Improving Visual-Semantic Embedding with Adaptive Pooling and Optimization Objective

Zijian Zhang∗,1, Chang Shu∗2,3, Ya Xiao1, Yuan Shen1, Di Zhu1, Jing Xiao2, Youxin Chen2, Jey Han Lau4, Qian Zhang3 and Zheng Lu3

1 Meituan, China
2 Ping An Technology (Shenzhen) Co., Ltd, China
3 University of Nottingham Ningbo, China
4 The University of Melbourne, Australia

Abstract

Visual-Semantic Embedding (VSE) aims to learn an embedding space where related visual and semantic instances are close to each other. Recent VSE models tend to design complex structures to pool visual and semantic features into fixed-length vectors and use hard triplet loss for optimization. However, we find that: (1) combining simple pooling methods is no worse than these sophisticated methods; and (2) only considering the most difficult-to-distinguish negative sample leads to slow convergence and poor Recall@K improvement. To this end, we propose an adaptive pooling strategy that allows the model to learn how to aggregate features through a combination of simple pooling methods. We also introduce a strategy to dynamically select a group of negative samples to make the optimization converge faster and perform better. Experimental results on Flickr30K and MS-COCO demonstrate that a standard VSE using our pooling and optimization strategies outperforms current state-of-the-art systems (at least 1.0% on the metrics of recall) in image-to-text and text-to-image retrieval. Source code of our experiments is available at https://github.com/96-Zachary/vse_2ad.

1 Introduction

Visual Semantic Embedding (VSE) (Frome et al., 2013; Faghri et al., 2018) is a representation learning method that embeds images and texts for efficient cross-modal retrieval, and typically has the following steps: (1) the image and text features are first extracted by separate visual and text encoders; (2) these features are then projected into a joint embedding space and pooled to form fixed-length vectors; (3) similarity is computed to measure the distance between image and text representation; and (4) a suitable objective is chosen for optimization (see Figure 1 for an illustration). Our paper focuses on improving the steps of feature aggregation and optimization.

For feature aggregation, the most commonly used methods are simple pooling aggregators. MaxPool (Wang et al., 2018) and MeanPool (Reimers and Gurevych, 2019) are designed to detect the salient and mean points of features, and K-MaxPool (Kalchbrenner et al., 2014) extracts the mean of top-K features. Some complex aggregation techniques have been proposed, e.g. local-importance projection (Gao et al., 2019), sequence-to-sequence encoder (Hu et al., 2019), graph convolution network (Li et al., 2019), exponential adaptive pooling (Stergiou and Poppe, 2021) and self-attention encoder (Wang et al., 2020). However, we found that carefully selected pooling functions can surpass complex methods (see Appendix A.1). Motivated by this, our paper proposes an approach that can automatically discover the best pooling functions. Specifically, we seek to improve the feature aggregation step by proposing a formulation that parameterizes the different pooling strategies and allows the model to learn the best configuration automatically via its objective, alleviating the need to do manual tuning. In other words, we’ve turned these hyper-parameters (i.e. choices of pooling functions) into parameters in the model.

For optimization, most VSE models optimize using the hinge triplet ranking loss with in-batch negative example (Faghri et al., 2018). The intuition of the objective is to encourage positive pairs to be embedded in a similar space while widening the
distance between a target with the hardest in-batch negative sample (Li et al., 2022a). In practice, however, it is often difficult for the model to find a good negative sample in the early stages of training (as instances are randomly distributed in space), resulting in slow convergence (see Appendix A.2). To improve optimization, we propose an adaptive optimization objective that selects multiple in-batch negative samples based on model quality during training. The intuition is that in the early stages of training, we want to sample more negative samples for faster convergence, and in the later stages, we want to sample fewer negative samples for finer tuning.

Over two public datasets, MS-COCO (Lin et al., 2014) and Flickr30K (Young et al., 2014), we show that a standard VSE model using our proposed feature aggregation and optimization strategies outperforms benchmark models substantially. In particular, our method obtains 1.4% relative gains on RSUM for MS-COCO and 1.0% for Flickr30K. Compared with a pre-trained vision-language model with similar performance (Geigle et al., 2022), our method is $4.3 \times$ faster.

## 2 Related Work

Depending on whether the image and text features have any form of cross-modal interaction before similarity calculation, existing image-text retrieval can be broadly categorized into two types.

The visual semantic embedding (VSE) (Faghri et al., 2018; Wang et al., 2020; Chun et al., 2021) methods process the multimodal instances independently before projecting them into a joint embedding space for similarity matching. Wang et al. (2018) design a two-branch neural networks, LIWE (Wehrmann et al., 2019) considers character-based alignment and embedding methods for language encoder and Faghri et al. (2018) extend it by using hard triplet loss for optimization. Following these ideas, PVSE (Song and Soleymani, 2019) and CVSE (Wang et al., 2020) are proposed to consider intra-modal polysemous and consensus information. Recently, Chun et al. (2021) samples instances as probabilistic distributions and achieves further improvement. These VSE-based methods are fast as they do not consider cross-modal interaction and as such the visual and text features can be pre-computed. The non-VSE methods concentrate on the interaction of modalities. For example, prior methods explore fusing multi-modal information by attention (Lee et al., 2018; Chen et al., 2020; Zhang et al., 2021a), alignment (Zhang et al., 2020), multi-view representation (Qu et al., 2020) or fine-grained reasoning (Qu et al., 2021; Zhang et al., 2022). The early-interaction methods (Geigle et al., 2022), like pre-trained vision-language models (Lu et al., 2019; Chen et al., 2019; Li et al., 2020; Jia et al., 2021; Li et al., 2022b), focus on maximizing performance while trading off efficiency.

The performance of VSE ultimately depends on the quality of the joint embedding space, which is usually learned with simple transformations (e.g. linear projection or multi-layer perceptron) and pooling aggregators (e.g. mean pooling (Faghri et al., 2018; Qu et al., 2020), max pooling (Zhang and Lu, 2018; Li et al., 2021), or a combination of them (Lee et al., 2018)). Compared to these simple aggregation methods, more complex aggregators that introduce a large number of trainable parameters have also been explored, e.g. inter-modal attention (Wehrmann et al., 2020) and self-attention mechanisms (Han et al., 2021). Zhang et al. (2021b) design a cross-modal guided pooling module that attends to local information dynamically. These sophisticated aggregators typically require more time, and don’t always outperform simple pooling strategies. Perhaps the closest study to our work is GPO (VSE∞) (Chen et al., 2021), which builds a generalized operator to learn the best pooling strategy that only considers the position information of the extracted features.

Some studies focus on improving the optimization objective, and the most widely adopted objective is the hinge-based hard triplet ranking loss (Faghri et al., 2018; Wei et al., 2020b; Messina et al., 2021), which dynamically selects the “hardest” negative sample within a mini-batch. Other studies explore solutions that choose multiple negative samples. Zhou et al. (2020) introduce a coherence metric to rank the “irrelevant” candidates. Extending the idea, Wei et al. (2020a) assign different weights for positive and negative pairs. To tackle the issue of noisy labels which impacts multimodal representation, Hu et al. (2021) propose maximizing the mutual information between different modalities. Huang et al. (2021) separate data into “clean” and “noisy” partitions by co-teaching. Ultimate though these methods did not explore modifying the behaviour of negative sample selection adaptively according to model performance during training, which we address in this work.
3 Methodology

3.1 Background of VSE

We first discuss the standard formulation of VSE, before introducing our innovation on improving feature aggregation (Section 3.2) and optimization (Section 3.3).

To compute the similarity of given multimodal instance (image & text), a VSE model (Figure 2) separately encodes them via a visual encoder (VisEnc(·)) and a text encoder (TextEnc(·)). There are three widely used visual features produced by different visual encoders — grid is the feature maps from convolutional networks (CNNs; He et al. (2016)), region is the region of interest features from object detectors (Anderson et al., 2018) and patch is the partition from vision transformer (Dosovitskiy et al., 2021). The text encoders are usually RNNs (Sutskever et al., 2014)) and BERT (Devlin et al., 2019). Formally:

\[ F_v = \text{VisEnc}(v) \]
\[ F_t = \text{TextEnc}(t) \]

where \( v \) and \( t \) are the input image and text.

Assuming the visual feature \( F_v \) has \( N \) object vectors (represented either as grids, regions or patches) in \( d_1 \) dimension, and the text feature \( F_t \) has \( M \) token vectors in \( d_2 \) dimension, we next project them to the same \( d \)-dimension:

\[ \{v_n\}_{n=1}^{N} = F_v W_v + b_v \]
\[ \{t_m\}_{m=1}^{M} = F_t W_t + b_t \]

where \( v_n \) and \( t_m \) now have the same dimension \( d \).

To aggregate the extracted features into fixed-length vectors, domain aggregators, \( f_{\text{vision}}(\cdot) \) and \( f_{\text{text}}(\cdot) \) are used to transform \( \{v_n\}_{n=1}^{N} \in \mathbb{R}^{N \times d_v} \) and \( \{t_m\}_{m=1}^{M} \in \mathbb{R}^{M \times d_t} \) into \( v \in \mathbb{R}^d \) and \( t \in \mathbb{R}^d \), respectively:

\[ v = f_{\text{vision}}(\{v_n\}_{n=1}^{N}) \]
\[ t = f_{\text{text}}(\{t_m\}_{m=1}^{M}) \]

And lastly, to measure how related the inputs we use cosine similarity:

\[ \text{sim}(t, v) = \frac{t^T v}{||t|| \cdot ||v||} \]

Existing optimization strategies generally use the hinge-based triplet ranking loss to optimize the VSE model. Given an anchor, it aims to maximize its similarity with positive samples while minimizing its similarity with the most “difficult” negative sample in the mini-batch (i.e. the example that has the highest similarity with the anchor that is not a positive example), and includes both text-to-image and image-to-text retrieval objectives:

\[ L_{\text{HardTriplet}} = \sum_{(t, v, \hat{t}, \hat{v}) \sim B} \left[ \alpha - \text{sim}(t, v) + \text{sim}(t, \hat{v}) \right]^+ + \left[ \alpha - \text{sim}(t, \hat{v}) + \text{sim}(\hat{t}, v) \right]^+ \]

where \( \alpha \) is the margin hyper-parameter, and \([x]^+ = \max(0, x)\). \((t, v)\) is a positive text-image pair in mini-batch \( B \) and \((\hat{t}, v)\) and \((t, \hat{v})\) are negative pairs, where \( t = \arg\max_{\hat{t}' \neq v} \text{sim}(\hat{t}', v) \) and \( \hat{v} = \arg\max_{\hat{v}' \neq v} \text{sim}(t, \hat{v}') \) are the hardest negative sentence and image respectively in \( B \).
3.2 Enhanced VSE with Adaptive Pooling

We now describe our approach (named adaptive pooling, ADPOOL) to improving feature aggregation, where we parameterize the pooling operations ($f_{\text{vision}}$ and $f_{\text{ext}}$) to allow VSE to learn the best way to aggregate the features via its objective adaptively (in other words, we have effectively turned the pooling methods — which are hyper-parameters — into model parameters). Our approach has the advantage of being much faster and simpler than complex aggregators, as it introduces only a small number of parameters. Note that we will describe our method only for the text feature (red rows in the input matrix) as an example, we follow the “sort → weight-sum” paradigm (Grefenstette et al., 2014) of simple pooling methods to first sort the feature matrix along the embedding axis. Next we let the model learn (via a fully-connected layer) how to weight each vector (red rows in the matrix) automatically (e.g. [0.1, 0.6, 0.3, 0.0] in Figure 3). Intuitively, this means we have now expressed MeanPool and K-MaxPool using a generalized function (through the “sort → weight-sum” process), and the parameters of the function is learned automatically so the feature aggregation step is “adaptive”. Formally:

$$t_{\text{tok}}^k = \sum_{m=1}^{M} \theta_m \cdot u_m$$
$$\{u_m\}_{m=1}^M = \text{sort}\left(\{t_m\}_{m=1}^M\right)$$

(3)

$$\theta = \text{softmax}\left(\{u_m\}_{m=1}^M \cdot W_{\text{tok}}\right)$$

where $\text{sort}()$ is a function that sorts the token vectors along the embedding axis.

3.2.1 Token-level Pooling

Recall that the text feature ($\{t_m\}_{m=1}^M$) has M tokens each of d-dimension. The exact form of $t_m$ can be token vectors. As for vision features, they can be grids, regions or image patches, depending on the visual encoder. Let $\text{max}_k(\cdot)$ be a function that extracts the top $k$-th value in a list, the common pooling mechanisms can be formulated as follows:

- **MeanPool** = $\frac{1}{M} \sum_{i=1}^{N} \{t_{ij}\}_{i=1}^{M}$ outputs the mean value among $N$ objects at position $j$;
- **MaxPool** = $\text{max}_1(\{t_{ij}\}_{i=1}^{M})$ returns the maximum values of $\{t_{ij}\}_{i=1}^{M}, \forall j$;
- **K-MaxPool** = $\frac{1}{K} \sum_{k=1}^{K} \text{max}_k(\{t_{ij}\}_{i=1}^{M})$ computes the mean of the top-$K$ values.

where $t_{ij}$ denotes the $j$-th element in the $i$-th token vector.

Using the token vectors $\{t_m\}_{m=1}^M$ in Figure 3 (red rows in the input matrix) as an example, we follow the “sort → weight-sum” paradigm (Grefenstette et al., 2014) of simple pooling methods to first sort the feature matrix along the embedding axis. Next we let the model learn (via a fully-connected layer) how to weight each vector (red rows in the matrix) automatically (e.g. [0.1, 0.6, 0.3, 0.0] in Figure 3). Intuitively, this means we have now expressed MeanPool and K-MaxPool using a generalized function (through the “sort → weight-sum” process), and the parameters of the function is learned automatically so the feature aggregation step is “adaptive”. Formally:

$$t_{\text{tok}}^k = \sum_{m=1}^{M} \theta_m \cdot u_m$$
$$\{u_m\}_{m=1}^M = \text{sort}\left(\{t_m\}_{m=1}^M\right)$$

(3)

$$\theta = \text{softmax}\left(\{u_m\}_{m=1}^M \cdot W_{\text{tok}}\right)$$

where $\text{sort}()$ is a function that sorts the token vectors along the embedding axis.

3.2.2 Embedding-level Pooling

With token-level pooling, we learn how to weight each (sorted) token vector and aggregate them. With embedding-pooling, we learn how to weight each (original unsorted) object vector *in each embedding dimension* to extract the most salient element within that dimension, and can be interpreted as a “soft” version of MaxPool:

$$t_{\text{emb}}^k = \sum_{i=1}^{M} \delta_{ij} \cdot t_{ij}, \forall j$$

$$\delta_{ij} = \frac{e^{\epsilon_{ij}}}{\sum_{i=1}^{M} e^{\epsilon_{ij}}}, \forall i$$

(4)
3.2.3 Combining Token-level and Embedding-level Pooled Vectors

Given the two pooled vectors \( t_{\text{tok}} \) and \( t_{\text{emb}} \) (which have the same dimension \( d \)), one straightforward way to combine them is to weight them with scaling hyper-parameters. To avoid these hyper-parameters (which requires manual tuning), we let the model learn these weights automatically with a trainable \( W_{\text{bal}} \in \mathbb{R}^{d \times 1} \):

\[
t = \omega_1 t_{\text{tok}} + \omega_2 t_{\text{emb}}
\]

\[
\omega_{1,2} = \text{softmax} \left( [t_{\text{tok}}W_{\text{bal}}, t_{\text{emb}}W_{\text{bal}}] \right)
\]  

(5)

For the visual feature \( \{v_n\}_{n=1}^N \) (Equation 1), we follow the same process to compute the token-level (where here an “token” is a unit of image) and embedding-level pooled vectors \( \{v_{\text{tok}} \text{ and } v_{\text{emb}}\} \) and combine them to produce \( v \in \mathbb{R}^d \).

3.3 Enhanced VSE with Adaptive Objective

Our next contribution is in improving the optimization step. The hard triplet loss (Equation 2) is the most commonly utilized training objective for optimizing VSE. However, we find that locating a “most-difficult” negative sample is challenging in the early stages of training (Appendix A.2), resulting in delayed convergence. We introduce a novel adaptive optimization objective, ADAPT, that automatically (or adaptively) selects \( K (K \geq 1) \) negative samples in each mini-batch \( B \) during training.

Wang and Isola (2020) introduce two key properties, Alignment and Uniformity, that correlate with a retrieval performance:

- **Alignment**: the positive \( (t \text{ and } v) \) should be mapped into a similar embedding space. Defining \( \gamma_{\text{align}} \in [0, 1] \) as the average similarity values for all positive pairs, the larger \( \gamma_{\text{align}} \) is, the better the VSE model.

\[
\gamma_{\text{align}} = \mathbb{E}_{(t,v) \sim B} [\text{sim}(t,v)]
\]

- **Uniformity**: all vectors should be roughly uniformly distributed on the unit hypersphere, and \( \gamma_{\text{uniform}} \in [0, 1] \) measures this quality.

\[
\gamma_{\text{uniform}} = \log \mathbb{E}_{t \sim B, v \sim B} [e^{\text{sim}(t,v)}]
\]

By combining \( \frac{\gamma_{\text{align}} + \gamma_{\text{uniform}} \times \pi}{4} \in [0, \frac{\pi}{2}] \), we can use it to assess the maturity of a VSE model during training and dynamically adjust the number of negative samples. That is, in the early stages of training, we expect the value to be close to 0 (as the model is unable to differentiate between positive and negative samples), and we would want to use more negative samples for optimization. Conversely in the later stages of training, the value should be close to \( \frac{\pi}{2} \) and we want less negative samples. Formally:

\[
K' = \text{round} \left( |B| \times \cos \left( \frac{(\gamma_{\text{align}} + \gamma_{\text{uniform}} \times \pi)}{4} \right) \right)
\]

\[
K = \max \left( 1, \min(K', |B| - 1) \right)
\]

(6)

where \( \cos \) is the cosine function to invert the sum\(^1\), \( K \) is the number of negative samples to be used, and round(\( x \)) is a function that rounds down the value of \( x \). As the hard triplet loss can only work with \( K = 1 \) (Equation 2), we therefore utilize the InfoNCE objective (Van den Oord et al., 2018), which is a commonly used contrastive objective (Radford et al., 2021):

\[
\mathcal{L} = \mathcal{L}_{v2t} + \mathcal{L}_{t2v}
\]

\[
\mathcal{L}_{v2t} = -\frac{1}{|B|} \sum_{(t,v) \in B} \log \frac{\exp(\text{sim}(t,v)/\tau)}{\sum_k \exp(\text{sim}(t_k,v)/\tau)}
\]

\[
\mathcal{L}_{t2v} = -\frac{1}{|B|} \sum_{(t,v) \in B} \log \frac{\exp(\text{sim}(t,v)/\tau)}{\sum_k \exp(\text{sim}(t,v_k)/\tau)}
\]

where \( \tau \) is the temperature hyper-parameter.

4 Experiments

4.1 Datasets and Metrics

**Datasets.** We conduct our experiments on MS-COCO and Flickr30K using various visual and text encoders for cross-modal retrieval. The MS-COCO dataset contains 123,287 images, each with 5 manually annotated sentences. Following the split method of Faghri et al. (2018), 113,287 images are used for training, 5,000 for validation, and 5,000 for testing.

Following prior studies (Faghri et al., 2018), we experiment with two evaluation settings: (1) MS-COCO 1K averages the results over 5-folds of 1K test images; and (2) MS-COCO 5K directly results on the whole 5K test images. Following Chen et al. (2021), we use the former to assess overall performance with state-of-the-art VSE models and the

\(^1\)The purpose of using \( \cos \) is to map \( \gamma_{\text{align}} + \gamma_{\text{uniform}} \times \pi \in [0, \pi/2] \) to the range of \([1, 0]\).
Table 1: Cross-modal retrieval results on MS-COCO and Flickr30K datasets. The top half of the table uses BiGRU as the text encoder; the bottom half is BERT. VF denotes vision feature, and R, G, and P mean region, grid and patch respectively. The best results are marked bold in black, blue and red for region feature (R), ensemble of region+grid features (RG) and ensemble of region+grid+patch features (RGP) respectively. VSE+2AD is our proposed model, which enhances the VSE model by using AdPOOL for aggregation and ADOPT for optimization.

<table>
<thead>
<tr>
<th>Text Encoder: BiGRU</th>
<th>Image Retrieval</th>
<th>Caption Retrieval</th>
<th>RSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>VF</td>
<td>R@1</td>
<td>R@5</td>
<td>R@10</td>
</tr>
<tr>
<td>R VSE++</td>
<td>54.0</td>
<td>85.6</td>
<td>92.7</td>
</tr>
<tr>
<td>R LITE</td>
<td>57.9</td>
<td>88.3</td>
<td>94.5</td>
</tr>
<tr>
<td>R PVSE</td>
<td>55.2</td>
<td>86.5</td>
<td>93.7</td>
</tr>
<tr>
<td>R CVSE∞</td>
<td>55.7</td>
<td>86.9</td>
<td>93.8</td>
</tr>
<tr>
<td>R VSE∞</td>
<td>61.7</td>
<td>90.3</td>
<td>95.6</td>
</tr>
<tr>
<td>R SCAN(i2t)</td>
<td>54.4</td>
<td>86.0</td>
<td>93.6</td>
</tr>
<tr>
<td>R SCAN(t2i)</td>
<td>56.4</td>
<td>87.0</td>
<td>93.9</td>
</tr>
<tr>
<td>R CAAN</td>
<td>61.3</td>
<td>89.7</td>
<td>95.2</td>
</tr>
<tr>
<td>R IRAMR</td>
<td>61.7</td>
<td>89.1</td>
<td>95.0</td>
</tr>
<tr>
<td>R VSE+2AD</td>
<td>63.5</td>
<td>91.8</td>
<td>96.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Text Encoder: BERT</th>
<th>Image Retrieval</th>
<th>Caption Retrieval</th>
<th>RSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>VF</td>
<td>R@1</td>
<td>R@5</td>
<td>R@10</td>
</tr>
<tr>
<td>R VSE++</td>
<td>54.0</td>
<td>85.6</td>
<td>92.5</td>
</tr>
<tr>
<td>R VSE∞</td>
<td>64.8</td>
<td>91.6</td>
<td>96.3</td>
</tr>
<tr>
<td>R DSNR</td>
<td>64.5</td>
<td>90.8</td>
<td>94.8</td>
</tr>
<tr>
<td>R DIME(i2t)</td>
<td>63.0</td>
<td>90.5</td>
<td>96.2</td>
</tr>
<tr>
<td>R DIME(t2i)</td>
<td>62.3</td>
<td>90.2</td>
<td>95.8</td>
</tr>
<tr>
<td>R VSE+2AD</td>
<td>67.5</td>
<td>93.6</td>
<td>97.7</td>
</tr>
</tbody>
</table>

Latter for further analyses such as ablation results. Flickr30K consists of 31,783 images with the same corresponding 5 captions, and 1,000 images are reserved for validation and testing.

Metrics. We evaluate cross-modal retrieval performance using recall@K (R@K), where K = {1, 5, 10}. The evaluation tasks include both caption retrieval (given caption, find images) and image retrieval (given image, find captions). We also compute RSUM which is a sum of all R@K metrics across both tasks to assess the overall performance.

4.2 Implementation Details

We implement our models using the PyTorch library. The dimension of the shared embedding space $d$ is 1024. We use the Adam optimizer with a mini-batch size of 128 and train our models with 25 epochs. The learning rate is set to 5e-4 with a decaying factor of 10% for every 15 epochs.

**Visual Encoders.** We use Faster-RCNN (Ren et al., 2015) (ResNet-101 is pre-trained on ImageNet and Visual Genome) to extract region feature directly (Anderson et al., 2018), and fine-tune it further with grid feature (resolution = 512×512) (Jiang et al., 2020) before using it as a grid feature extractor. For the patch feature, we fine-tune the pre-trained Vision Transformer (Dosovitskiy et al., 2021) using images with a resolution of 224×224 to use it as a patch feature extractor.

**Text Encoders.** We experiment with BiGRU (Faghri et al., 2018) and BERT-base (Devlin et al., 2019). Additional implementation and training details are given in the Appendix B.

4.3 Main Results

We compare against several mainstream baselines with our proposed method, all of which have been described in Section 2. For these benchmark models, we use the results reported in the original papers (test set is the same).

Table 1 presents the full results over both image and caption retrieval and across two datasets (MS-
COCO and Flickr30K). Results for MS-COCO is an average of over 5-folds of 1K test images. Our method is “VSE+2AD”, which enhances the standard VSE model by introducing AdPool and AdOpt. The top half of the table presents results where we use BiGRU as the text encoder, and the bottom half uses BERT. For models where we combine visual features from multiple visual encoders (e.g. “RG” which combines region and grid feature), we do so by simply taking the average similarity values to rank the candidates.

Looking at the results, VSE+2AD (our model) outperforms almost all baselines/benchmark models consistently. Our model displays consistent improvement over the state-of-the-art method VSE∞ (Chen et al., 2021) with the same visual (region feature by BUTD (Anderson et al., 2018)) and text encoders (BiGRU). In particular, with BiGRU as the text encoder, it obtains 1.4% (520.5 → 527.8) and 1.0% (498.1 → 503.1) relative gains on RSUM for MS-COCO and Flickr30K datasets respectively. These improvements are consistent irrespective of the combination of visual and text encoders. We also see that combining visual encoders (“RG” vs. “R”) further boosts its performance (like RSUM from 527.8 to 534.4 for MS-COCO), and utilizing all types of visual features (“RGP”) produces the best performance (539.1 > 534.4 > 527.8).

4.4 Comparison with Pre-trained Models

We next compare VSE+2AD (with BERT as the text encoder) to pre-trained vision language models: ViLBERT (Lu et al., 2019), UNITER (Chen et al., 2019), OSCAR (Li et al., 2020), ALIGN (Jia et al., 2021), CLIP (Radford et al., 2021) and MVP (Li et al., 2022b) in Table 2. These results are evaluated on the COCO 5K test images.

Without using any large-scale paired image-text corpus for pre-training, our ensemble VSE+2ADRGP (that combines region/grid/patch features) outperforms two out of six pre-trained VL methods and is not much worse than OSCAR, even though it does not use any cross-modal interaction. Our model is substantially faster than these pre-trained models: our slowest ensemble model is still an order of magnitude faster. As for our method we can pre-compute and cache the visual and text features, so the retrieval only need similarity calculation and ranking operations. Overall, these results demonstrate that our model strikes a good balance between performance and efficiency.

4.5 Ablation Study

To understand the impact of AdPool (which improves feature aggregation in Section 3.2) and AdOpt (which improves optimization in Section 3.3), we perform several ablation studies based on the COCO 5K test. For this experiment (in Table 3), we use only the region feature and BiGRU as the text encoder. Looking at the aggregate RSUM performance, we see both the token-level pooling (“-tok AdPOOL”; Section 3.2.1) and embedding-level pooling (“-emb AdPOOL”; Section 3.2.2) appear to be useful, although token-level pooling is arguably more important. In the case where we remove AdPool entirely (“-AdPOOL”) and use MeanPool as the aggregation method, the performance drops even more, suggesting complementarity. AdOpt is the most impactful method, where taking it out produces the worst performance. To understand its impact qualitatively, we also look at the training curve of a VSE model trained with and without AdOpt. We found that that AdOpt is particularly useful in the early stages of train-
Table 4: Performance of VSE+2AD using different aggregators. #OIPs denotes operating items per second, and larger is better. Note that our proposed ADPOOL aggregation method is only slower than simple pooling, 5.6× > 4.9×(ADPOOL) > others, but it no need for fussily manual tuning.

<table>
<thead>
<tr>
<th>Aggregator</th>
<th>R@1</th>
<th>R@5</th>
<th>R@1</th>
<th>R@5</th>
<th>RSUM</th>
<th>#OIPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIP</td>
<td>38.4</td>
<td>69.3</td>
<td>52.2</td>
<td>81.1</td>
<td>1414.8</td>
<td>4.6×</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>38.2</td>
<td>68.9</td>
<td>52.1</td>
<td>80.9</td>
<td>1415.7</td>
<td>1.8×</td>
</tr>
<tr>
<td>GCN</td>
<td>40.7</td>
<td>71.3</td>
<td>53.5</td>
<td>81.5</td>
<td>1419.2</td>
<td>1.0×</td>
</tr>
<tr>
<td>SelfAttn</td>
<td>40.9</td>
<td>71.3</td>
<td>53.8</td>
<td>82.2</td>
<td>1420.1</td>
<td>1.6×</td>
</tr>
<tr>
<td>AdaPool</td>
<td>40.2</td>
<td>71.8</td>
<td>52.8</td>
<td>81.4</td>
<td>1418.3</td>
<td>4.1×</td>
</tr>
<tr>
<td>SoftPool</td>
<td>39.2</td>
<td>69.8</td>
<td>52.8</td>
<td>81.4</td>
<td>1418.3</td>
<td>5.1×</td>
</tr>
<tr>
<td>GPO</td>
<td>41.2</td>
<td>71.1</td>
<td>55.6</td>
<td>82.0</td>
<td>1422.9</td>
<td>4.3×</td>
</tr>
<tr>
<td>Manual</td>
<td>38.7</td>
<td>71.6</td>
<td>53.7</td>
<td>82.1</td>
<td>1419.3</td>
<td>5.6×</td>
</tr>
<tr>
<td>ADPOOL</td>
<td>41.4</td>
<td>72.3</td>
<td>55.9</td>
<td>83.0</td>
<td>1426.9</td>
<td>4.9×</td>
</tr>
</tbody>
</table>

5.6 Case Study

To validate the effectiveness of VSE+2AD, we present two examples for image retrieval and caption retrieval in Figure 4(a) and Figure 4(b), respectively. As we can see in Figure 4(a), the “incorrect” sentence retrieved by our model seems sensible, suggesting that this is likely noise in the data. Figure 4(b), on the other hand, shows some genuine erroneous images retrieved by our model, and we suspect this is because it is a particularly difficult example where the caption is very descriptive and the details are difficult to be captured by VSE’s bi-encoder approach.

5 Conclusion and Discussion

In this paper, we propose methods to improve VSE’s feature aggregation and optimization objective. For the former, we introduce a way to parameterize the aggregation function to allow the visual and text encoders to learn the best way to combine their features to produce a fixed-size embedding. For the latter, we propose a method that dynamically selects many negative samples that allow the VSE to converge faster with a better final performance. We compare our enhanced VSE model to several baselines and state-of-the-art models over two public datasets and demonstrate that it marries both performance (state-of-the-art retrieval results) and efficiency (orders of magnitude faster than pre-trained models). In terms of practical impact, we also tested a real-world application of our method in Dianping (a Chinese searching platform) for a unimodal retrieval task (semantic textual similarity) and found that it worked very well, demonstrating its applicability to real-world applications.
Limitations

Our method’s performance has only been verified on English datasets; other languages can be explored to understand whether these results translate to them. In a similar vein, we’ve also only tested our methods in domain-general datasets; cross-modal retrieval is a particularly important task in the general intelligence domain, and with limited paired image-text datasets it would be interesting to see whether our proposed enhancements work with these specialised domains. Lastly, our idea of parameterizing the pooling and optimization strategies can technically be applied to tasks other than cross-modal retrieval, and it’s something worth exploring in future works.

Acknowledgements

We are grateful for the support of Biao Tang and Gong Zhang from the platform search group in Meituan-Dianping for testing the practical utility of our method. This work was supported by Ningbo Science and Technology Bureau under Major Projects Fund with project code 2021Z089 and 2022Z060.

References


Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. 2021. Scaling up visual and vision-language representation learning with noisy text supervision. In ICML.


Chenxi Zhang, Zijian Zhang, Jiangfeng Li, Qin Liu, and Hongming Zhu. 2021a. Ctnr: Compress-then-reconstruct approach for multimodal abstractive summarization. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.


Appendix

A Verification of Assumptions

A.1 Effectiveness of Simple Pooling Methods

Although various complex methods (described in Section 1 and Section 2) are explored for aggregation, we find that the simple pooling strategy works no worse than those complex methods by numerous experiments. It needs to be carefully manually tuned, like 5-MaxPool for the visual feature (region feature from BUTD) and MeanPool for the text feature (extracted by BiGRU). The results are shown in Table A.1-1, where the similar conclusion is also verified by VSE∞ (Chen et al., 2021).

<table>
<thead>
<tr>
<th>Aggregator</th>
<th>R@1</th>
<th>R@5</th>
<th>R@1</th>
<th>R@5</th>
<th>RSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIP</td>
<td>38.4</td>
<td>69.3</td>
<td>52.2</td>
<td>81.1</td>
<td>414.8</td>
</tr>
<tr>
<td>Seq2Seq</td>
<td>38.2</td>
<td>68.9</td>
<td>52.1</td>
<td>80.9</td>
<td>415.7</td>
</tr>
<tr>
<td>GCN</td>
<td>40.7</td>
<td>71.3</td>
<td>53.5</td>
<td>81.5</td>
<td>419.2</td>
</tr>
<tr>
<td>SelfAttn</td>
<td>40.9</td>
<td>71.3</td>
<td>53.8</td>
<td>82.2</td>
<td>420.1</td>
</tr>
<tr>
<td>AdaPool</td>
<td>40.2</td>
<td>71.8</td>
<td>52.8</td>
<td>81.4</td>
<td>418.3</td>
</tr>
<tr>
<td>SoftPool</td>
<td>39.2</td>
<td>69.8</td>
<td>52.8</td>
<td>81.4</td>
<td>419.8</td>
</tr>
<tr>
<td>GPO</td>
<td>41.2</td>
<td>71.1</td>
<td>55.6</td>
<td>82.0</td>
<td>422.9</td>
</tr>
<tr>
<td>Manual</td>
<td>38.7</td>
<td>71.6</td>
<td>53.7</td>
<td>82.1</td>
<td>419.3</td>
</tr>
</tbody>
</table>

Table 1: Comparison of complex aggregators and manually chosen pooling method evaluated with MS-COCO 5K. The best results are marked bold.

A.2 Slow Convergence of Hard Triplet Loss

Figure A.2-1 shows the comparison of VSE model with and without our proposed optimization objective, ADOPT. Note that hard triplet loss (Faghri et al., 2018) is used for optimization when ADOPT is not used. The optimization target ADOPT can fit the model more quickly, thus further improving the potential of the model, that is performance is better in the latter stage.

B Additional Implementation Details

Basic settings. The margin α for hard triplet loss is set to 0.2 and the τ used in InfoNCE is 0.05. The image and text features extracted from the encoders use L2 normalization. The common learning rate is set to 5e-4, while the learning rate of the pre-trained modules (like BERT, ResNet and ViT) is 10% of its.

Vision Encoders. When using region feature that is directly extracted from BUTD (Anderson et al., 2018), the multilayer perceptron is used to map the visual feature dimension into 1024 as the same as the text feature dimension. For the grid feature, the warm-up strategy is used for the first epoch. Then, all parameters are optimized in the rest of the 24 epochs. For the patch feature, the original image is changed to 224×224 resolution with the size of a patch as 16.

Text Encoders. When using BiGRU as the backbone, the token dimension is 300 and the hidden dimension is 1024. Only one layer is considered and the bidirectional features are averaged as the output feature. For BERT, the hidden dimension is 768 and the multilayer perceptron is also used to map the text feature dimension into 1024 as the same as the visual feature dimension.

C Additional Experiments and Results

We present Figure C-2 to show how the size of the negative samples changes during training with ADOPT. We can see that ADOPT starts with a large number of negative samples, but that decreases over time to only 4–5 samples at the end of the 5th epoch (yellow line). Different from the common hard triplet optimization that only considers one hardest negative sample, the max_violation strategy (Faghri et al., 2018) considers the rest of samples...
### Table 2: Different choices of parameters for fusing token-level and embedding-pooling.

<table>
<thead>
<tr>
<th>Weights</th>
<th>Image Retrieval</th>
<th>Caption Retrieval</th>
<th>RSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>emb</td>
<td>R@1</td>
<td>R@5</td>
</tr>
<tr>
<td>0.25</td>
<td>0.75</td>
<td>39.8</td>
<td>70.2</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>40.1</td>
<td>70.6</td>
</tr>
<tr>
<td>0.75</td>
<td>0.25</td>
<td>40.6</td>
<td>71.4</td>
</tr>
<tr>
<td>random</td>
<td>random</td>
<td>39.3</td>
<td>69.4</td>
</tr>
<tr>
<td>AdPool</td>
<td>AdPool</td>
<td>41.4</td>
<td>72.3</td>
</tr>
</tbody>
</table>

within the same mini-batch as negative samples only on the first epoch and the rest of epochs are the same as the hard triplet loss.

We last investigate the impact of the balance module in AdPool (Equation 5). Table C-2 shows that manually tuned weights underperform substantially compared to automatically learned weights\(^4\).

\(^4\)For “random”, we select the weights randomly and run them 5 times and compute the average to reduce variance.