OpenImages. One possible explanation is that high portion of $\mathbf{p}_i$ in scoring function ($\mathbf{v}_i + \gamma\mathbf{p}_i$) may disturb the optimization of image and class embedding matrices. In other words, the impact of class embedding vectors $\mathbf{v}_c$ will way overweight that of image embedding vectors $\mathbf{v}_i$ in the score, which may result in the issue of under-fitting. Note that the best settings of parameter γ are 0.02, 0.03 and 0.04 on NUS-WIDE, Flickr and IAPR-TC12 respectively, which also share the similar effects above mentioned.

Lastly, parameter α indicates the significant level of the non-positive weights related to the positive weights. We also set it in the range of $[0, 1]$ to find the best value. The results of tuning α on OpenImages are given in the right sub-figure of Fig. 2 (b). The experimental results shows that our VSE-fs model reaches the best performance with a proper value between 0 and 1, which outperforms the settings of both $\alpha = 0$ and $\alpha = 1$. Therefore, the weighting strategy of our model is superior to the uniform weighting strategy ($\alpha = 0$) as well as the basic '$popularity$' based weighting strategy ($\alpha = 1$). To be specific, our VSE-fs model reaches the best performance with $\alpha = 0.35$ on OpenImages. Note that the result of $\alpha = 1$ is better than that of $\alpha = 0$, implying that '$popularity$' based weighting strategy works better than the basic uniform weighting strategy. In addition, the optimal settings of parameter α are 0.85, 0.43 and 0.52 on NUS-WIDE, Flickr and IAPR-TC12, respectively.

5 Conclusion

In this paper, we studied the efficiency problem of visual-semantic embedding (VSE) models. We analytically pointed out that traditional approaches to sample negative examples may lead to performance loss due to the potential missing of informative examples. Thus, we proposed a novel VSE model with full-sample optimization (named *VSE-fs*) rather than only a sampled portion of negative examples. To reduce time complexity and boost model training, we devised two *separate*-transformation strategies, aiming to simplify the computation of inner-product operations and the traversing over all non-positive sample. The experimental results on four datasets demonstrated that our approach can obtain significant performance gains in terms of both accuracy and convergence, comparing with state-of-the-art models with negative sampling.

Acknowledgments

This work is supported by the National Natural Science Foundation for Young Scientists of China under Grant (No. 61772125, 61702084 and 61702090), and by the Fundamental Research Funds for the Central Universities under Grant No. N181705007.

References

[1] G. Guo, S. Zhai, F. Yuan, Y. Liu, and X. Wang, "VSE-ens: Visual-semantic embeddings with efficient negative sampling," in *AAAI Conference on Artificial Intelligence*. AAAI Press, 2018, pp. 290–297.

[2] J. Weston, S. Bengio, and N. Usunier, "WSABIE: Scaling up to large vocabulary image annotation," in *Proceedings of the International Joint Conference on Artificial Intelligence*. IJCAI/AAAI, 2011, pp. 2764–2770.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2015, pp. 1–9.

[4] D. Grangier and S. Bengio, "A discriminative kernel-based approach to rank images from text queries," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 8, pp. 1371–1384, 2008.

[5] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *the 27th Annual Conference on Neural Information Processing Systems*. Curran Associates, Inc., 2013, pp. 3111–3119.

[6] B. Yoshua and S. J S, "Adaptive importance sampling to accelerate training of a neural probabilistic language model," *IEEE Transactions on Neural Networks*, vol. 19, no. 4, pp. 713–722, 2008.

[7] I. Krasin, T. Duerig, N. Alldrin, and Ferrari, "OpenImages: A public dataset for large-scale multi-label and multi-class image classification." *Available from https://github.com/openimages*, 2017.

[8] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "NUS-WIDE: A real-world web image database from national university of singapore," in *Proceedings of the ACM International Conference on Image and Video Retrieval*. ACM, 2009, pp. 1–9.

[9] H. J. Escalante, C. A. Hernandez, and J. A. Gonzalez, "The segmented and annotated IAPR TC-12 benchmark," *Computer Vision and Image Understanding*, vol. 114 Issue 4, no. 4, pp. 419–428, 2010.

[10] X. He, H. Zhang, M. Kan, and T. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *SIGIR*. ACM, 2016, pp. 549–558.

Songlin Zhai is a graduate student in the Software College, Northeastern University, China. He received his B.S. degree in computer science and technology from Yanshan University, China, in 2017. His research interests include recommender systems, image classification and knowledge base question answering. His email address is 1771061@stu.neu.edu.cn.

Guibing Guo is currently a professor with the Software College, Northeastern University, China. He received his Ph.D. degree in computer science from Nanyang Technological University, Singapore, in 2015. His research interests include recommender systems, deep learning, natural language processing and data mining. His email address is guogb@swc.neu.edu.cn.

Fajie Yuan is now a senior researcher in the PCG (Platform & Content) group of Tencent Inc., China. He received his Ph.D. in the School of Computing Science at the University of Glasgow in 2018. His research interests span recommender system, natural language processing and transfer learning. His work have appeared in several top-tier conferences, including UAI, WSDM, IJCAI, and ACL, and he has won the best student paper award in ICTAI 2016. His email address is fajieyuan@tencent.com.

Yuan Liu is currently an Associate Professor with the Software College, Northeastern University, China. She received the Ph.D. degree from the School of Computer Engineering, Nanyang Technological University, Singapore, in 2014. Her research interests include trust-based incentive mechanism design, multi-agent systems, trust management, and blockchain technology-based reputation systems. Her email address is liuyuan@swc.neu.edu.cn.

Xingwei Wang is currently a Professor at the College of Software, Northeastern University, China. He received the B.S., M.S., and Ph.D. degrees in computer science from the Northeastern University, Shenyang, China, in 1989, 1992, and 1998 respectively. His research interests include cloud computing and future Internet, etc. His email address is wangxw@swc.neu.edu.cn.