SwAMP: Swapped Assignment of Multi-Modal Pairs for Cross-Modal Retrieval

Minyoung Kim
Samsung AI Center Cambridge, UK
mikim21@gmail.com

Abstract

We tackle the cross-modal retrieval problem, where learning is only supervised by relevant multi-modal pairs in the data. Although the contrastive learning is the most popular approach for this task, it makes potentially wrong assumption that the instances in different pairs are automatically irrelevant. To address the issue, we propose a novel loss function that is based on self-labeling of the unknown semantic classes. Specifically, we aim to predict class labels of the data instances in each modality, and assign those labels to the corresponding instances in the other modality (i.e., swapping the pseudo labels). With these swapped labels, we learn the data embedding for each modality using the supervised cross-entropy loss. This way, cross-modal instances from different pairs that are semantically related can be aligned to each other by the class predictor. We tested our approach on several real-world cross-modal retrieval problems, including text-based video retrieval, sketch-based image retrieval, and image-text retrieval. For all these tasks our method achieves significant performance improvement over the contrastive learning.

1 Introduction

Cross-modal retrieval, the task of retrieving the most relevant items in the database of one modality (e.g., images) for a given query from another modality (e.g., texts), has received unprecedented attention in computer vision and related areas (Chen et al., 2015; Faghri et al., 2018; Lee et al., 2018; Li et al., 2019; Zhang et al., 2020; Chun et al., 2021; Miech et al., 2021, 2019, 2020; Wang et al., 2021; Dey et al., 2019; Sain et al., 2021). The crux of the problem is to learn the underlying relevance or similarity metric between data instances that live in heterogeneous modalities with highly different distributions. Although there are several different learning problem formulations in the literature, in this paper we mainly focus on the *paired* training data setup, in which training is only supervised by relevant pairs in the training data, and there are no semantic class labels annotated. That is, the training data consist of only pairs of relevant multi-modal data instances, e.g., $(image, text)$, which may require minimal human annotation effort (e.g., web scraping of images and nearby texts).

The contrastive (or triplet loss) learning (Chopra et al., 2005; Hadsell et al., 2006) is recognised as the most popular and successful approach, which aims to learn the cross-modal similarity measure by the intuitive criteria that pull together relevant pairs and push away irrelevant ones. However, it makes potentially wrong assumption that instances in different pairs are automatically irrelevant. The pairs in the training data are usually collected by considering relevant pairs only (e.g., nearby images and texts in a web page), and the relevance of instances in different pairs is usually not checked However, this is implicitly assumed in the contrastive loss. The issue was also raised in recent work (Kim et al., 2019; Zhou et al., 2020; Patrick et al., 2020; Wray et al., 2021; Chen et al., 2021). In this paper we propose a novel learning algorithm that addresses the issue via self-labeled clustering approach.

Motivated from the recent clustering-based representation learning in the self-supervised learning literature (Asano et al., 2020; Caron et al., 2020), we propose a novel loss function for cross-modal retrieval that is based on self-labeling of the unknown classes. Specifically, we introduce (latent) semantic class labels to be assigned to data instances, where class labels supposedly decide the relevance of cross-modal data instances (i.e., the same class label means relevant items, and vice versa). We predict class labels of the data instances in each modality, and assign the predicted labels to the corresponding instances in the other modality (i.e., swapping the pseudo labels). With these swapped pseudo labels, we learn the data embedding for each modality using the supervised cross-entropy loss. This way, cross-modal instances from different pairs that

Proceedings of the 26$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

are semantically related can be aligned to each other by the class predictor. The whole process of label prediction and supervised learning with swapped classes is alternated to learn the optimal feature extraction networks. We call this approach *Swapped Assignment of Multi-modal Pairs* (SwAMP).

The main benefits of the SwAMP are in two folds: i) Unlike the contrastive loss, SwAMP does not make potentially wrong assumption that instances from different pairs are automatically irrelevant. The optimized class assignment finds similar instances from other pairs, and the feature extractor is trained in such a way that the same-class instances, even in different pairs, are well aligned. This feature of aligning instances in different pairs is hardly exploited in the contrastive loss. ii) Since the learning does not fully resort to pair-based losses as in contrastive learning, the sampling complexity can be reduced. This comes from the class-based loss adopted in the SwAMP, where similar ideas were exploited previously in self-supervised representation learning (Caron et al., 2018; Asano et al., 2020; Caron et al., 2020). Our approach is generically applicable to different types of cross-modal retrieval problems. We empirically demonstrate that the SwAMP loss improves retrieval performance significantly over the contrastive learning, on various real-world cross-modal retrieval problems, including text-video, sketch-image, and image-text retrieval.

2 Problem Setup & Background

Let x^A and x^B denote data instances from modality A and modality B, respectively. For instance, x^A is an image from the image modality, while x^B is a text/caption from the text modality. Throughout the paper we deal with *modality-wise feature representation*, meaning that we have modality-wise feature extractors (neural networks) $\phi^A(\cdot)$ and $\phi^B(\cdot)$ applied to x^A and x^B, respectively. Also known as *dual encoders*, it produces a succinct vector representation for each modality, $\phi^A(x^A) \in \mathbb{R}^d$ and $\phi^B(x^B) \in \mathbb{R}^d$. The shared feature space ($\subset \mathbb{R}^d$) allows us to define the similarity score $s(x^A, x^B)$ as a cosine angle between $\phi^A(x^A)$ and $\phi^B(x^B)$. The goal is to learn the feature extractors so that the relevant pairs x^A and x^B have a high similarity score $s(x^A, x^B)$, while irrelevant pairs have a low similarity score. The main benefit of the modality-wise feature representation is the computational efficiency, scalable to billions of instances at training/test time, thanks to the efficient dot-product. There is an alternative approach that directly computes the similarity score without having modality-wise representation. A typical example is the *cross-modal attention* models (Lee et al., 2018; Lu et al., 2019; Desai and Johnson, 2020; Huang et al., 2020) (details in Sec. 4). Although they can capture interactions between cross-modal local features, they are computationally demanding, not scalable to large-scale data.

The training data are composed of relevant pairs $\mathcal{D} = \{(x_i^A, x_i^B)\}_{i=1}^N$, where x_i^A and x_i^B are the instances in the i-th relevant pair. At test time, a query is given from the query modality, say x^A, and the goal is to find the most relevant instance, say x^B, from the other modality, where the search is performed on the given test set $\{x_i^B\}_{i=N+1}^{N+M}$.

2.1 Contrastive Learning

In contrastive learning (Chopra et al., 2005; Hadsell et al., 2006), it is implicitly assumed that data instances from different pairs are irrelevant, although it may not be true. The loss function is defined to capture the intuition: penalize low (high) similarity scores for relevant (irrelevant, resp.) pairs. By introducing the margin α (e.g., 0.2) and considering the most violating irrelevant pairs (i.e., hard negatives), the loss can be written as (subscript c stands for contrastive):

$$\mathcal{L}_c(\phi^A, \phi^B) = \sum_{i\in\mathcal{D}} \big(s(x_i^A, x_i^B) - \max_{j\in\mathcal{D}\setminus i} s(x_i^A, x_j^B)\big)_{\geq\alpha} + \big(s(x_i^A, x_i^B) - \max_{j\in\mathcal{D}\setminus i} s(x_j^A, x_i^B)\big)_{\geq\alpha} \quad (1)$$

where $(z)_{\geq\alpha} = \max(0, \alpha - z)$ only incurs positive loss when $z < \alpha$. A main issue of the contrastive learning is that we cannot guarantee that data instances from different pairs in the training data are irrelevant, because the data are usually collected by considering relevant pairs only (e.g., web scraping of images and nearby texts), and the relevance of instances in different pairs is usually not checked. However, this is assumed in the contrastive loss.

3 Our Approach: SwAMP

Our idea is to introduce *(latent) semantic class labels* for data instances and use them to learn the feature extractors. The class labels supposedly decide the relevance of data instances from different modalities, that is, x^A and x^B are considered relevant if their class labels are the same, and vice versa. Obviously, the paired cross-modal instances in the training data must have the same class labels. But beyond this, instances from different pairs can also be deemed relevant if they belong to the same semantic class labels. The motivation is that if we estimate the class labels accurately, the feature extractor learning can be turned into a supervised classification problem.

More formally, we consider (unknown) class labels to be assigned to the data instances. Let $y^A, y^B \in \{1, \ldots, K\}$ be the class labels for x^A and x^B, respectively, where K is chosen by the user. The relevance of x^A and x^B is determined by their class labels: x^A and x^B are deemed relevant if $y^A = y^B$ and irrelevant if $y^A \neq y^B$. If we knew the class labels that bear such semantics in the training data, then training becomes supervised learning that can be done for each modality, which allows us to avoid pairwise terms in the loss function. However, we don't have class labels, and we optimize them (i.e., self-supervised learning) together with the feature extractors $\phi^A(\cdot)$ and $\phi^B(\cdot)$. To this end, we build linear classifiers $p(y|x^A)$ and $p(y|x^B)$ on the extracted features. For each modality $M \in \{A, B\}$,

$$p(y=j|x^M) = \frac{\exp(p_j^\top \phi^M(x)/\tau)}{\sum_l \exp(p_l^\top \phi^M(x)/\tau)}, \tag{2}$$

where $P = \{p_1, \dots, p_K\}$ are trainable parameters that are shared between two modalities, and τ is the temperature in the softmax. We can regard each p_j as the *prototype vector* for class j that lies in the shared feature space. Since we have classification models, the (supervised) cross-entropy loss minimization is a natural choice to optimize them. That is, letting $p_{true}(y|x^A)$ be the true conditional class distribution for modality A, we minimize $\mathbb{E}_{p_{true}(y|x^A)}[-\log p(y|x^A)]$ with respect to P and the network parameters of $\phi^A(\cdot)$ (similarly for modality B). Since we cannot access $p_{true}(y|x^A)$, one may be tempted to use the model $p(y|x^A)$ in (7) instead. However, it can easily lead to a degenerate solution such as the one that puts all the probability mass on a particular single class all the time (thus attaining the optimal cross-entropy loss 0). Moreover, this would make learning $\phi^A(\cdot)$ and $\phi^B(\cdot)$ nearly independent and less interacted with each other, merely through the shared prototypes P.

Instead, we form an optimization problem to estimate a surrogate of $p_{true}(y|x^A)$, which we denote by $q(y|x^A)$, using the information from the other modality B, while imposing additional constraints to avoid the degenerate solutions. More specifically, we optimize the surrogate $q(y|x^A)$ with the following two criteria. First, $q(y|x^A)$ needs to be well aligned with the current estimate $p(y|x^B)$ for x^B that is paired with x^A. This is due to the aforementioned requirements for the class labels, where the class labels (more generally, their distributions) of the paired instances should match. Secondly, the marginal distribution $q(y) = \mathbb{E}_{x^A \sim \mathcal{D}}[q(y|x^A)]$ is constrained to be a *uniform* distribution[1]. This constraint naturally arises from the symmetry of class labels, a reasonable assumption about the true class distribution, and successfully leaves out the degenerate solutions discussed above. To summarize, the following is the optimization problem for $q(y|x^A)$, where Q^A is the $(N \times K)$ matrix with $Q^A_{iy} := q(y|x^A_i)$. Recall that $\mathcal{D} = \{(x^A_i, x^B_i)\}_{i=1}^N$ is the training data of paired instances.

$$\min_{Q^A} \ \mathbb{E}_{i\sim\mathcal{D}}\big[\mathbb{E}_{q(y|x^A_i)}[-\log p(y|x^B_i)]\big] \tag{3}$$
$$\text{s.t. } \ \mathbb{E}_{i\sim\mathcal{D}}[q(y|x^A_i)] = 1/K, \ \forall y.$$

We perform similar optimization for $q(y|x^B)$ ($Q^B_{iy} := q(y|x^B_i)$) to approximate $p_{true}(y|x^B)$ by exchanging the roles of A and B. The optimal solutions (surrogates) are denoted by q^A and q^B, where we use the superscript to distinguish the two modalities. Note that during the optimization of (3) for q^A and q^B, we fix the model parameters, that is, P and the feature extractor networks. The overall optimization is *alternation* between: i) surrogate optimization (3) with P, ϕ^A, ϕ^B fixed, and ii) supervised (cross-entropy) loss minimization with q^A and q^B fixed, where the latter can be written as (subscript s stands for SwAMP):

$$\min_{P,\phi^A,\phi^B} \mathcal{L}_s := \mathbb{E}_{i\sim\mathcal{D}}\big[\mathbb{E}_{q^A(y|x^A_i)}[-\log p(y|x^A_i)]\big] + \mathbb{E}_{i\sim\mathcal{D}}\big[\mathbb{E}_{q^B(y|x^B_i)}[-\log p(y|x^B_i)]\big] \tag{4}$$

Now we discuss how to optimize (3). It is essentially the optimal transport (OT) problem (Villani, 2008; Cuturi, 2013), specifically with the cost matrix $C_{iy} = -\log p(y|x^B_i)$ and the marginal constraints $\sum_i Q^A_{iy} = 1/K, \forall y$ (and implicitly $\sum_y Q^A_{iy} = 1/N, \forall i \in \mathcal{D}$). Although the OT is known to be an instance of the linear program (LP), conventional LP solvers are not suitable for large-scale problems. As is common practice, we relax the problem by augmenting the loss with the entropic regularizer for $q(y|x^A)$, namely $\frac{1}{\eta}\sum_{iy} Q^A_{iy} \log Q^A_{iy}$ added to the loss (thus, penalizing small entropy), which can be solved by the efficient Sinkhorn-Knopp (SK) algorithm (Cuturi, 2013). Here η is the regularization trade-off hyperparameter. The SK algorithm finds the optimal solution as $Q^A = \text{Diag}(u) A \text{Diag}(v)$, where $A_{iy} = e^{-\eta C_{iy}}$ and the vectors $u \in \mathbb{R}^N_+$ and $v \in \mathbb{R}^K_+$ are the fixed points of $u_i = \frac{1}{N}/(Av)_i$, $v_j = \frac{1}{K}/(A^\top u)_j$ for $i = 1, \dots, N$, $j = 1, \dots, K$. The fixed point iteration usually converges quickly after a few iterations. We denote the algorithm as:

$$Q \leftarrow \text{SK}(\text{cost} = C, \text{reg} = \eta). \tag{5}$$

One challenge in optimizing (3) with the SK, however, is that it involves the entire dataset $\mathcal{D}$ in the loss, which means that the model update (4) has to be deferred until q is optimized for an entire data epoch. Simply replacing $\mathcal{D}$ with a minibatch might be dangerous since the population class marginal distributions are poorly covered by a minibatch. We need an even larger subset of $\mathcal{D}$ to roughly meet the (uniform) class constraint. To this end, we adopt the (FIFO) queues, where we accumulate the embeddings $\phi^A(x^A)$ and $\phi^B(x^B)$ from the latest minibatches into the queues. The optimization (3) is then performed on the queue data ($\mathcal{D}$ replaced by the data in the queues). To have the uniform class constraint meaningful, we choose the queue size to be greater than K. Note that (3) is solved by the SK algorithm, and thus no backprop is required, hence enlarging the queue size does not incur computational issue. Similar ideas were used in the self-supervised representation learning literature, e.g., (He et al., 2019) and (Caron et al., 2020). To have the queues filled with the latest features, we insert the features

[1]This means balanced clusters. Even when data exhibit imbalance in semantic classes (e.g., long-tail distributions), our clustering model can still handle it by learning semantically redundant multiple clusters, thus forming *super*-clusters while rendering others minor classes. See Sec. A for illustration.

Algorithm 1 SwAMP Training.

Input: Class cardinality K, queue size, temp. τ, η in SK.
Initialize: $P = \{p_k\}_{k=1}^{K}$, ϕ^A, ϕ^B. Empty queue $\mathcal{Q}$.
Output: Trained model $\{P, \phi^A(\cdot), \phi^B(\cdot)\}$.
Repeat until convergence:
1. Sample a minibatch of paired data $\mathcal{B} = \{(x_i^A, x_i^B)\}$.
2. Evaluate $\phi^A(x_i^A)$ and $\phi^B(x_i^B)$ for $i \in \mathcal{B}$ (forward pass).
3. Insert $\{(\phi^A(x_i^A), \phi^B(x_i^B))\}_{i\in\mathcal{B}}$ into the queue $\mathcal{Q}$.
4. Solve (3) for modality A and B:
$\{q^A(y|i)\}_{i\in\mathcal{Q}} \leftarrow$ SK(cost=$\{-\log p(y|x_i^B)\}_{i\in\mathcal{Q}}$, reg= η)
$\{q^B(y|i)\}_{i\in\mathcal{Q}} \leftarrow$ SK(cost=$\{-\log p(y|x_i^A)\}_{i\in\mathcal{Q}}$, reg= η)
5. Take the minibatch portions $\{q^A(y|i), q^B(y|i)\}_{i\in\mathcal{B}}$;
Do SGD update with $\mathcal{L}$ in (6).

of the current minibatch into the queues, then perform the SK algorithm. Once (3) is done, we can optimize (4) by gradient descent, but only the current minibatch portion of q is used. The final loss function is a combination of the SwAMP loss and the contrastive loss,

$$\mathcal{L}(P, \phi^A, \phi^B) = \mathcal{L}_c(\phi^A, \phi^B) + \lambda \mathcal{L}_s(P, \phi^A, \phi^B), \quad (6)$$

where λ is the trade-off hyperparameter.

As we estimate the surrogate q^A using the current classification model in modality B, and vice versa, the class assignment is *swapped*. The pseudo code of our algorithm is shown in Alg. 1. The idea of optimizing class labels in the representation learning was previously introduced in (Asano et al., 2020; Caron et al., 2020), however, they aimed for self-supervised representation learning as an instance discrimination pretext task with augmented data. In this paper, we deal with the *cross-modal retrieval* problem, where we estimate the class labels of instances in one modality using the features from the other modality. Unlike the contrastive loss, SwAMP does not make any assumption that instances from different pairs are automatically irrelevant. The OT class assignment finds similar instances from other pairs, and the feature extractor is trained in such a way that the same-class instances, even in different pairs, are well aligned. This feature of aligning instances in different pairs is hardly exploited in the contrastive loss.

4 Related Work

Cross-modal retrieval. It is beyond the scope of the paper to enumerate all previous works on cross-modal retrieval, and we refer the readers to recent survey papers such as (Wang et al., 2016). Recently, the most interesting cross-modal tasks involve, among others, video-text (Liu et al., 2019; Gabeur et al., 2020; Patrick et al., 2020; Miech et al., 2021, 2019, 2020; Wang et al., 2021; Chen et al., 2021), image-text (Chen et al., 2015; Faghri et al., 2018; Lee et al., 2018; Chun et al., 2021; Li et al., 2019; Zhang et al., 2020), and sketch-photo (Dey et al., 2019; Sain et al., 2021). For the training data of relevant pairs, most approaches commonly rely on the idea of contrastive learning (Chopra et al., 2005; Hadsell et al., 2006). Beyond the intuitive triplet forms (Wang et al., 2014; Schroff et al., 2015), more sophisticated losses were introduced in (Sohn, 2016; Song et al., 2016; Wang et al., 2019a,b) to deal with a positive and multiple negative pairs as well as hard examples. To reduce the super-linear time computational overhead, several sophisticated sampling strategies were proposed (Wu et al., 2017; Harwood et al., 2017; Yuan et al., 2017). As discussed in Sec. 2, there are broadly two different ways to define the similarity metric between instances of different modalities: modality-wise feature representation and cross-modal attention. The main benefit of the former is the computational efficiency, scalable to billions of instances at training/test time, thanks to the efficient dot-product. The latter directly computes the similarity score without having modality-wise representation (Lee et al., 2018; Lu et al., 2019; Desai and Johnson, 2020; Huang et al., 2020) using the transformer-like attentive neural networks which aim to capture interactions between local features in the instances from different modalities. Although they can capture cross-modal interactions between local features of data instances from different modalities, they are computationally demanding and very slow due to the quadratic complexity in the number of local features. In (Miech et al., 2021), a hybrid of the two is introduced, which retains the two models, but performs re-ranking/distillation at test time for speed-up.

Clustering-based approaches. There were previous attempts to cluster (group) data instances, or equivalently self-labeling, to improve saliency in representation learning. Some approaches perform offline K-means clustering for every epoch (Caron et al., 2018; Alwassel et al., 2020), which can make training slow. The idea of optimizing class labels in the representation learning was previously introduced in (Asano et al., 2020; Caron et al., 2020). However, all these previous approaches aimed for self-supervised representation learning as an instance discrimination pretext task with augmented data. On the other hand, we perform simultaneous learning of class labels and the feature extraction networks for the cross-modal retrieval setting. More recently (Chen et al., 2021) proposed a clustering-based cross-modal retrieval method based on the semantic similarity. However, our approach is mainly different from it in that we adopt the OT-based class label assignment forming a joint feature-label optimization, instead of simple fusion of multi-modal features for clustering as in (Chen et al., 2021).

5 Experimental Results

We test the proposed SwAMP loss on several different types of real-world cross-modal retrieval problems. For each problem/dataset, we choose the most popular and successful method in the literature, and replace its loss function (mostly contrastive loss) with the proposed SwAMP loss to demonstrate the performance improvement. To this end, for fair comparison, we faithfully follow the same optimization

Table 1: Text-video retrieval results on YouCook2.

Methods	R@1 ↑	R@5 ↑	R@10 ↑	Med-R ↓
Random	0.03	0.15	0.3	1675
FV-CCA	4.6	14.3	21.6	75
Contrastive (No PT)	4.2	13.7	21.5	65
SwAMP (No PT)	**4.8**	**14.5**	**22.5**	**57**
Contrastive (PT)	8.2	24.5	35.3	24
SwAMP (PT)	**9.4**	**24.9**	35.3	**22**

strategy and hyperparameters as the baseline methods.

5.1 Text-based Video Retrieval

We first consider the text-to-video retrieval task where the goal is to find the most relevant video clip for a given natural language text query. We consider three datasets for this task: i) **YouCook2** (Zhou et al., 2018) of cooking videos and instructions, ii) **MSR-VTT** (Xu et al., 2016) of generic videos and captions from YouTube, and iii) **LSMDC** (Rohrbach et al., 2017) of movie clips and subtitles. All these datasets provide pairs of video clip and text description, forming a multi-modal paired data format $(text, video)$ which conforms to our SwAMP framework.

For the raw text/video features and the feature extractor networks, as well as the training/test protocols, we follow the methods in (Miech et al., 2019), and the details are described in Appendix (Sec. C). Following (Miech et al., 2019), there are two training strategies: i) No-pretraining (No-PT) where the feature extraction networks are randomly initialized, and the training is done on the training split of the dataset, and ii) Pretraining (PT) where the feature extractors are first pretrained on the large-scale HowTo100M dataset (Miech et al., 2019), and finetuned on the target dataset. In (Miech et al., 2019), they adopt the contrastive (triplet) loss for training the feature extractors. Although we also compare our approach with the state-of-the-arts, the main focus in this experiment is to demonstrate the performance improvement achieved by the proposed SwAMP loss against vanilla contrastive learning. The SwAMP hyperparameter λ, the weight/impact of the SwAMP loss against the contrastive loss in (6) is chosen as $\lambda = 0.25$ for all three datasets, except the LSMDC-PT case for which $\lambda = 0.1$. We also choose temperature in softmax $\tau = 0.25$, entropic regularization trade-off in SK $\eta = 5.0$, the number of classes $K = 500$, and the queue size $2,048$ for the SwAMP. The other learning hyperparameters common in SwAMP and contrastive losses are not changed from (Miech et al., 2019), and summarized in Appendix (Sec. C).

YouCook2. This cooking video dataset collected from YouTube, contains 89 recipes and 14K video clips annotated with textual descriptions from paid human workers. The test data are formed by taking 3.5K clips from the validation set, and the test set comprises of $3,350$ pairs. The retrieval performance metrics are recall-at-k (R@k) with $k = 1, 5, 10$ and the median rank (Med-R). Hence, the random guess attains R@1= 0.03% Med-R=1, 675. The results are summarized in Table 7. In the bottom four rows, we see the performance improvement achieved by the proposed SwAMP against the contrastive loss (Miech et al., 2019). For both training strategies, No PT (random model initialization) and PT (initialized with the HowTo100M-pretrained model), our SwAMP improves the retrieval performance significantly (e.g., about 12% reduction in Median Rank for the No PT case). SwAMP also outperform the CCA baseline FV-CCA (Klein et al., 2015).

MSRVTT. This dataset (Xu et al., 2016) collected from YouTube contains videos of specific categories including music, sports, and movie. There are 200K video-caption pairs obtained by human annotation. We follow the retrieval training/test protocol of (Yu et al., 2018; Miech et al., 2019). The test set consists of 1K pairs. As reported in Table 2, our SwAMP loss improves the performance over the contrastive learning significantly for both no-pretraining and pretraining cases: about 24% in R@1 in the No PT case, and 27% in the PT case. Furthermore, the SwAMP outperforms with large margin the state-of-the-arts: C+LSTM+SA+FC7 (Torabi et al., 2016), VSE-LSTM (Kiros et al., 2014), Temporal Tessellation (Kauman et al., 2017), CT-SAN (Yu et al., 2017), and JSFusion (Yu et al., 2018).

LSMDC. This dataset of movie video clips is comprised of 101K video-caption pairs. The captions are collected either from the movie scripts or the audio descriptions. The test set contains 1K pairs. For this dataset, $\lambda = 0.1$ (impact of the SwAMP loss against contrastive) for the PT case. The results are shown in Table 2. Similar to the other two datasets, our SwAMP is consistently better than the contrastive learning (about $7 \sim 9\%$ in Median Rank).

5.2 Sketch-based Image Retrieval

In sketch-based image retrieval, the model takes a user's sketch (quick drawing) of an object as input query, and retrieves the photo images that correspond to the same object category as query's. We follow the recent framework of (Dey et al., 2019) which reports the state-of-the-art performance on the three large-scale sketch-image benchmarks: Sketchy-Extended (Sangkloy et al., 2016), TU-Berlin-Extended (Eitz et al., 2012), and QuickDraw-Extended (Dey et al., 2019). The datasets roughly consist of 100–200 object classes with hundreds to thousands of sketch/photo images for each class. For all these datasets, we have *zero-shot* setting, meaning that training/test splits have instances from disjoint object categories.

In this experiment we aim to show the improvement in the retrieval performance when our SwAMP loss is augmented to the existing loss function. To this end, we follow the same embedding networks for images and sketches, as well as the same loss function as the Doodle2Search. The loss func-

Table 2: Text-Video retrieval results on MSRVTT and LSMDC.

Methods	MSRVTT				LSMDC			
	R@1 ↑	R@5 ↑	R@10 ↑	Med-R ↓	R@1 ↑	R@5 ↑	R@10 ↑	Med-R ↓
Random	0.1	0.5	1.0	500	0.1	0.5	1.0	500
C+LSTM+SA+FC7	4.2	12.9	19.9	55	4.3	12.6	18.9	98
VSE-LSTM	3.8	12.7	17.1	66	3.1	10.4	16.5	79
SNUVL	3.5	15.9	23.8	44	3.6	14.7	23.9	50
Temporal Tessellation	4.7	16.6	24.1	41	4.7	15.9	23.4	64
CT-SAN	4.4	16.6	22.3	35	4.5	14.1	20.9	67
JSFusion	10.2	31.2	43.2	13	9.1	21.2	34.1	36
Contrastive (No PT)	12.1	35.0	48.0	12	7.2	18.3	25.0	44
SwAMP (No PT)	**15.0**	**38.5**	**50.3**	**10**	**7.7**	**19.3**	**27.7**	**40**
Contrastive (PT)	14.9	40.2	52.8	9	7.1	19.6	27.9	40
SwAMP (PT)	**19.0**	**42.4**	**55.2**	**8**	**8.3**	**20.0**	**28.9**	**37**

Table 3: Sketch-based image retrieval results. The contrastive-learning-based Doodle2Search Dey et al. (2019) (denoted by D2S) is compared with the proposed SwAMP learning.

Methods / Datasets	Sketchy			TU-Berlin			QuickDraw		
	mAP	mAP@200	P@200	mAP	mAP@200	P@200	mAP	mAP@200	P@200
ZSIH Shen et al. (2018)	25.40	-	-	22.00	-	-	-	-	-
CVAE Yelamarthi et al. (2018)	19.59	22.50	33.30	0.50	0.90	0.30	0.30	0.60	0.30
D2S Dey et al. (2019)	36.91	46.06	37.04	10.94	15.68	12.08	7.52	9.01	6.75
SwAMP	**40.32**	**51.94**	**40.81**	**17.63**	**24.49**	**19.75**	**8.19**	**11.62**	**9.10**

tion consists of three losses: *Triplet loss* is the conventional triplet loss, *Domain loss* uses an adversarial domain classifier to penalize misalignment between embedding distributions of photo images and sketches, and *Semantic loss* urges the embeddings of the photo images and sketches to reconstruct the pretrained word embedding of the corresponding object word. We also use the same attention-based embedding networks for photo and sketch modalities. Then, we add our SwAMP loss to the Doodle2Search's loss with the impact $\lambda = 0.1$ for all three datasets. We use the queue size 1000 (2000 for QuickDraw-Extended) and class cardinality $K = 500$, softmax temperature $\tau = 0.25$, entropic regularization impact $\eta = 5.0$. The retrieval performances on the three datasets are summarized in Table 3. The performance metrics are mean average precision (mAP), mAP@200, and the precision-at-200 (P@200). As shown, our SwAMP loss when added to the existing contrastive-based loss, significantly improves the retrieval performance (about 9% in mAP for Sketchy and about 60% for TU-Berlin).

5.3 Image-Text Retrieval

For the image-text cross-modal retrieval task, we follow the features and protocols from the well-known *stacked cross attention network* (SCAN) (Lee et al., 2018). In their framework, each image is represented by a set of local features $V = \{v_1, \dots, v_k\}$, where v_i $(\in \mathbb{R}^D) = W_v f_i + b_v$ and f_i's are the CNN features extracted from salient image regions detected by the Faster-R-CNN model (Ren et al., 2015). The raw features f_i's are fixed and $\{W_v, b_v\}$ are learnable parameters. The text (sentence) is also treated as a set of word features $E = \{e_1, \dots, e_n\}$, where e_i $(\in \mathbb{R}^D) = (h_i^{lr} + h_i^{rl})/2$ and $h_i^{lr/rl}$ are the outputs of the bi-directional GRU (Bahdanau et al., 2015; Schuster and Paliwal, 1997) with the sequence of word embeddings as input. Both the word embeddings and GRU parameters are learnable. These image/text features contain rich local information, however, one challenge is that both representations are *sets*, hence the number of elements (k and n) can vary from instance to instance.

In (Lee et al., 2018), they proposed a cross-modal attention model, where each local feature from one modality is transformed by the attention (Vaswani et al., 2017) with the set of local features in the other modality; e.g., v_i is transformed to $attn(v_i; \{e_j\}_{j=1}^n)$ = the weighted sum of *values* $\{e_j\}_{j=1}^n$ with v_i as a *query* and $\{e_j\}_{j=1}^n$ as *keys* (this denoted by i-t, while the other attention direction t-i can be used alternatively). Then the similarity score between image V and text E is defined as $pool(\{cos(v_i, attn(v_i; \{e_j\}_{j=1}^n))\}_{i=1}^K)$, where $cos(a, b)$ is the cosine similarity and $pool$ is the pooling operation, either of AVG (average) or LSE (log-sum-exp). Then the triplet contrastive loss of (1) is employed. Although the cross-attention is useful for capturing interaction between local features, computing the similarity score takes quadratic time in the number of local features in the instances. This is time consuming compared to the simple dot-product of the modality-wise embedding vectors (See Table 12 for wall-clock times).

To have modality-wise succinct representation instead (for SwAMP), we adopt the *induced-set attention* idea from Set-Transformer (Lee et al., 2019). Specifically, we introduce p learnable prototype (query) vectors $\{q_j\}_{j=1}^p$, $q_j \in \mathbb{R}^D$. Then we compute the attention for each query with V (or E), i.e., $z_j = attn(q_j; \{v_i\}_{i=1}^k)$. We define $\phi^{image}(V) = concat(z_1, \dots, z_p)$, similarly for $\phi^{text}(E)$,

Table 4: Image-text retrieval results on Flickr30K.

Methods	Image → Text			Text → Image		
	R@1	R@5	R@10	R@1	R@5	R@10
DAN Nam et al. (2017)	55.0	81.8	89.0	39.4	69.2	79.1
DPC Zheng et al. (2017)	55.6	81.9	89.5	39.1	69.2	80.9
VSE++ Faghri et al. (2018)	52.9	-	87.2	39.6	-	79.5
SCO Huang et al. (2018)	55.5	82.0	89.3	41.1	70.5	80.1
SCAN i-t AVG	67.9	89.0	94.4	43.9	74.2	82.8
SCAN t-i AVG	61.8	87.5	93.7	45.8	74.4	83.0
SCAN t-i AVG + i-t LSE	67.4	90.3	95.8	48.6	77.7	85.2
Contrastive-PAR	65.7	86.8	92.4	48.2	75.8	84.2
SwAMP-PAR	**67.8**	**88.5**	**94.0**	**49.1**	**76.1**	**83.7**

Table 5: Image-text retrieval results on MS-COCO.

Methods	5-fold (1K test images)						Entire (5K test images)					
	Image → Text			Text → Image			Image → Text			Text → Image		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
DPC Zheng et al. (2017)	65.6	89.8	95.5	47.1	79.9	90.0	41.2	70.5	81.1	25.3	53.4	66.4
VSE++ Faghri et al. (2018)	64.6	-	95.7	52.0	-	92.0	41.3	-	81.2	30.3	-	72.4
GXN Gu et al. (2018)	68.5	-	97.9	56.6	-	94.5	42.0	-	84.7	31.7	-	74.6
SCO Huang et al. (2018)	69.9	92.9	97.5	56.7	87.5	94.8	42.8	72.3	83.0	33.1	62.9	75.5
PCME Chun et al. (2021)	68.8	-	-	54.6	-	-	44.2	-	-	31.9	-	-
SCAN i-t	69.2	93.2	97.5	54.4	86.0	93.6	46.4	77.4	87.2	34.4	63.7	75.7
SCAN t-i + i-t	72.7	94.8	98.4	58.8	88.4	94.8	50.4	82.2	90.0	38.6	69.3	80.4
Contrastive-PAR	71.8	94.3	97.9	56.8	86.9	93.8	48.4	78.1	88.1	34.3	64.4	76.2
SwAMP-PAR	**72.6**	**94.6**	**98.0**	**57.4**	**87.6**	**94.1**	**49.7**	**79.1**	**88.3**	**35.0**	**65.1**	**76.6**

where *concat* refers to concatenation. We share the same $\{q_j\}_{j=1}^p$ for both modalities. We also have multi-head extension. We call these modality-wise features as *prototype attention representation* (PAR). Note that computing PAR features has linear complexity in the number of local features (p assumed constant), and the cross-modal similarity is simply dot-product of PAR features, and can be computed in linear time (See also Table 12 for comparison with SCAN's cross-modal attention).

We test our approach on the popular image-text retrieval datasets, MS-COCO and Flickr30K. The details of the datasets and training/test protocols are described in Appendix (Sec. D). The results are summarized in Table 10 and Table 11. We specifically highlight the comparison between the contrastive loss and our SwAMP loss with the modality-wise feature representation (Contrastive-PAR vs. SwAMP-PAR). For the PAR features, we choose the number of prototypes $p = 20$, attention weight temperature $T = 0.5$, and the number of heads $H = 1$ for Flickr, and $p = 10, T = 0.5, H = 2$ for MS-COCO. For the SwAMP hyperparameters, we use the impact of SwAMP loss $\lambda = 1.0$, softmax temperature $\tau = 0.025$, the number of classes $K = 1,000$, queue size $1,280$ for both datasets. SwAMP performs consistently better than the contrastive loss and outperforms several state-of-the-arts including the recent sophisticated probabilistic embedding strategy (PCME) (Chun et al., 2021).

When compared with the computationally expensive SCAN, SwAMP mostly outperforms SCAN except for the SCAN's best attention direction/combination choices. To see the computational advantage of SwAMP-PAR, we compare the actual training/test times for the two approaches in Table 12, measured on the same machine with a single GPU (RTX 2080 Ti) and Core i7 3.50GHz CPU. Our SwAMP-PAR is about 4 times faster than SCAN for training on both datasets, while the difference becomes even more pronounced during test; SwAMP-PAR is about two orders of magnitude faster than the cross-modal attention model.

Table 6: Running times (seconds) measured on (Core i7 3.50GHz CPU / 128GB RAM / 1 RTX-2080Ti GPU). Per-batch times for training, entire times for test. For MS-COCO test, times for 5K test images (1K test in parentheses).

Methods	Flickr30K		MS-COCO	
	Train	Test	Train	Test
SCAN i-t AVG	0.35	336.9	0.33	9352.0 (350.3)
SwAMP-PAR	0.09	3.8	0.08	25.9 (16.3)

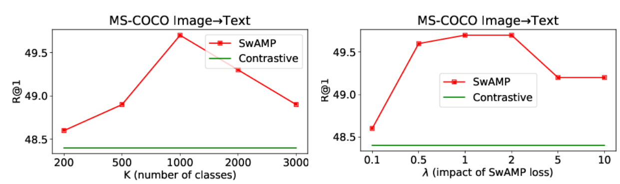


Figure 1: Impact of K (the number of classes) and λ.

5.4 Ablation Study

We perform empirical study on the impact of two important hyperparameters in our model: the number of classes K and SwAMP loss trade-off λ.

Number of classes (K). Recall that the best K values we chose were: $K = 1000$ for the image-text retrieval datasets and $K = 500$ for text-based video retrieval. To see how the retrieval performance is affected by other choices of K, we conduct experiments by varying K around the optimal values. The results on MS-COCO ($I \to T$) and YouCook2 tasks are shown in Fig. 16 (Left). (More results on other datasets can be found in Appendix (Fig. 4–8, Sec. B).) Clearly, very small K has low retrieval performance (R@1), and increasing K leads to improvement. However, beyond certain points, there is no benefit of increasing K and we even see performance degradation, which agrees with the observations from previous work (Asano et al., 2020; Caron et al., 2020). This is perhaps due to the difficulty of assigning meaningful cluster labels in optimal transport. Overall, with properly chosen K, SwAMP outperforms contrastive learning, signifying that SwAMP's grouping/clustering of similar instances is more effective than vanilla instance discrimination. The fact that the optimal K values are different in two tasks (image-text and video-text) implies that the best cardinality of semantic clusters is highly dependent on the dataset characteristics (e.g., size and semantic diversity).

SwAMP impact (λ). The sensitivity to λ is shown in Fig. 16 (Right), and more results and further discussions are in Appendix (Fig. 9–13, Sec. B).

5.5 Visualization of Learned Clusters

As qualitative analysis, we visualize the learned clusters to see if they capture meaningful semantic information. On MS-COCO (trained with the number of classes $K = 1000$), we organize images and texts by their assigned cluster labels using the learned prototype classification model (i.e., (7)). We first visually inspect individual clusters, images and texts that belong to each cluster. As we show a few examples in Fig. 2 (more in Appendix (Fig. 2,3, Sec. A)), each cluster contains semantically coherent data samples. Then we inspect texts (captions) in each cluster, and select a few keywords, those words that appear the most frequently in the texts. These keywords for each cluster consist of objects (noun) and/or actions (verb) that faithfully describe the cluster and data samples that belong to it. The full list is shown in Appendix (Fig. 1, Sec. A), but to enumerate a few of them (*cluster ID: keywords*), for instance, *0014: giraffe/feeding*, *0169: soccer/playing*, *0283: bus/parked*, *0405: pizza/oven*, *0597: vase/flowers*, *0713: dog/ball*, *0818: kite/flying*, *0956: parking/meter*.

Although the last three clusters in Fig. 2 all have the semantic meaning of *baseball*, they have different details in either activity or focus/scene: *swing*, *base playing*, and *crowd scene*. This means that SwAMP finds clusters based on the whole contents (objects, acitivities, and scenes), instead of doing merely object-based clustering. Although we have roughly equal numbers of samples per cluster, we found that some clusters are overlapped with others in terms of semantic meaning (redundant clusters in Appendix (Fig. 1,

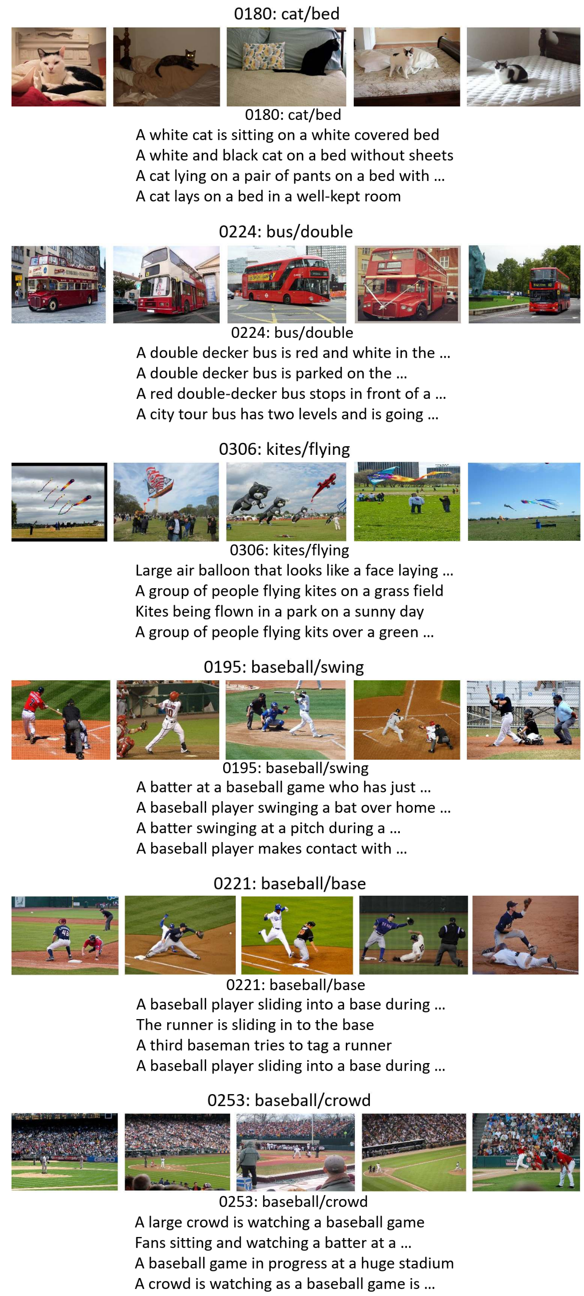


Figure 2: Some randomly selected clusters with images and texts that belong to them. Each cluster, titled by *ID: keywords*, shows randomly chosen 5 images and 4 texts.

Sec. A)), constituting larger *super*-clusters. These clusters are related to dominant data samples (e.g., cat, dog, tennis, baseball). This implies that the SwAMP can effectively deal with imbalance of semantic classes that can reside in data.

6 Conclusion

We have proposed a novel clustering-based loss function for cross-modal retrieval. The swapped class assignment over the modalities enables improved feature alignment with increased flexibility, while discovering meaningful latent semantic classes. The efficacy of our approach was demonstrated on several real-world cross-modal retrieval problems in diverse modalities, text-video, sketch-photo, and image-text, where our method achieved significant performance improvement over the contrastive learning for all these tasks.

References

Humam Alwassel, Dhruv Mahajan, Bruno Korbar, Lorenzo Torresani, Bernard Ghanem, and Du Tran. Self-supervised learning by cross-modal audio-video clustering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yuki M. Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations (ICLR)*, 2020.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate, 2015. International Conference on Learning Representations (ICLR).

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision (ECCV)*, 2018.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, 2020.

J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Brian Chen, Andrew Rouditchenko, Kevin Duarte, Hilde Kuehne, Samuel Thomas, Angie Boggust, Rameswar Panda, Brian Kingsbury, Rogerio Feris, David Harwath, James Glass, Michael Picheny, and Shih-Fu Chang. Multimodal Clustering Networks for Self-supervised Learning from Unlabeled Videos, 2021. International Conference on Computer Vision.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick. Microsoft COCO Captions: Data Collection and Evaluation Server. *arXiv preprint arXiv:1504.00325*, 2015.

S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

Sanghyuk Chun, Seong Joon Oh, Rafael Sampaio de Rezende, Yannis Kalantidis, and Diane Larlus. Probabilistic embeddings for cross-modal retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8415–8424, June 2021.

Marco Cuturi. Sinkhorn Distances: Lightspeed Computation of Optimal Transportation Distances. In *Advances in Neural Information Processing Systems*, 2013.

Karan Desai and Justin Johnson. VirTex: Learning Visual Representations from Textual Annotations. *arXiv preprint arXiv:2006.06666*, 2020.

Sounak Dey, Pau Riba, Anjan Dutta, Josep Llados, and Yi-Zhe Song. Doodle to Search: Practical Zero-Shot Sketch-based Image Retrieval. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM Transactions on Graphics*, 31(4): 1–10, 2012.

F. Faghri, D.J. Fleet, J.R. Kiros, and S. Fidler. VSE++: Improved visual-semantic embeddings with hard negatives. *In Proc. of British Machine Vision Conference*, 2018.

V. Gabeur, C. Sun, K. Alahari, and C. Schmid. Multi-modal transformer for video retrieval, 2020. European Conference on Learning Representations.

J. Gu, J. Cai, S. Joty, L. Niu, and G. Wang. Look, imagine and match: Improving textual-visual cross-modal retrieval with generative models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

K. Hara, H. Kataoka, and Y. Satoh. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet? In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Ben Harwood, Vijay Kumar B G, Gustavo Carneiro, Ian Reid, and Tom Drummond. Smart mining for deep metric learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

Y. Huang, Q. Wu, and L. Wang. Learning semantic concepts and order for image and sentence matching. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Zhicheng Huang, Zhaoyang Zeng, Bei Liu, Dongmei Fu, and Jianlong Fu. Pixel-BERT: Aligning image pixels with text by deep multi-modal transformers. *arXiv preprint arXiv:2004.00849*, 2020.

A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

D. Kauman, G. Levi, T. Hassner, and L. Wolf. Temporal tessellation: A unified approach for video analysis. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

S. Kim, M. Seo, I. Laptev, M. Cho, and S. Kwak. Deep metric learning beyond binary supervision, 2019. Computer Vision and Pattern Recognition.

D. P. Kingma and L. J. Ba. Adam: A Method for Stochastic Optimization, 2015. International Conference on Learning Representations (ICLR).

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014.

B. Klein, G. Lev, G. Sadeh, and L. Wolf. Associating neural word embeddings with deep image representations using Fisher vectors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3744–3753, 2019.

Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. Stacked cross attention for image-text matching. In *European Conference on Computer Vision (ECCV)*, 2018.

Kunpeng Li, Yulun Zhang, Kai Li, Yuanyuan Li, and Yun Fu. Visual semantic reasoning for image-text matching. In *International Conference on Computer Vision (ICCV)*, 2019.

Y. Liu, S. Albanie, A. Nagrani, and A. Zisserman. Use what you have: Video retrieval using representations from collaborative experts, 2019. British Machine Vision Conference.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, 2019.

A. Miech, I. Laptev, and J. Sivic. Learning a text-video embedding from incomplete and heterogeneous data. *arXiv preprint arXiv:1804.02516*, 2018.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips. In *International Conference on Computer Vision (ICCV)*, 2019.

Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-End Learning of Visual Representations from Uncurated Instructional Videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Antoine Miech, Jean-Baptiste Alayrac, Ivan Laptev, Josef Sivic, and Andrew Zisserman. Thinking fast and slow: Efficient text-to-visual retrieval with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9826–9836, June 2021.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

H. Nam, J.W. Ha, and J. Kim. Dual attention networks for multimodal reasoning and matching. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

M. Patrick, P. Y. Huang, Y. Asano, F. Metze, A. Hauptmann, J. Henriques, and A. Vedaldi. Support-set bottlenecks for video-text representation learning, 2020. International Conference on Learning Representations.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015.

A. Rohrbach, A. Torabi, M. Rohrbach, N. Tandon, C. Pal, H. Larochelle, A. Courville, and B. Schiele. Movie description. *International Journal of Computer Vision*, 123: 94–120, 2017.

Aneeshan Sain, Ayan Kumar Bhunia, Yongxin Yang, Tao Xiang, and Yi-Zhe Song. Stylemeup: Towards style-agnostic sketch-based image retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8504–8513, June 2021.

Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: Learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics*, 35(4): 1–12, 2016.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45 (11):2673–2681, 1997.

Yuming Shen, Li Liu, Fumin Shen, and Ling Shao. Zero-shot sketch-image hashing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective, 2016. In Advances in Neural Information Processing Systems.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

A. Torabi, N. Tandon, and L. Sigal. Learning language visual embedding for movie understanding with natural language. *arXiv preprint arXiv:1609.08124*, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Cédric Villani. *Optimal Transport: Old and New*. Springer, 2008.

Jiang Wang, Yang Song, T. Leung, C. Rosenberg, Jingbin Wang, J. Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Kaiye Wang, Qiyue Yin, Wei Wang, Shu Wu, and Liang Wang. A comprehensive survey on cross-modal retrieval. *arXiv preprint arXiv:1607.06215*, 2016.

Xiaohan Wang, Linchao Zhu, and Yi Yang. T2vlad: Global-local sequence alignment for text-video retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5079–5088, June 2021.

Xinshao Wang, Yang Hua, Elyor Kodirov, Guosheng Hu, Romain Garnier, and Neil M Robertson. Ranked list loss for deep metric learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019a.

Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R. Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5022–5030, 2019b.

M. Wray, H. Doughty, and D. Damen. On Semantic Similarity in Video Retrieval, 2021. Computer Vision and Pattern Recognition.

Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

J. Xu, T. Mei, T. Yao, and Y. Rui. MSR-VTT: A large video description dataset for bridging video and language. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Sasikiran Yelamarthi, Shiva Krishna Reddy M, Ashish Mishra, and Anurag Mittal. A zero-shot framework for sketch based image retrieval. In *European Conference on Computer Vision (ECCV)*, 2018.

Y. Yu, H. Ko, J. Choi, and G. Kim. End-to-end concept word detection for video captioning, retrieval, and question answering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Y. Yu, J. Kim, and G. Kim. A joint sequence fusion model for video question answering and retrieval. In *European Conference on Computer Vision (ECCV)*, 2018.

Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-aware deeply cascaded embedding. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

Qi Zhang, Zhen Lei, Zhaoxiang Zhang, and Stan Z. Li. Context-aware attention network for image-text retrieval. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Z. Zheng, L. Zheng, M. Garrett, Y. Yang, and Y.D. Shen. Dual-path convolutional image-text embedding. *arXiv preprint arXiv:1711.05535*, 2017.

L. Zhou, C. Xu, and J. J. Corso. Towards automatic learning of procedures from web instructional videos. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.

Mo Zhou, Zhenxing Niu, Le Wang, Zhanning Gao, Qilin Zhang, and Gang Hua. Ladder Loss for Coherent Visual-Semantic Embedding. In *AAAI Conference on Artificial Intelligence*, 2020.

Appendix

- Visualization of Learned Clusters (Sec. A)
- Ablation Study (Sec. B)
- (Detailed) Text-based Video Retrieval (Sec. C)
- (Detailed) Image-Text Retrieval (Sec. D)
- (Extra Experiments) Synthetic Data (Sec. E)

A Visualization of Learned Clusters

As qualitative analysis, we visualize the learned clusters to see if they capture meaningful semantic information. On MS-COCO (trained with the number of classes $K = 1000$), we organize images and texts by their assigned cluster labels using the learned prototype classification model,

$$p(y = j|x^M) = \frac{\exp(p_j^\top \phi^M(x)/\tau)}{\sum_l \exp(p_l^\top \phi^M(x)/\tau)}, \quad M \in \{A, B\} \tag{7}$$

in our SwAMP. We first visually inspect individual clusters, images and texts that belong to each cluster. Some examples are shown in Fig. 4 and Fig. 5, each cluster contains semantically coherent data samples. Then we inspect texts (captions) in each cluster, and select a few keywords, those words that appear the most frequently in the texts. These keywords for each cluster consist of objects (noun) and/or actions (verb) that faithfully describe the cluster and data samples that belong to it. The full list is provided in Fig. 3.

A.1 Clustering based on Whole Contents

In Fig. 4 and Fig. 5, looking at clusters:

- Group-A $= (0100, 0234, 0359, 0405, 0428)$
- Group-B $= (0195, 0208, 0221, 0253)$
- Group-C $= (0180, 0683)$

the clusters within these groups are all related to the semantic meaning of *pizza*, *baseball*, and *cat*, respectively. However, they have different details in either activity or focus/scene: *man eating*, *people and table*, *on plate*, *oven*, and *with woman* for Group-A; *swing*, *on field*, *base playing*, and *crowd scene* for Group-B; *on bed* and *television* for Group-C. This means that SwAMP finds clusters based on the whole contents (objects, acitivities, and scenes), instead of doing merely object-based clustering.

A.2 Class Imbalance

Although we have roughly equal numbers of samples per cluster, we can see many redundant/repeated clusters in Fig. 3. This means that some clusters are overlapped with others in terms of semantic meaning, constituting larger *super*-clusters. These clusters are related to dominant data samples (e.g., cat, dog, tennis, baseball). This implies that the SwAMP can effectively deal with imbalance of semantic classes that can reside in data.

B Ablation Study

In our experiments, we chose the hyperparameters by cross validation with grid search. We perform empirical study on the impact of two important hyperparameters in our model: the number of classes K and SwAMP loss trade-off λ.

Number of classes (K). Recall that the best K values we chose were: $K = 1000$ for the image-text retrieval datasets and $K = 500$ for text-based video retrieval. To see how the retrieval performance is affected by other choices of K, we conduct experiments by varying K around the optimal values. The results are shown in Fig. 6, Fig. 7, Fig. 8, Fig. 9, and Fig. 10. Clearly, very small K has low retrieval performance (R@1), and increasing K leads to improvement. However, beyond certain points, there is no benefit of increasing K and we even see performance degradation, which agrees with the observations from previous work (Asano et al., 2020; Caron et al., 2020). This is perhaps due to the difficulty of assigning meaningful cluster labels in optimal transport. Overall, with properly chosen K, SwAMP outperforms contrastive learning, signifying that SwAMP's grouping/clustering of similar instances is more effective than vanilla instance discrimination. The fact that the optimal K values are different in two tasks (image-text and video-text) implies that the best cardinality of semantic clusters is highly dependent on the dataset characteristics (e.g., size and semantic diversity).

SwAMP loss trade-off (λ). We perform sensitivity analysis on λ, the strength of the SwAMP loss. For different values of λ, the retrieval scores (R@1) are shown in Fig. 11, Fig. 12, Fig. 13, Fig. 14, and Fig. 15. Our model remains better than contrastive learning for large intervals of different λ's, and the performance is not very sensitive to λ.

C Text-based Video Retrieval

We consider the text-to-video retrieval task where the goal is to find the most relevant video clip for a given natural language text query. We consider three datasets for this task: i) **YouCook2** (Zhou et al., 2018) of cooking videos and instructions, ii) **MSR-VTT** (Xu et al., 2016) of generic videos and captions from YouTube, and iii) **LSMDC** (Rohrbach et al., 2017) of movie clips and subtitles. All these datasets provide pairs of video clip and its text description, forming a multi-modal paired data format $(text, video)$ which conforms to our SwAMP framework.

For the raw text/video features and the feature extractor networks, as well as the training/test protocols, we follow the methods in (Miech et al., 2019). Whereas the details of the datasets and experimental setups are described in the subsequent sections, the features are specifically built by the following procedures. First, the raw features are obtained by the pretrained networks: (a) raw video features (4096D) are concatenation of frame-level and video-level features extracted from the pretrained 2D/3D CNNs (the ImageNet pre-trained Resnet-152 (He et al., 2016) for 2D features and the Kinetics (Carreira and Zisserman, 2017) pre-trained ResNeXt-101 16-frame model (Hara et al., 2018) for 3D features), (b) raw text features (4096D) are the GoogleNews pre-trained word2vec embeddings (Mikolov et al., 2013) for the pre-processed transcribed video narrations with the common stop words removed. Then the feature extractor networks $\phi^{video}(\cdot)$ and $\phi^{text}(\cdot)$ transform these raw features into 4096D features by the sigmoid-gated linear transform where the gating functions are two-layer linear networks (Miech et al., 2018). We fix the raw features and train only the latter sigmoid-gated networks, which comprise about 67M parameters.

Following (Miech et al., 2019), there are two training strategies: i) No-pretraining (No-PT) where the feature extraction networks are randomly initialized, and the training is done on the training split of the dataset, and ii) Pretraining (PT) where the feature extractors are first pretrained on the large-scale HowTo100M dataset (Miech et al., 2019), and finetuned on the target dataset. In (Miech et al., 2019), they adopt the contrastive (triplet) loss for training the feature extractors. Although we also compare our approach with the state-of-the-arts, the main focus in this experiment is to demonstrate the performance improvement achieved by the proposed SwAMP loss against vanilla contrastive learning. The SwAMP hyperparameter λ, the weight/impact of the SwAMP loss against the contrastive loss is chosen as $\lambda = 0.25$ for all three datasets, except the LSMDC-PT case for which $\lambda = 0.1$. We also choose temperature in softmax $\tau = 0.25$, entropic regularization trade-off in SK $\eta = 5.0$, the number of classes $K = 500$, and the queue size $2,048$ for the SwAMP. The other learning hyperparameters common in SwAMP and contrastive losses are not changed from (Miech et al., 2019).

YouCook2. This cooking video dataset collected from YouTube, contains 89 recipes and 14K video clips annotated with textual descriptions from paid human workers. The test data are formed by taking 3.5K clips from the validation set, and the test set comprises of $3,350$ pairs. The retrieval performance metrics are recall-at-k (R@k) with $k = 1, 5, 10$ and the median rank (Med-R). Hence, the random guess attains R@1$= 0.03\%$ Med-R=$1,675$. The results are summarized in Table 7. In the bottom four rows, we see the performance improvement achieved by the proposed SwAMP against the contrastive loss (Miech et al., 2019). For both training strategies, No PT (random model initialization) and PT (initialized with the HowTo100M-pretrained model), our SwAMP improves the retrieval performance significantly (e.g., about 12% reduction in

Table 7: Text-video retrieval results on YouCook2.

Methods	R@1 $\uparrow$	R@5 $\uparrow$	R@10 $\uparrow$	Med-R $\downarrow$
Random	0.03	0.15	0.3	1675
FV-CCA	4.6	14.3	21.6	75
Contrastive (No PT)	4.2	13.7	21.5	65
SwAMP (No PT)	**4.8**	**14.5**	**22.5**	**57**
Contrastive (PT)	8.2	24.5	35.3	24
SwAMP (PT)	**9.4**	**24.9**	35.3	**22**

Table 8: Text-video retrieval results on MSRVTT.

Methods	R@1 $\uparrow$	R@5 $\uparrow$	R@10 $\uparrow$	Med-R $\downarrow$
Random	0.1	0.5	1.0	500
C+LSTM+SA+FC7	4.2	12.9	19.9	55
VSE-LSTM	3.8	12.7	17.1	66
SNUVL	3.5	15.9	23.8	44
Temporal Tessellation	4.7	16.6	24.1	41
CT-SAN	4.4	16.6	22.3	35
JSFusion	10.2	31.2	43.2	13
Contrastive (No PT)	12.1	35.0	48.0	12
SwAMP (No PT)	**15.0**	**38.5**	**50.3**	**10**
Contrastive (PT)	14.9	40.2	52.8	9
SwAMP (PT)	**19.0**	**42.4**	**55.2**	**8**

Table 9: Text-Video retrieval results on LSMDC.

Methods	R@1 $\uparrow$	R@5 $\uparrow$	R@10 $\uparrow$	Med-R $\downarrow$
Random	0.1	0.5	1.0	500
C+LSTM+SA+FC7	4.3	12.6	18.9	98
VSE-LSTM	3.1	10.4	16.5	79
SNUVL	3.6	14.7	23.9	50
Temporal Tessellation	4.7	15.9	23.4	64
CT-SAN	4.5	14.1	20.9	67
JSFusion	9.1	21.2	34.1	36
Contrastive (No PT)	7.2	18.3	25.0	44
SwAMP (No PT)	**7.7**	**19.3**	**27.7**	**40**
Contrastive (PT)	7.1	19.6	27.9	40
SwAMP (PT)	**8.3**	**20.0**	**28.9**	**37**

Median Rank for the No PT case). SwAMP also outperform the CCA baseline FV-CCA (Klein et al., 2015).

MSRVTT. This generic video-text dataset (Xu et al., 2016) collected from YouTube contains videos of specific categories including music, sports, and movie. There are 200K video-caption pairs obtained by human annotation. We follow the retrieval training/test protocol of (Yu et al., 2018; Miech et al., 2019). The test set consists of 1K pairs. As reported in Table 8, our SwAMP loss improves the performance over the contrastive learning significantly for both no-pretraining and pretraining cases: about 24% in R@1 in the No PT case, and 27% in the PT case. Furthermore, the SwAMP outperforms with large margin the state-of-the-arts: C+LSTM+SA+FC7 (Torabi et al., 2016), VSE-LSTM (Kiros et al., 2014), Temporal Tessellation (Kauman et al., 2017), CT-SAN (Yu et al., 2017), and JSFusion (Yu et al., 2018).

LSMDC. The LSMDC (Rohrbach et al., 2017)[2] is a dataset of movie video clips, comprised of 101K video-caption pairs. The captions are collected either from the movie scripts or the audio descriptions. The test set contains 1K pairs. For this dataset, we use the SwAMP hyperparameter (impact of the SwAMP loss against the contrastive loss) $\lambda = 0.1$ for the PT case. The results are shown in Table 9. Similar to the other two datasets, our SwAMP is consistently better than the contrastive learning (about $7 \sim 9\%$ in Median Rank).

[2] `https://sites.google.com/site/describingmovies/lsmdc-2016/movieretrieval`

D Image-Text Retrieval

For the image-text cross-modal retrieval task, we follow the features and protocols from the well-known *stacked cross attention network* (SCAN) (Lee et al., 2018). In their framework, each image is represented by a set of local features $V = \{v_1, \ldots, v_k\}$, where v_i $(\in \mathbb{R}^D) = W_v f_i + b_v$ and f_i's are the CNN features extracted from salient image regions detected by the Faster-R-CNN model (Ren et al., 2015). The raw features f_i's are fixed and $\{W_v, b_v\}$ are learnable parameters. The text (sentence) is also treated as a set of word features $E = \{e_1, \ldots, e_n\}$, where e_i $(\in \mathbb{R}^D) = (h_i^{lr} + h_i^{rl})/2$ and $h_i^{lr/rl}$ are the outputs of the bi-directional GRU (Bahdanau et al., 2015; Schuster and Paliwal, 1997) with the sequence of word embeddings as input. Both the word embeddings and GRU parameters are learnable. These image/text features contain rich local information, however, one challenge is that both representations are *sets*, hence the number of elements (k and n) can vary from instance to instance.

In the original SCAN paper (Lee et al., 2018), they proposed a cross-modal attention model, where each local feature from one modality is transformed by the attention (Vaswani et al., 2017) with the set of local features in the other modality; e.g., v_i is transformed to $attn(v_i; \{e_j\}_{j=1}^n)$ = the weighted sum of *values* $\{e_j\}_{j=1}^n$ with v_i as a *query* and $\{e_j\}_{j=1}^n$ as *keys* (this denoted by i-t, while the other attention direction t-i can be used alternatively). Then the similarity score between image V and text E is defined as $pool(\{cos(v_i, attn(v_i; \{e_j\}_{j=1}^n))\}_{i=1}^K)$, where $cos(a, b)$ is the cosine similarity and $pool$ is the pooling operation, either of AVG or LSE (log-sum-exp). Then the triplet contrastive loss is employed. For the details, please refer to (Lee et al., 2018).

Note that in the SCAN, there is no succinct modality-wise embedding vector representation, but the similarity score between instances of two modalities is rather computed by highly complex attention operations. Although this is helpful for capturing the interactions between local features, computing the similarity score takes quadratic time in the number of elements (local features) in the instances. This is time consuming compared to simple dot-product of the modality-wise embedding vectors (See Table 12 for the actual running times compared with the approaches based on modality-wise feature representation). Moreover, it is not applicable to our SwAMP approach since we need to predict the class labels for each modality from modality-wise representation $\phi^{image}(V)$, $\phi^{text}(E)$.

To have modality-wise representation, we adopt the idea of *induced-set attention* (ISA) from the Set Transformer (Lee et al., 2019). Specifically, we introduce p learnable prototype (query) vectors $\{q_j\}_{j=1}^p$ where $q_j \in \mathbb{R}^D$. Then we compute the attention for each query with V (or E), i.e., $z_j = attn(q_j; \{v_i\}_{i=1}^k)$. Then we define $\phi^{image}(V) = concat(z_1, \ldots, z_p)$, similarly for $\phi^{text}(E)$, where $concat$ refers to concatenation. Thus the parameters for $\phi^{image}()$ are $\{W_v, b_v\}$ and $\{q_j\}_{j=1}^p$, and the parameters for $\phi^{text}()$ are the word embeddings, GRU parameters, and $\{q_j\}_{j=1}^p$. We share the same $\{q_j\}_{j=1}^p$ for both modalities. We also have multi-head extension by computing these features multiple times and concatenating them. We call these modality-wise features as *prototype attention representation* (PAR). Note that computing PAR features has linear complexity in the number of local features (assuming p is constant), and the cross-modal similarity is simply dot-product of PAR features, and can be computed in linear time (See also Table 12).

D.1 Datasets and Results

We test our approach on the popular image-text retrieval datasets, MS-COCO and Flickr30K. There are 31K images and five captions for each image in Flickr30K. MS-COCO contains $123,287$ images, where each image is annotated with five text descriptions. Following the widely-used split (Karpathy and Fei-Fei, 2015; Faghri et al., 2018), for the Flickr30K, we have 1K images for validation, 1K images for testing, and the rest for training. For MS-COCO, there are 5K test images (and 25K captions, five captions for each image). We also follow two standard protocols for measuring the test retrieval performance for MS-COCO: 1) using the entire 5K test images or 2) splitting the test set into 5 folds and report the average retrieval performance over the 5 folds.

The results are summarized in Table 10 (Flickr) and Table 11 (MS-COCO). We specifically highlight the comparison between the contrastive loss and our SwAMP loss with the modality-wise feature representation (Contrastive-PAR vs. SwAMP-PAR). For the PAR features, we choose the number of prototypes $p = 20$, attention weight temperature $T = 0.5$, and the number of heads $H = 1$ for Flickr, and $p = 10, T = 0.5, H = 2$ for MS-COCO. For the SwAMP hyperparameters, we use the impact of SwAMP loss $\lambda = 1.0$, softmax temperature $\tau = 0.025$, the number of classes $K = 1,000$, queue size $1,280$ for both datasets. As shown, the SwAMP loss performs consistently better than the contrastive loss. SwAMP also outperforms several state-of-the-arts including the recent sophisticated probabilistic embedding strategy (Chun et al., 2021).

When compared with the computationally expensive SCAN, SwAMP mostly outperforms SCAN except for the SCAN's best attention direction/combination choices. Note that SwAMP uses the simple feature aggregation strategy (PAR) to have

Table 10: Image-text retrieval results on Flickr30K.

Methods	Image → Text			Text → Image		
	R@1	R@5	R@10	R@1	R@5	R@10
DAN (Nam et al., 2017)	55.0	81.8	89.0	39.4	69.2	79.1
DPC (Zheng et al., 2017)	55.6	81.9	89.5	39.1	69.2	80.9
VSE++ (Faghri et al., 2018)	52.9	-	87.2	39.6	-	79.5
SCO (Huang et al., 2018)	55.5	82.0	89.3	41.1	70.5	80.1
SCAN i-t AVG	67.9	89.0	94.4	43.9	74.2	82.8
SCAN t-i AVG	61.8	87.5	93.7	45.8	74.4	83.0
SCAN t-i AVG + i-t LSE	67.4	90.3	95.8	48.6	77.7	85.2
Contrastive-PAR	65.7	86.8	92.4	48.2	75.8	84.2
SwAMP-PAR	**67.8**	**88.5**	**94.0**	**49.1**	**76.1**	**83.7**

Table 11: Image-text retrieval results on MS-COCO.

Methods	5-fold (1K test images)						Entire (5K test images)					
	Image → Text			Text → Image			Image → Text			Text → Image		
	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
DPC (Zheng et al., 2017)	65.6	89.8	95.5	47.1	79.9	90.0	41.2	70.5	81.1	25.3	53.4	66.4
VSE++ (Faghri et al., 2018)	64.6	-	95.7	52.0	-	92.0	41.3	-	81.2	30.3	-	72.4
GXN (Gu et al., 2018)	68.5	-	97.9	56.6	-	94.5	42.0	-	84.7	31.7	-	74.6
SCO (Huang et al., 2018)	69.9	92.9	97.5	56.7	87.5	94.8	42.8	72.3	83.0	33.1	62.9	75.5
PCME (Chun et al., 2021)	68.8	-	-	54.6	-	-	44.2	-	-	31.9	-	-
SCAN i-t	69.2	93.2	97.5	54.4	86.0	93.6	46.4	77.4	87.2	34.4	63.7	75.7
SCAN t-i + i-t	72.7	94.8	98.4	58.8	88.4	94.8	50.4	82.2	90.0	38.6	69.3	80.4
Contrastive-PAR	71.8	94.3	97.9	56.8	86.9	93.8	48.4	78.1	88.1	34.3	64.4	76.2
SwAMP-PAR	**72.6**	**94.6**	**98.0**	**57.4**	**87.6**	**94.1**	**49.7**	**79.1**	**88.3**	**35.0**	**65.1**	**76.6**

Table 12: Running time comparison for SCAN (cross-modal attention) and our SwAMP-PAR. Running times (seconds) are measured on the same machine (Core i7 3.50GHz CPU, 128GB RAM, and a single GeForce RTX-2080Ti GPU). We report per-batch times for training, and entire retrieval times for test. For MS-COCO test, the running times for 5K test images are reported, where times for 1K test images averaged over 5 folds are shown in the parentheses. For SCAN, when we use features in both directions (e.g., t-i AVG + i-t LSE), the running times are roughly doubled.

Methods	Flickr30K		MS-COCO	
	Train	Test	Train	Test
SCAN i-t AVG	0.35	336.9	0.33	9352.0 (350.3)
SwAMP-PAR	0.09	3.8	0.08	25.9 (16.3)

fast and succinct modality-wise feature representation, whereas SCAN relies on the cross-modal attention similarity scoring model, which is computationally expensive. To see the computational advantage of SwAMP-PAR, we compare the actual training/test times for the two approaches in Table 12, measured on the same machine with a single GPU (RTX 2080 Ti), Core i7 3.50GHz CPU, and 128 GB RAM. As shown, our SwAMP-PAR is about 4 times faster than SCAN for training on both datasets, while the difference becomes even more pronounced during test; SwAMP-PAR is about two orders of magnitude faster than the cross-modal attention model.

E Synthetic Data

In this section we devise a synthetic dataset not only for performing the proof-of-concept test of our SwAMP algorithm, but also analyze the impacts of the various hyperparameters and training options in the proposed algorithm. For the former, we especially focus on the retrieval performance improvement achieved by our SwAMP compared to the contrastive loss or its popular variants (e.g., online hard-example mining loss).

The dataset is constructed by the following procedure: We randomly generate 20 Gaussians in $\mathbb{R}^5$, each of which is considered to represent a *semantic class*. For each Gaussian (class), we sample a latent vector $z \in \mathbb{R}^5$, and a pair of instances

Table 13: Retrieval results on the synthetic data.

Error type	Method	R@1 ↑	R@5 ↑	R@10 ↑	Med-R ↓
Pair-based	Contrastive	84.10	98.60	99.55	1
	SwAMP	**90.80**	**99.95**	**100.0**	**1**
Class-based	Contrastive	91.60	99.70	99.90	1
	SwAMP	**95.70**	**99.95**	**100.0**	**1**

$(x^A \in \mathbb{R}^{100}, x^B \in \mathbb{R}^{100})$ is then generated by $x^A = f_A(z)$ and $x^B = f_B(z)$ where f_A and f_B are randomly initialized fully-connected DNNs with two hidden layers of 50 units. We generate 500 pairs for each class that leads to $10,000$ data pairs, and split them into 7000/1000/2000 train/validation/test sets. The validation recall-at-1 (R@1) performance is evaluated at every training epoch, and the model at the epoch with the best validation performance is selected as the final model. Note that during training we only use the paired data (x^A, x^B) with the semantic class labels hidden to the training algorithms.

For training, we adopt the embedding networks $\phi^A(x^A)$ and $\phi^B(x^B)$ as fully-connected neural nets with two hidden layers of 50 units. The embedding dimension is chosen as 5. We train the model with this same network architecture, using the contrastive loss and our SwAMP loss. For both loss functions, the batch size is 128, and the Adam optimizer (Kingma and Ba, 2015) is used with learning rate 10^{-3}, and the maximum epoch is 100.

For the contrastive loss, we adopt the (online) hard-example mining with the margin parameter $\alpha = 0.1$. For the SwAMP loss, the defaults parameters are as follows: temperature $\tau = 0.01$ for the softmax classifier, the reciprocal impact of the max-entropy regularizer for the Sinkhorn-Knopp $\eta = 1/0.05$ (i.e., we add the entropic regularizer with the weight $\eta^{-1} = 0.05$ to the objective of the OT problem. Also, by default, we choose the number of classes $K = 1000$ and the queue size $1,280$, 10 times the batch size (and greater than K). For both loss functions, the embedding networks are initialized randomly.

For test, we perform the cross-modal retrieval task $x^A \to x^B$, treating each x^A in the test set as a query, retrieving x^B from the test set. There are two ways to define the retrieval error: i) *pair-based* which treats the retrieved x^B as a correct retrieval only if the query x^A and the retrieved x^B are found as a pair in the data, and ii) *class-based* which compares only the classes of the query x^A and the retrieved x^B. Hence the pair-based error is more strict than the class-based since it counts only the data item that appears in the data as correct retrieval, without comparing the semantic classes of the retrieved item and the query.

E.1 Ablation study on hyperparameters

There are several hyperparameters in our SwAMP model, and we have conducted several ablation-type study on the impacts of the hyperparameters. The hyperaparameters that are deemed to be the most critical are: i) the number of classes K, ii) the size of the queues, iii) initialization of the feature extraction networks (either random initialization or pretrained one with the contrastive loss), iv) entropic regularization trade-off η in Sinkhorn-Knopp, and v) the soft/hard cluster assignment after OT clustering.

Number of classes (K). We vary the number of classes K for 200, 500, 1000, 2000, 3000, and record the R@1 scores for both pair and class based error types for our SwAMP model. The results are shown in Fig. 16. We see that allowing more clusters improves the performance. However, once K is around 1000 or greater than 1000, there is no significant benefit of increasing K. This implies that SwAMP does not merely do instance discrimination, but seeks for grouping/clustering of similar instances. Although we did not include it in the figure, having $K = 20$, i.e., the true number of semantic classes, yielded poor performance (worse than $K = 200$). This means that it is very difficult to expect that the model would discover the underlying semantic classes correctly.

Size of queues. Another important hyperparameter is the size of the queues, where the OT clustering is performed on the latest features that are stored in the queues. In addition to the default queue size $1280 = 10 \times 128$ (batch size), we try with different queue sizes $\{0, 1, 2, 5, 20\} \times 128$. Note that the OT clustering is performed on the union of the features in the queue and the current batch, hence zero queue size implies that we only use the current batch for OT clustering. The results are reported in Fig. 17. As shown, increasing the queue sizes accordingly improves the performance, where with the queue size of two times the batch size outperforms the contrastive loss. Also, not using the queues ("No queue") resulted in poor performance, signifying the importance of using the queues. Interestingly, too large queue size ($20\times$) deteriorates the performance, which might be explained by the negative effects of the stale features obtained several iterations ago from the old feature extractor networks. This suggests the trade off of the queue size: too small queue size does not generalize well to the clustering of entire data, while too large queue size can be harmful due to the inconsistent stale features.

Initialization of feature extractor networks. In our default setup, the feature extractor networks $\phi^A(\cdot)$ and $\phi^B(\cdot)$ are initialized randomly. Now we test the performance of the SwAMP when the feature extractor networks are initialized from the pretrained ones by the contrastive loss training. We initially expected that this warm-start training may expedite the training with the SwAMP loss, however, as the results in Fig. 18 indicates, it does not outperform the random initialization although the warm-start is still better than contrastive loss training. This may imply that the SwAMP loss defines a very different loss landscape from the contrastive loss, and the contrastive-loss optimized model may lie at the region far from the optima of the SwAMP loss, thus the warm-start even hinders convergence to the SwAMP optima.

Impact of the entropic regularization ($1/\eta$). In the Sinkhorn-Knopp (SK) algorithm, we have the reciprocal trade-off $1/\eta$ for the entropy term of the optimization variables $q(y|x)$. Too much emphasizing the entropy term (by increasing $1/\eta$ or decreasing η) would lead to near uniform $q(y|x)$, which means that it carries little information about the meaningful classes, and cluster assignment can be more or less random. On the other hand, having too small impact of the entropy term would make the SK algorithm converge too slowly, and the output of the SK with only a few iterations would produce non-optimal solutions. To see the impact, we vary $1/\eta$ from 0.01, 0.05 (default), and 0.1, and the results are shown in Fig. 19. We see that there is slight performance degradation for small and large $1/\eta$ values from the optimal choice.

Soft or hard cluster assignment after OT. We also check if the hard cluster assignment thresholding after OT optimization would be beneficial or not. Recall that the default is to use the output $q(y|x)$ of the SK algorithm as it is (i.e., soft cluster assignment). In the hard assignment we further threshold $q(y|x)$ to have one-hot encoding, which is then used in the cross-entropy loss optimization. As shown in Fig. 20, the hard assignment is harmful, which implies that retaining uncertainty in cluster estimation is important to have accurate clustering and feature learning.

0000: skateboard
0001: airplane/fly
0002: bears/teddy
0003: giraffes
0004: pizza
0005: phone/cell
0006: skis/snow
0007: train/bridge
0008: vase/flowers
0009: man/sandwich
0010: room/chairs
0011: cat/laptop
0012: food/tray
0013: skateboard/boy
0014: giraffe/feeding
0015: baseball
0016: tennis/racket
0017: man/tie
0018: bus/double
0019: wine/glass
0020: water/skiing
0021: dogs
0022: traffic light
0023: doughnut
0024: birds
0025: microphone
0026: boat
0027: bananas
0028: vase/sitting
0029: dog/frisbee
0030: sheep/herd
0031: umbrella
0032: tennis/court
0033: tennis/ball
0034: train
0035: woman/phone
0036: shower/bathroom
0037: bear/black
0038: dog/walking
0039: tennis/player
0040: giraffe/grass
0041: bicycle/bike
0042: giraffe
0043: tennis/woman
0044: cakes/cupcakes
0045: bowls/food/table
0046: food/table
0047: zebras
0048: bench/sitting
0049: sign/stop
0050: group/people
0051: standing/couple
0052: cake
0053: tennis/group
0054: flying/airplane
0055: hydrant/red
0056: window/mirror
0057: luggage/airport
0058: tennis/boy
0059: airport/runway
0060: cat/bowl
0061: elephant
0062: laptop
0063: kite/beach
0064: wine
0065: wave/surfboard
0066: baby/bed/laying
0067: table/chairs
0068: cake/birthday
0069: riding/bike
0070: table/plates
0071: pizza
0072: laptop/bed
0073: banana
0074: laptop/monitor
0075: fries/sandwich
0076: cake/strawberries
0077: kitchen
0078: oven
0079: kitchen
0080: water/kites
0081: room
0082: street/signs
0083: boat
0084: vegetables/market
0085: pizza/wooden
0086: boat/people
0087: tennis/ball
0088: teddy/bear
0089: bus/standing
0090: cows
0091: laptop/computer
0092: surfboard/water
0093: cat/sitting
0094: traffic/light
0095: zebra/grazing
0096: scissors
0097: cat/television
0098: snow/snowboard
0099: bear/brown
0100: pizza/man/eating
0101: flying/plane
0102: motorcycles/police
0103: truck
0104: train/tracks
0105: elephant
0106: frisbee/beach
0107: cows/field
0108: dog/kitchen
0109: trucks/parked
0110: bathroom/mirror
0111: skateboard/stairs
0112: donuts
0113: building
0114: cows/laying
0115: skateboard/riding
0116: cow/standing
0117: wave/surfboard
0118: train/station
0119: surfboard/riding
0120: clock/building
0121: baseball/glove
0122: baseball/bat
0123: woman/mirror
0124: beach/cow
0125: tennis/girl
0126: tennis/court
0127: bench/park
0128: snowboard/snow
0129: kitchen
0130: sheep/goats
0131: phone/talking
0132: dog/bottle
0133: mirror/reflection
0134: bus/street
0135: school/bus/yellow
0136: tennis/women
0137: man/tie/shirt
0138: tennis/man
0139: video/game
0140: sheep/hay
0141: cake/cutting
0142: child/skis/
0143: snowboard
0144: bananas/fruit
0145: train/steam
0146: bears/stuffed
0147: beach/frisbee
0148: beach/kite
0149: boy/game
0150: bananas
0151: sign/street
0152: toilet/bathroom
0153: cake/cutting
0154: beach
0155: kites/flying
0156: cow/fence
0157: dog/skateboard
0158: horse/jumping
0159: woman/phone
0160: bench/park
0161: bathroom/sink
0162: urinals/bathroom
0163: beach/playing
0164: people/snow
0165: child/cow
0166: laptop
0167: bench/woman
0168: man/bike
0169: soccer/playing
0170: horse/man
0171: kite
0172: kitchen/stove
0173: bus/street
0174: street/city
0175: pizza/eating
0176: pizza/box
0177: tennis/ball
0178: bus/green
0179: skateboard/air
0180: cat/bed
0181: elephants
0182: train/man
0183: woman/frisbee
0184: wine/glass
0185: book/reading
0186: snow/skis
0187: bowl/fruit
0188: couch/sitting
0189: bus/red
0190: man/suit/tie
0191: bed/bear/teddy
0192: elephants
0193: train/people
0194: bathroom/toilet
0195: baseball/swinging
0196: bed/bunk/room
0197: boat/water
0198: television
0199: kitchen/pots/pans
0200: birds/flying
0201: fire/truck
0202: buses/parked
0203: umbrella/holding
0204: man/phone
0205: boy/table/sitting
0206: skis/snow
0207: tennis/court
0208: baseball/field
0209: snow/skiing
0210: bear/tree
0211: ocean/surfers
0212: dog/man/frisbee
0213: cat/desk/computer
0214: bear/walking
0215: elephant
0216: sandwich/plate
0217: clock/brick/tower
0218: horse/carriage
0219: people/game
0220: clock/pole
0221: baseball/base
0222: oven/woman
0223: rice/plate
0224: bus/double/decker
0225: blender/counter
0226: elephants
0227: giraffe
0228: baby/teeth
0229: boats/harbor
0230: giraffes/tree
0231: snow/covered
0232: room/luggage
0233: toilet/urinal
0234: pizza/table
0235: bears
0236: clock/building
0237: cat/bed
0238: baseball/player
0239: cat/suitcase
0240: wine/plate/table
0241: dog/beach
0242: sign/street
0243: people/sitting
0244: wine/glasses
0245: dog/car
0246: cup/coffee
0247: table/sitting
0248: elephant
0249: plate/fries
0250: baseball
0251: vegetables/fruits
0252: toilet/bathroom
0253: baseball/crowd
0254: broccoli/plant
0255: sandwich/salad
0256: luggage/suitcase
0257: refrigerator
0258: flying/sky/jet
0259: bathroom/toilet
0260: bathroom/shower
0261: broccoli/bowl
0262: food/plates
0263: people/group
0264: baseball/batter
0265: pizza/cheese
0266: skateboard/tricks
0267: boat/water
0268: airplane/runway
0269: plane/field
0270: umbrellas
0271: bed/bedroom
0272: cake/cutting
0273: remote/control
0274: zebra
0275: dog/sitting
0276: table/eating
0277: man/car
0278: train/blue
0279: group/standing
0280: cat/couch
0281: man/bed
0282: giraffe/grass
0283: bus/parked
0284: train/red
0285: desk/computers
0286: bench/sitting
0287: train/traveling
0288: bathroom/window
0289: toothbrush
0290: clock/tower
0291: refrigerator
0292: donuts/doughnuts
0293: hydrant/fire
0294: room/living
0295: giraffes
0296: wave/surfboard
0297: skateboard
0298: woman/phone
0299: bird/flying
0300: umbrella/holding
0301: horse/cows
0302: street
0303: banana
0304: Wii/remote
0305: water/bears
0306: kites/flying
0307: game/Wii
0308: bike/riding
0309: bed/woman
0310: cats/bed
0311: sign/stop
0312: kites/sky
0313: desk/office/chair
0314: light/traffic
0315: motorcycle
0316: man/woman
0317: monitor/keyboard
0318: food/kitchen
0319: zebra/standing
0320: dog/boat
0321: plate/sandwiches
0322: skateboard
0323: bowl/soup
0324: traffic/cars
0325: giraffes
0326: dog/chair
0327: clock
0328: bike/parked
0329: bird/beach
0330: baseball/throwing
0331: sitting/suitcase
0332: cat/keyboard
0333: clock/sign
0334: motorcycle
0335: dog/bed
0336: clock/large
0337: bus/people
0338: bear/teddy
0339: horse
0340: bus/double/decker
0341: statue/clock
0342: elephants/riding
0343: street/people
0344: vases/flowers
0345: pizza/pepperoni
0346: man/luggage
0347: surfboard
0348: apple
0349: horse
0350: bed/dogs
0351: bathroom/toilet
0352: down/skiing/slope
0353: dog/truck
0354: laptop/computer
0355: sheep/fence
0356: horse/carriage
0357: men/table/sitting
0358: skateboard
0359: pizza/plate
0360: cat/mirror
0361: tennis/court
0362: dog/desk
0363: baseball/pitcher
0364: broccoli/pasta
0365: books
0366: skateboard
0367: boat/people
0368: sign/street
0369: living/television
0370: zebras/hay
0371: dog/window
0372: plate/food
0373: woman/cell/phone
0374: bed/laying
0375: buses/double
0376: horses/riding
0377: man/suit/tie
0378: refrigerator
0379: sign/road
0380: banana
0381: boy/surfboard
0382: kitchen/sink
0383: bananas/bunch
0384: cat/bench/wooden
0385: bed/laying
0386: elephant
0387: scissors
0388: sheep/field
0389: woman/playing
0390: microwave
0391: truck/parked
0392: luggage/stacked
0393: snow/skis
0394: flowers/vase
0395: kitchen/dining
0396: bear/teddy
0397: man/frisbee
0398: woman/sitting
0399: airplane/runway
0400: refrigerator
0401: giraffe/water
0402: phone/holding
0403: desk/computer
0404: sandwich/bread
0405: pizza/oven
0406: umbrellas
0407: table/sitting
0408: coffee/donut
0409: dog/bench
0410: snowboard
0411: motorcycle
0412: sign/stop
0413: phone/talking
0414: dirt/bike
0415: jet/runway
0416: people/sitting
0417: man/frisbee
0418: umbrellas
0419: clock/tower
0420: baseball/batter
0421: vase/flower
0422: clock
0423: suitcase/luggage
0424: kitchen/cabinets
0425: table/Wii
0426: motorcycle
0427: laptop
0428: pizza/woman
0429: baseball/catcher
0430: pizza/table
0431: zebras/grass
0432: sheep/grass
0433: horse/fence
0434: dog/hat
0435: tennis
0436: playing/game
0437: snowboard/slope
0438: bear/teddy
0439: man/tie/glasses
0440: street/night/city
0441: laptops
0442: train/yellow
0443: bus/stop
0444: room/television
0445: sign/stop
0446: kitchen/counter
0447: motorcycles
0448: tennis/swinging
0449: sitting/phone
0450: umbrellas/street
0451: sheep/herd
0452: phone/cell/talking
0453: boats/water
0454: bowl/oranges
0455: cake/candles
0456: mirror/bathroom
0457: giraffe
0458: bed/television
0459: bathroom/towels
0460: cow/woman/man
0461: bicycle
0462: bear/polar/white
0463: riding/bicycle
0464: bird/water
0465: zebras
0466: man/tie
0467: street/bikes
0468: train/station
0469: sign/stop
0470: cake
0471: cat
0472: room/living/couch
0473: cars/street
0474: kites/beach
0475: vegetables/pan
0476: train/tracks
0477: horse
0478: umbrella
0479: blender
0480: bedroom/wooden
0481: parking/meters
0482: airplane
0483: people/group
0484: snow/people
0485: street/cars
0486: water/cows
0487: apples/bowl
0488: pizza/salad
0489: sandwich
0490: food/plates
0491: cat/chair
0492: snow/people
0493: plate/coffee/cup
0494: frisbee/field
0495: elephant
0496: sandwich/soup
0497: store/people
0498: frisbee/people
0499: horse/riding
0500: fire/hydrant
0501: boats/water
0502: cake/cutting
0503: sandwich
0504: baseball
0505: hot/eating/dog
0506: snow/skier
0507: kitchen/microwave
0508: toilet/dirty
0509: bench/beach
0510: bird/man
0511: living/fireplace
0512: truck/parked
0513: airplane
0514: cat/table
0515: sheep/grazing
0516: soccer/ball
0517: cake/wedding
0518: bear/teddy
0519: plants/vase
0520: tennis/racquet
0521: umbrella
0522: items/table
0523: tennis/player
0524: bench/park
0525: frisbee/field
0526: snow/people
0527: skiing/snow
0528: luggage/suitcase
0529: laptop/computer
0530: cat/bag
0531: donuts
0532: people/sitting
0533: wave/surfboard
0534: bowl/vegetables
0535: cat/window
0536: baseball/bat
0537: oranges
0538: oven
0539: chairs/table
0540: train/track
0541: umbrella
0542: bed/window
0543: vase/flowers
0544: bathroom
0545: elephants
0546: skiers/snow
0547: kite
0548: bananas/market
0549: fries/hot/dog
0550: bird/perched
0551: hair
0552: tennis
0553: beach/surfboards
0554: skateboard
0555: giraffe
0556: cows
0557: bears/teddy
0558: laptop
0559: tennis/ball
0560: building
0561: horse
0562: computer/monitor
0563: phone/cell
0564: computer/laptop
0565: cake/chocolate
0566: phone/picture
0567: skateboard
0568: clock/wall
0569: people
0570: bird/perched
0571: laptop
0572: keyboard/desk
0573: sheep
0574: wave/surfboard
0575: refrigerator
0576: clock/tower
0577: laptop
0578: microwave/oven
0579: horses/water
0580: cat/shoes/pair
0581: mouse/keyboard
0582: toothbrush/teeth
0583: people/tables
0584: room/furniture
0585: skateboard
0586: fire/hydrant
0587: wave/riding
0588: cake/girl
0589: birds/water
0590: umbrella
0591: signs/street
0592: hot/dogs
0593: skateboard
0594: pizza
0595: motorcycles
0596: trains/station
0597: vase/flowers
0598: giraffe
0599: teeth/brushing
0600: train/bus
0601: man/horse
0602: stove/oven
0603: sign/stop
0604: train/traveling
0605: phone/cell/girl
0606: motorcycle
0607: skateboard
0608: hydrant/fire
0609: snow/snowboard
0610: kite/flying/sky
0611: zebra/grass
0612: plate/broccoli
0613: refrigerator
0614: plate/carrots
0615: city/water
0616: bear/teddy
0617: bathroom/toilet
0618: giraffes
0619: tennis/court
0620: dog/frisbee/mouth
0621: computer/monitors
0622: Christmas/tree
0623: plate/fruit
0624: table/sitting
0625: surfing
0626: sign/stop
0627: umbrella
0628: hydrant/fire
0629: kite/beach
0630: bench/sitting
0631: sandwich/plate
0632: boat/river
0633: elephant
0634: bear/teddy
0635: vegetables
0636: appliances
0637: kitchen/cabinets
0638: motorcycle
0639: bike/riding
0640: elephant
0641: bananas/bunch
0642: game/video/Wii
0643: airplane
0644: boy/wearing/tie
0645: dog/couch
0646: cat/bathroom
0647: toilet/sink
0648: tree/orange
0649: cows/grazing
0650: laptop
0651: clock/tower
0652: skateboards
0653: hot/dog
0654: vase/flowers
0655: bus/street
0656: motorcycle
0657: cake/white
0658: water/ocean
0659: bananas
0660: baseball/bat
0661: baseball/player
0662: cat/sitting/car
0663: man/sitting/grass
0664: zebras/herd
0665: umbrella
0666: bathroom/toilet
0667: dog
0668: truck
0669: cell/phones
0670: frisbee/throwing
0671: girl/eating/food
0672: group/people
0673: carrots/board
0674: airport
0675: clock/tower
0676: air/skis
0677: bird/window
0678: tennis/court
0679: beds/room
0680: luggage/car
0681: pizza/box
0682: eating/doughnut
0683: cat/television
0684: boy/frisbee
0685: motorcycle
0686: skateboard
0687: vegetables/rice
0688: skier/snow
0689: bench/people
0690: bed/bedroom
0691: bench/sitting
0692: kite/boy/girl
0693: cows/field
0694: cake/birthday
0695: fire/hydrant
0696: cat/car/sitting
0697: woman/table
0698: cat
0699: truck
0700: plate/donuts
0701: skateboard
0702: boat/truck
0703: pizza/plate
0704: airplane
0705: fruit/market
0706: pizza
0707: room/living
0708: benches/park
0709: room/living
0710: boat/water/small
0711: motorcycle
0712: truck/driving
0713: dog/ball
0714: man/suit/tie
0715: surfboard
0716: bridge/train
0717: kitchen/preparing
0718: airport/airplane
0719: cake/plate
0720: horses
0721: skier
0722: clocks/wall
0723: skis/child/young
0724: plate/orange
0725: baseball
0726: pizza/cheese
0727: cat/dog
0728: surfboard
0729: runway/airplane
0730: car
0731: bed/pillows
0732: horse/riding
0733: bird/sitting
0734: cows/road/walking
0735: kitchen/stove
0736: woman/tie/wearing
0737: motorcycle
0738: clock/tower
0739: truck
0740: motorcycle
0741: train/station
0742: sign/stop
0743: cat/bathroom
0744: surfboard/ocean
0745: ducks/swimming
0746: water/plane
0747: bikes
0748: orange
0749: toilet/sitting
0750: beach/umbrellas
0751: horse/beach
0752: bed
0753: toilet
0754: sign/stop
0755: bear/teddy
0756: giraffe
0757: brushing/teeth
0758: couch/laptop
0759: display/case
0760: game/video/Wii
0761: baseball
0762: surfboard/beach
0763: dog/sheep
0764: tennis/player
0765: zebras/grass
0766: kitchen/island
0767: ties
0768: living/room
0769: group/game/video
0770: sink/bathroom
0771: airplane
0772: people/cell
0773: bird/grass
0774: bike
0775: skiing
0776: train/graffiti
0777: elephants
0778: pizza/pan
0779: toilet/bathroom
0780: scissors
0781: window/vase/
0782: elephant/fence
0783: vases
0784: giraffe
0785: bench/wooden
0786: snowboard
0787: motorcycle
0788: sign/building
0789: dog/snow
0790: table/eating
0791: skateboard
0792: flying/kite
0793: scissors
0794: sink/bathroom
0795: baseball/bat
0796: kitchen/counter
0797: hot/dog
0798: skateboard
0799: bench/snow/
0800: surfboard/beach
0801: suitcase/luggage
0802: baseball/pitcher
0803: bathroom/tub
0804: kitchen/boy
0805: baseball/swinging
0806: street/sign
0807: boat/water
0808: refrigerator
0809: dog/motorcycle
0810: horses/beach
0811: street/traffic
0812: food/truck
0813: giraffe/zebra
0814: elephant/water
0815: laptop/bed
0816: giraffes
0817: man/food/kitchen
0818: kite/flying
0819: horse
0820: playing/frisbee
0821: parked/car
0822: donut/holding
0823: skis/down
0824: horses/riding
0825: giraffe/fence
0826: bear/teddy
0827: meat/vegetables
0828: basketball
0829: knife/cutting
0830: soccer
0831: frisbee
0832: horses
0833: computer/man
0834: bear/black
0835: sign/street
0836: skiing/down
0837: bedroom
0838: playing/Wii
0839: woman/kitchen
0840: sign/street
0841: skateboard
0842: wine/glasses
0843: flying/airplane
0844: bathroom/tub
0845: train
0846: plate/fork
0847: food/people/table
0848: baseball
0849: motorcycles
0850: motorcycle
0851: street/building
0852: table/man/woman
0853: baby/teddy/bear
0854: dog/frisbee
0855: tennis/woman
0856: baseball
0857: plate/broccoli
0858: truck
0859: polar/bears
0860: parking/meter
0861: horse
0862: wine/bar
0863: plate/food
0864: beach
0865: cow
0866: man/frisbee
0867: dog/woman
0868: motorcycle
0869: cat/bear/dog
0870: room/couch/table
0871: plate/hot/dogs
0872: elephant
0873: skateboard
0874: toilet/paper
0875: airplane
0876: woman/donut
0877: tie/man/shirt
0878: dog/cat/suitcase
0879: snow/skis
0880: skateboard
0881: church/clock
0882: water/boats
0883: people/walking
0884: kites/flying
0885: cat/remote
0886: toilet
0887: beach/kite
0888: birds/tree
0889: flying/sky
0890: bathroom/curtain
0891: bird/branch
0892: bus/yellow
0893: zebra/road
0894: street/night
0895: planes/flying
0896: cat
0897: snow/street
0898: bench/flowers
0899: plate/vegetables
0900: kite
0901: beach/surfboard
0902: sandwich
0903: sign/street
0904: cat/woman
0905: street/traffic
0906: phone/talking
0907: kitchen
0908: bat/baseball/boy
0909: bear/teddy
0910: men/standing
0911: cow/street
0912: luggage/people
0913: runway/airplane
0914: tennis
0915: skateboard
0916: toilet/cat
0917: bus
0918: sitting/bench
0919: vase/flowers
0920: kitchen/preparing
0921: traffic/lights
0922: clock/building
0923: hot/dog
0924: bus/street/city
0925: soccer
0926: beach/boats
0927: hydrant/fire
0928: zebra
0929: beach/people
0930: soccer
0931: kite/girl
0932: train/tracks
0933: man/cat/dog
0934: dog/beach
0935: giraffes
0936: bathroom
0937: room/chairs
0938: snowboarding
0939: birds/pigeons
0940: cake/birthday
0941: baseball
0942: pizza
0943: fruit/apples
0944: horses
0945: zebras
0946: flying/jets
0947: people/skis
0948: bird/tree
0949: umbrella/girl
0950: street/people
0951: women/umbrella
0952: game/remote
0953: zebras
0954: water/board
0955: street/signs
0956: parking/meter
0957: skateboard
0958: train/snow
0959: hydrant/fire
0960: train/model/toy
0961: horse
0962: truck
0963: banana/woman
0964: sheep/man/woman
0965: crowd/street
0966: people/park
0967: walking/standing
0968: pizza/cheese
0969: motorcycles
0970: train/subway
0971: bus/double/decker
0972: room/fireplace
0973: beach/umbrella/
0974: zebra
0975: banana
0976: street/sign
0977: toilets
0978: truck
0979: sign/traffic
0980: tennis/woman
0981: wave/surfboard
0982: train/tracks
0983: train/tracks
0984: bench/man
0985: bed/desk
0986: snow/skis
0987: man/playing/game
0988: box/donuts
0989: ramp/skateboard
0990: ski/lift
0991: giraffe
0992: woman/skis
0993: toilet
0994: train
0995: surfboard
0996: elephant
0997: pizzas
0998: window/cats
0999: sheep/grazing

Figure 3: 1000 clusters trained with MS-COCO. For each cluster, we depict a few most frequent keywords in the captions that belong to the cluster.

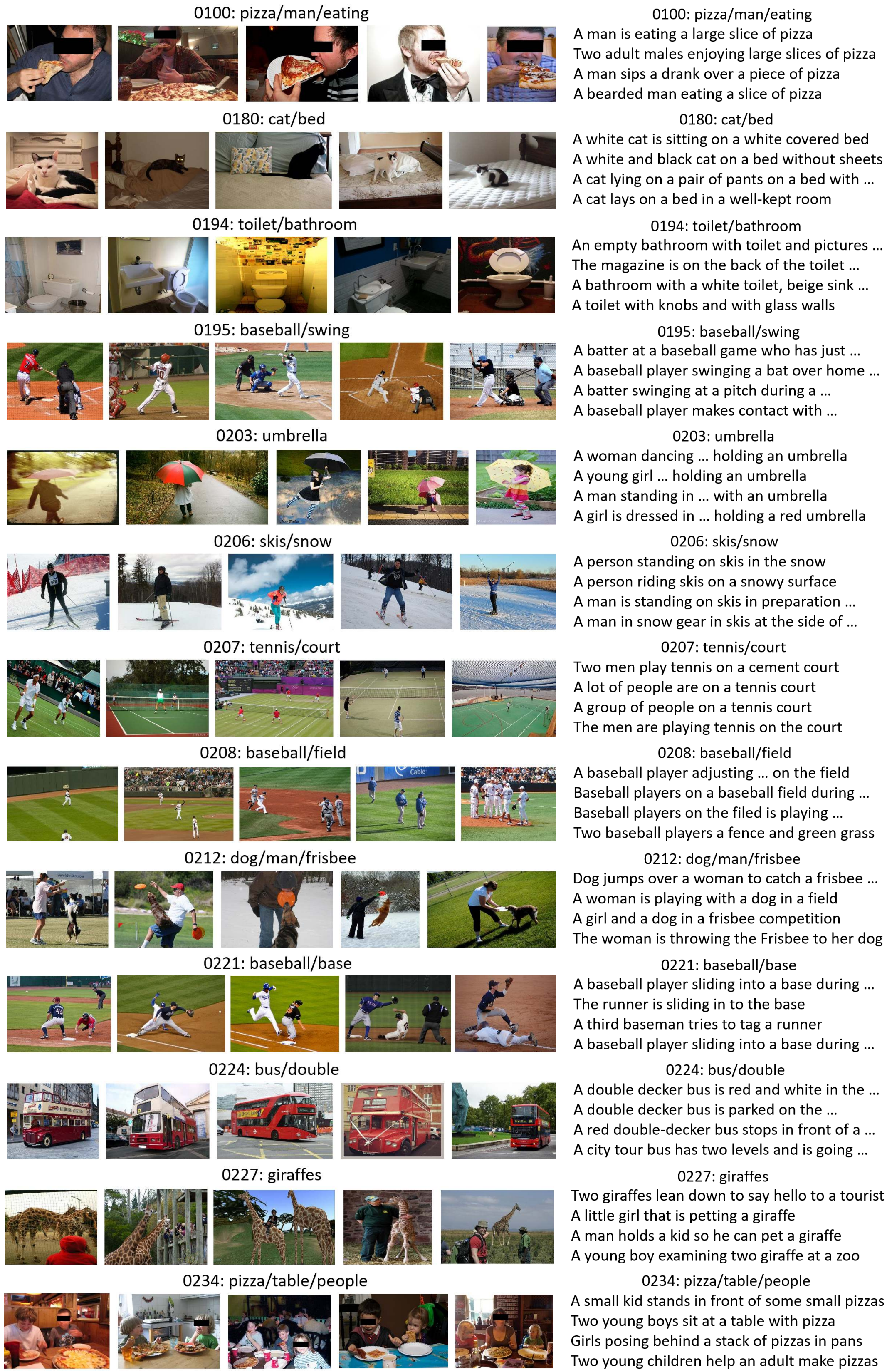
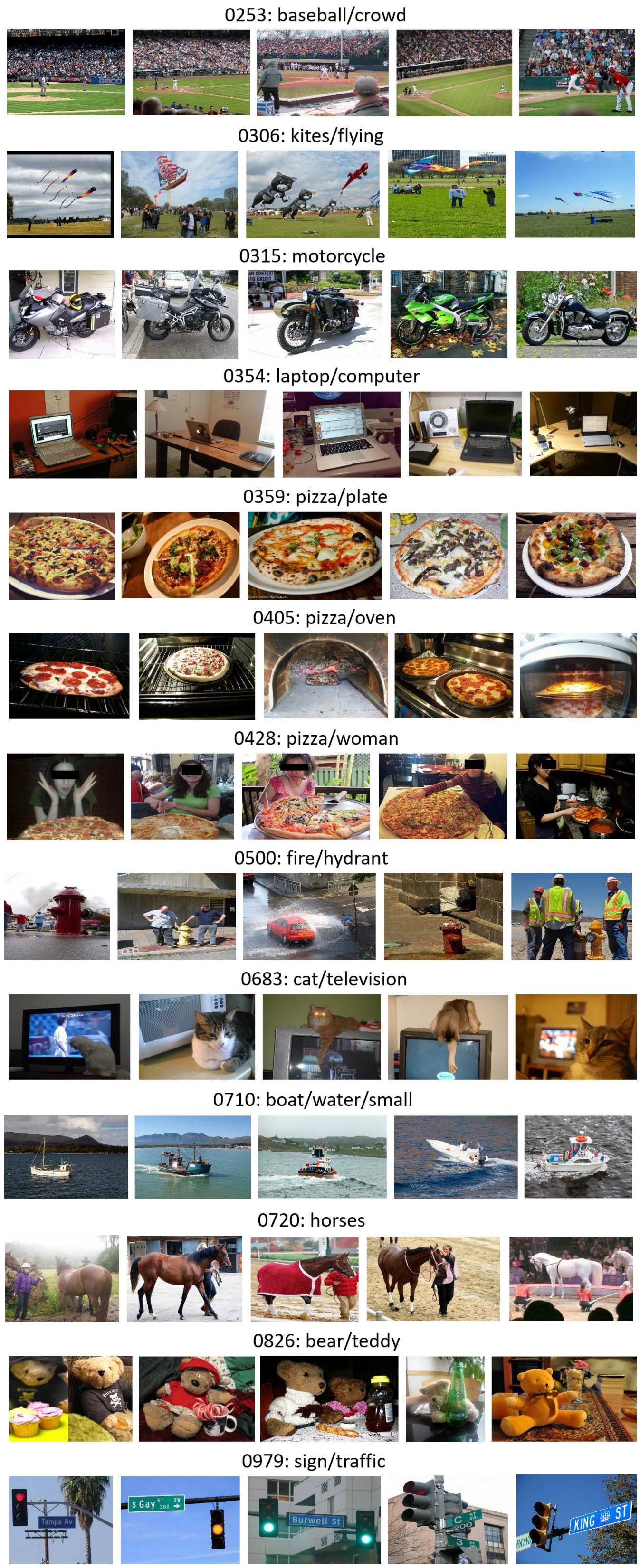


Figure 4: Some randomly selected clusters with images and texts that belong to them. Each cluster, titled by *ID: keywords*, shows randomly chosen 5 images and 4 texts.



0253: baseball/crowd
A large crowd is watching a baseball game
Fans sitting and watching a batter at a ...
A baseball game in progress at a huge stadium
A crowd is watching as a baseball game is ...

0306: kites/flying
Large air balloon that looks like a face laying ...
A group of people flying kites on a grass field
Kites being flown in a park on a sunny day
A group of people flying kits over a green ...

0315: motorcycle
A motorcycle that is parked in a field
A bike and a motorcycle parked near each ...
A motorcycle with several dead chickens ...
A yellow motorcycle parked on the curb ...

0354: laptop/computer
A nice looking table with a laptop on it
An open laptop computer sitting on a ...
The laptop was left open on the desk ...
A laptop computer sitting on top of a desk

0359: pizza/plate
A cooked whole pizza with various toppings ...
A personal size pizza with bacon on a table
A closeup of a pizza on a plate
A homemade pizza with toppings served on ...

0405: pizza/oven
A frozen pizza in the oven already cooked
A view of a pizza cooking in an oven
A pizza is being cooked on the stove
A brick oven with a pan of pizza baking in it

0428: pizza/woman
A woman holds her stomach in front of pizza
A woman cutting a pizza with a knife
A woman eating pizza in a crowded outdoor ...
A lady using a knife and fork to cut a pizza

0500: fire/hydrant
A fireman working on a fire hydrant near ...
A man is painting a fire hydrant red ...
A uniformed man painting a fire hydrant red ...
Fire hydrant is locked up for only firemen ...

0683: cat/television
A grey and white cat sits on a cable box
A black cat sitting on top of a television
A cat is watching other cats on a television
A cat stares at a television from close up

0710: boat/water/small
An empty boat in the water near a tree
A boat that is sitting in the water
A fishing boat moving through the water of ...
A boat is running in the water with a low sun ...

0720: horses
A man walking a horse down a street next to ...
Two women with horses at a stable
A group of people walking horses down a ...
The horse is getting ready for the show

0826: bear/teddy
A teddy bear cake for a first birthday
A stuffed bear sitting on a mason jar near ...
There is a stuffed bear sitting on jars
A teddy bear and a vase of flowers sit on a ...

0979: sign/traffic
A black and white image of a streetlight
A traffic light and a street sign are mounted ...
Two street signs and traffic lights in ...
Two street signs attached to a light post

Figure 5: More clusters (continued from Fig. 4).

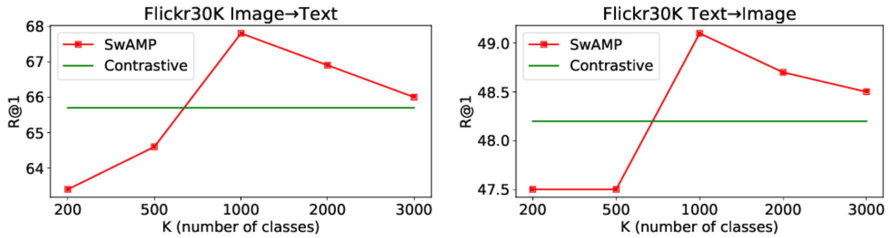


Figure 6: (Flickr30K) Impact of the number of classes (K).

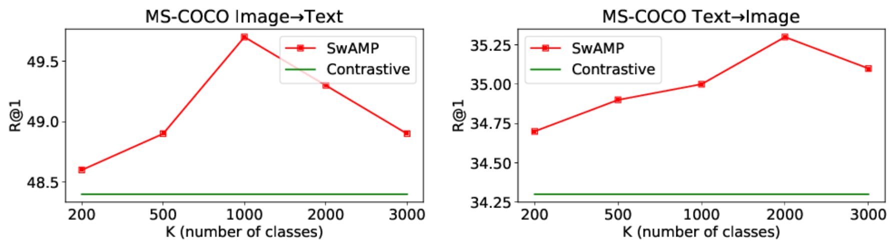


Figure 7: (MS-COCO) Impact of the number of classes (K).

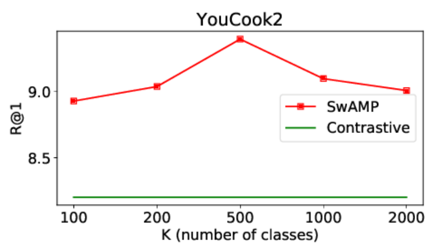


Figure 8: (YouCook2) Impact of the number of classes (K).

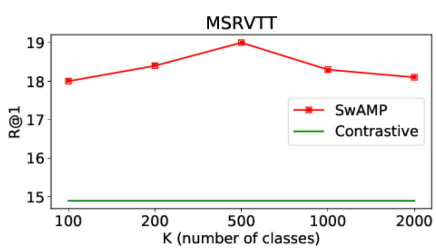


Figure 9: (MSRVTT) Impact of the number of classes (K).

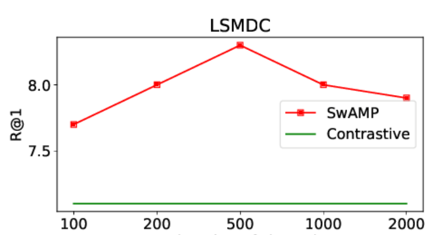


Figure 10: (LSMDC) Impact of the number of classes (K).

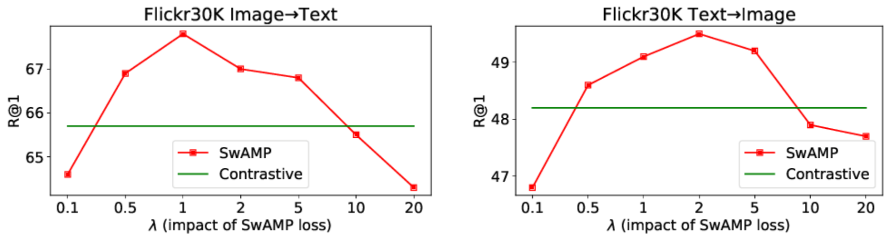


Figure 11: (Flickr30K) Impact of the SwAMP loss (λ).

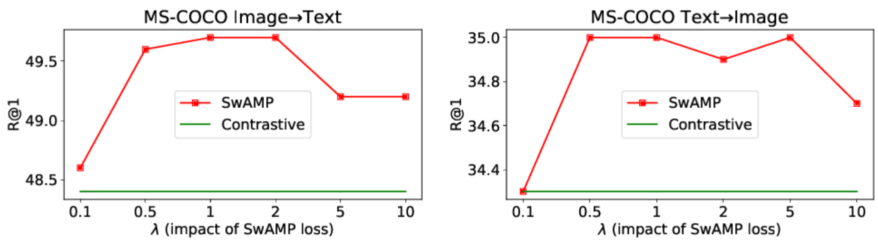


Figure 12: (MS-COCO) Impact of the SwAMP loss (λ).

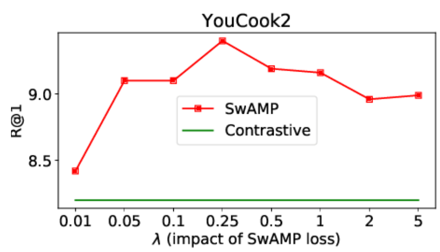


Figure 13: (YouCook2) Impact of the SwAMP loss (λ).

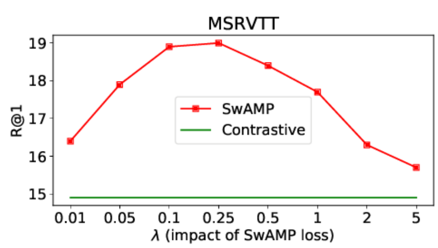


Figure 14: (MSRVTT) Impact of the SwAMP loss (λ).

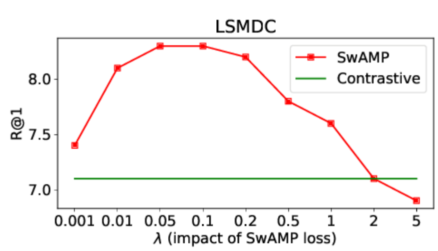


Figure 15: (LSMDC) Impact of the SwAMP loss (λ).

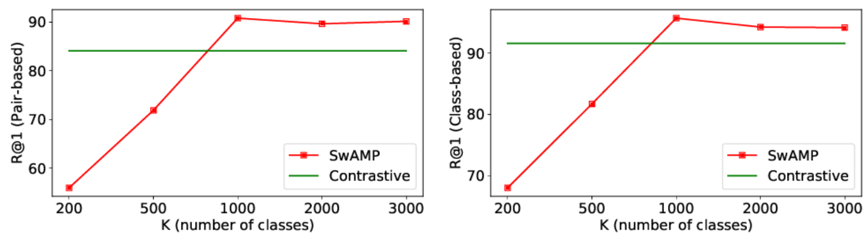


Figure 16: (Synthetic data) Impact of the number of classes (K).

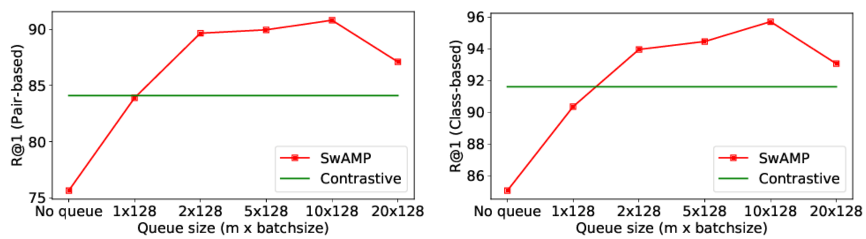


Figure 17: (Synthetic data) Impact of the size of the queues.

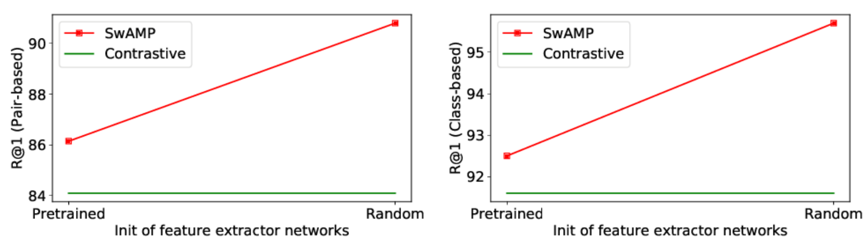


Figure 18: (Synthetic data) Impact of the initialization of feature extractor networks.

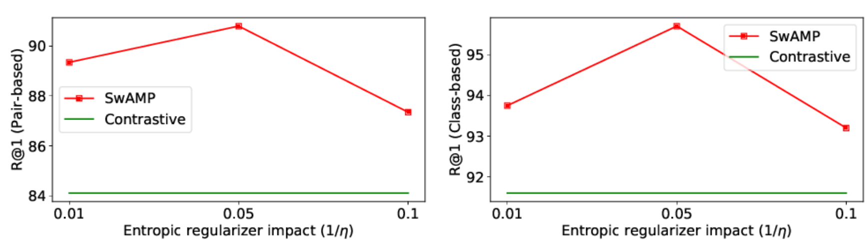


Figure 19: (Synthetic data) Impact of entropic regularization ($1/\eta$) in Sinkhorn-Knopp.

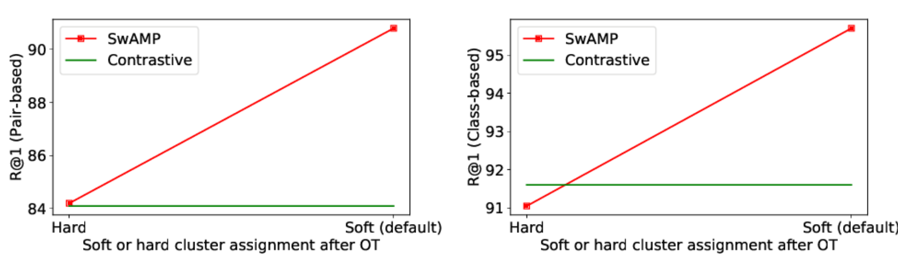


Figure 20: (Synthetic data) Soft (default) or hard cluster assignment after OT.