Do Lessons from Metric Learning Generalize to Image-Caption Retrieval?

Maurits Bleeker and Maarten de Rijke

University of Amsterdam, Amsterdam, The Netherlands
{m.j.r.bleeker, m.derijke}@uva.nl

Abstract. The triplet loss with semi-hard negatives has become the de facto choice for image-caption retrieval (ICR) methods that are optimized from scratch. Recent progress in metric learning has given rise to new loss functions that outperform the triplet loss on tasks such as image retrieval and representation learning. We ask whether these findings generalize to the setting of ICR by comparing three loss functions on two ICR methods. We answer this question negatively: the triplet loss with semi-hard negative mining still outperforms newly introduced loss functions from metric learning on the ICR task. To gain a better understanding of these outcomes, we introduce an analysis method to compare loss functions by counting how many samples contribute to the gradient w.r.t. the query representation during optimization. We find that loss functions that result in lower evaluation scores on the ICR task, in general, take too many (non-informative) samples into account when computing a gradient w.r.t. the query representation, which results in sub-optimal performance. The triplet loss with semi-hard negatives is shown to outperform the other loss functions, as it only takes one (hard) negative into account when computing the gradient.

1 Introduction

Given a query item in one modality, *cross-modal retrieval* is the task of retrieving similar items in another modality [43]. We focus on *image-caption retrieval* (ICR) [11, 24, 25, 40]. For the ICR task, given an image or a caption as a query, systems have to retrieve the positive (e.g., matching or similar) item(s) in the other modality. Most ICR methods work with a separate encoder for each modality to map the input data to a representation in a shared latent space [11, 12, 16, 24, 25]. The encoders are optimized by using a contrastive-loss criterion, so as to enforce a high degree of similarity between representations of matching items in the latent space. For retrieval, a similarity score between a query and each candidate in a candidate set is computed to produce a ranking with the top-k best matching items. A lot of recent work on ICR relies on (1) pre-training on large amounts of data [16, 26, 35], and (2) more sophisticated (and data-hungry) model architectures [5, 11, 12, 24, 25, 31]. However, pre-training on large-scale datasets is not always an option, either due to a lack of compute power, a lack of data, or both. Hence, it is important to continue to develop effective ICR methods that only rely on a modest amount of data.

To learn the similarity between a query and candidate representations, most ICR work relies on the standard Triplet loss with semi-hard negatives (Triplet SH) [4, 5, 11, 12, 24, 25, 31] or on the cross-entropy based NT-Xent [6, 16] loss.

In *metric learning*, the focus is on loss functions that result in more accurate item representations (in terms of a given evaluation metric) that can distinguish between similar and dissimilar items in a low-dimensional latent space [32]. There has been important progress in metric learning, with the introduction of new loss functions that result in better evaluation scores on a specific (evaluation) task. For example SmoothAP [1], it is a smooth approximation of the discrete evaluation metric Average Precision. By using SmoothAP, a retrieval method can be optimized with a discrete ranking evaluation metric and can handle multiple positive candidates simultaneously, which is not possible for the standard Triplet loss. Loss functions such as SmoothAP narrow the gap between the training setting and a discrete evaluation objective and thereby improve evaluation scores.

Research goal. Most metric learning functions work with general representations of similar/dissimilar candidates and, in principle, there is no clear argument why obtained results on a specific task/method should not generalize to other tasks or methods. Hence, *can newly introduced metric learning approaches, that is, alternative loss functions, be used to increase the performance of ICR methods*? We compare three loss function for the ICR task: (1) the Triplet loss [22], including semi-hard negative mining, (2) NT-Xent loss [7], and (3) SmoothAP [1]. We expect SmoothAP to result in the highest performance based on the findings in in the context of image retrieval [1] and in representation learning [38].

Main findings. Following [32], we evaluate the three loss functions on fixed methods, with different datasets, and with a fixed training regime (i.e., training hyper-parameters) to verify which loss function uses the given training data as effectively as possible. Surprisingly, the lessons from metric learning do not generalize to ICR. The Triplet loss with semi-hard negative mining still outperforms the other loss functions that we consider. The promising results obtained by SmoothAP and the NT-Xent loss in other fields do not generalize to the ICR task.

To get a better grasp of this unexpected outcome, we propose *counting contributing samples* (COCOS), a method for analyzing contrastive loss functions. The gradient w.r.t. the query for the Triplet loss, NT-Xent and SmoothAP can be formulated as a sum over the representations of the positive and negative candidates in the training batch. The main difference between the loss functions lies in the number of samples used when computing the gradient w.r.t. the query and how each sample is weighted. We compare loss functions by counting how many samples contribute to the gradient w.r.t. the query representation at their convergence points. This yields an explanation of why one loss function outperforms another on the ICR task.

Main contributions. (1) We experimentally compare three loss functions from the metric learning domain to determine if promising results from metric learning generalize to the ICR task, and find that the Triplet loss semi-hard (SH) still results in the highest evaluation scores. (2) We propose COCOS, a way of analyzing contrastive loss functions, by defining a count that tells us how many candidates in the batch contribute to the gradient w.r.t. the query. On average, the best performing loss function takes at most one (semi-hard) negative sample into account when computing the gradient.

2 Background and Related Work

Notation. We follow the notation introduced in [1, 7, 38]. We start with a multi-modal image-caption dataset $\mathcal{D}=\{(\mathbf{x}_I^i,\mathbf{x}_{C_1}^i,\ldots,\mathbf{x}_{C_k}^i)^i,\ldots\}_{i=1}^N$ that contains N image-caption tuples. For each image $\mathbf{x}_I^i$, we have k matching/corresponding captions, $\mathbf{x}_{C_1}^i,\ldots,\mathbf{x}_{C_k}^i$.

In the ICR task, either an image or a caption can function as a query. Given a query $\mathbf{q}$, the task is to rank all candidates in a candidate set $\Omega=\{\mathbf{v}_i \mid i=0,\ldots,m\}$. A matching candidate is denote as $\mathbf{v}^+$ and a negative candidate(s) as $\mathbf{v}^-$. For each query $\mathbf{q}$, we can split the candidate set Ω into two disjoint subsets: $\mathbf{v}^+ \in \mathcal{P}_\mathbf{q}$ (*positive* candidate set) and $\mathbf{v}^- \in \mathcal{N}_\mathbf{q}$ (*negative* candidate set), where $\mathcal{N}_\mathbf{q}=\{\mathbf{v}^- \mid \mathbf{v}^- \in \Omega, \mathbf{v}^- \notin \mathcal{P}_\mathbf{q}\}$. We assume a binary match between images and captions, they either match or they do not match.

The set with similarity scores for each $\mathbf{v}_i \in \Omega$ w.r.t. query $\mathbf{q}$ is defined as: $\mathcal{S}_\Omega^\mathbf{q} = \{s_i = \langle \frac{\mathbf{q}}{\|\mathbf{q}\|} \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|} \rangle, i = 0,\ldots,m\}$. We use cosine similarity as a similarity scoring function. $S_\Omega^\mathbf{q}$ consists of two disjoint subsets: $S_\mathcal{P}^\mathbf{q}$ and $S_\mathcal{N}^\mathbf{q}$. $S_\mathcal{P}^\mathbf{q}$ contains the similarity scores for the positive candidates and $S_\mathcal{N}^\mathbf{q}$ the similarity scores for the negative candidates. During training, we randomly sample a batch $\mathcal{B}$ with image-caption pairs. Both the images and captions will functions as queries and candidates.

Image-caption retrieval. The ICR task can be divided into *image-to text* (i2t) and *text-to-image* (t2i) retrieval. We target specific ICR methods that are optimized for the ICR-task only and satisfy three criteria: (1) The methods we use have solely been trained and evaluated on the same benchmark dataset; (2) the ICR methods we use compute one global representation for both the image and caption; and (3) the methods do not require additional supervision signals besides the contrastive loss for optimization. Below we evaluate two ICR methods with different loss functions: VSE++ [12] and VSRN [25]. In the online appendix of this work,[1] we provide a detailed description of VSE++ and VSRN.

VSE++. The best performing method of VSE++ uses a ResNet-152 [15]) to compute a global image representation. The caption encoder is a single directed GRU-based [10] encoder. Faghri et al. [12] introduce the notion of mining semi hard-negative triplets for the ICR task. By using the hardest negative in the batch for each positive pair (i.e. the negative candidate with the highest similarity score w.r.t. the query), their method outperforms state-of-the-art methods that do not apply this semi-hard negative mining.

VSRN. VSRN takes a set of pre-computed image region features as input. A Graph Convolutional Network [21] is used to enhance the relationships between each region vector. The sequence of region feature vectors is put through an RNN network to encode the global image representation. VSRN uses the same caption encoder and loss as [12].

Other methods. Following VSE++ and VSRN, the SGRAF [11] and IMRAM [4] methods have been introduced. We do not use these two methods as they either do not outperform VSRN [4] or rely on similar principles as VSRN [11]. The main recent progress in ICR has been characterized by a shift towards transformer-based [39] methods. To the best of our knowledge, TREN/TERAN [30, 31] and VisualSparta [29] are the only transformer-based ICR methods that are solely optimized using MS-COCO [27] or Flickr30k [42]. We do not use transformer-based methods, as optimizing them does not scale well for a reproduciblity study with moderately sized datasets. Methods

[1] https://github.com/MauritsBleeker/ecir-2022-reproducibility-bleeker/blob/master/appendix

such as OSCAR [26], UNITER [9], Vilbert [28] and ViLT-B [20] use additional data sources and/or loss functions for training. They focus on a wide variety of tasks such as visual QA, image captioning, and image retrieval.

Loss functions for ICR. In this section we introduce three loss functions for ICR.

Triplet loss with semi hard-negative mining. The Triplet loss is commonly used as a loss function for ICR methods [5, 11, 12, 24, 25, 31]. The *Triplet loss with semi-hard negative mining* (Triplet loss SH), for a query $\mathbf{q}$ is defined as:

$$\mathcal{L}^{\mathbf{q}}_{TripletSH} = \max(\alpha - s^+ + s^-, 0), \tag{1}$$

where α is a margin parameter, $s^- = \max(S^{\mathbf{q}}_{\mathcal{N}})$ and $s^+ = s_0 \in S^{\mathbf{q}}_{\mathcal{P}}$. Here, $\mathcal{S}^{\mathbf{q}}_{\mathcal{P}}$ only contains one element per query. The Triplet loss SH over the entire training batch is defined as:

$$\mathcal{L}_{TripletSH} = \sum_{\mathbf{q} \in \mathcal{B}} \mathcal{L}^{\mathbf{q}}_{TripletSH}. \tag{2}$$

Triplet loss SH performs a form of soft-negative mining per query by selecting the negative candidate with the highest similarity score w.r.t. the query, we also refer to this as the maximum violating query. For computational efficiency, this soft-negative mining is executed within the context of the training batch $\mathcal{B}$ and not over the entire training set.

As opposed to the definition above, another possibility is to take the Triplet-loss over all triplets in the batch $\mathcal{B}$. This is the definition of the standard *Triplet-loss* [22]:

$$\mathcal{L}^{\mathbf{q}}_{Triplet} = \sum_{s^- \in S^q_{\mathcal{N}}} \max(\alpha - s^+ + s^-, 0) \tag{3a}$$

$$\mathcal{L}_{Triplet} = \sum_{\mathbf{q} \in \mathcal{B}} \mathcal{L}^{\mathbf{q}}_{Triplet}. \tag{3b}$$

NT-Xent loss. The *NT-Xent loss* [7] is a loss function commonly used in the field of self-supervised representation learning [7, 33]. A similar function has also been proposed by Zhang and Lu [45] in the context of ICR. The NT-Xent loss is defined as:

$$\mathcal{L}_{NT\text{-}Xent} = -\frac{1}{|\mathcal{B}|} \sum_{\mathbf{q} \in \mathcal{B}} \log \frac{\exp(s^+/\tau)}{\sum_{s_i \in \mathcal{S}^{\mathbf{q}}_{\Omega}} \exp(s_i/\tau)}, \tag{4}$$

where τ functions as a temperature parameter. As for the Triplet-loss formulation: $s^+ = s_0 \in S^{\mathbf{q}}_{\mathcal{P}}$. The major difference between the Triplet-loss SH is that the NT-Xent loss takes the entire negative candidate set into account.

SmoothAP loss. The Average Precision metric w.r.t. a query $\mathbf{q}$ and candidate set Ω is defined as:

$$AP_{\mathbf{q}} = \frac{1}{|\mathcal{S}^{\mathbf{q}}_{\mathcal{P}}|} \sum_{i \in \mathcal{S}^{\mathbf{q}}_{\mathcal{P}}} \frac{\mathcal{R}(i, \mathcal{S}^{\mathbf{q}}_{\mathcal{P}})}{\mathcal{R}(i, \mathcal{S}^{\mathbf{q}}_{\Omega})}, \tag{5}$$

where $\mathcal{R}(i, \mathcal{S})$ is a function that returns the ranking of candidate $i \in \mathcal{S}$ in the candidate set:

$$\mathcal{R}(i, \mathcal{S}) = 1 + \sum_{j \in \mathcal{S}, i \neq j} \mathbb{1}\{s_i - s_j < 0\}. \tag{6}$$

Let us introduce the $M \times M$ matrix D, where $D_{ij} = s_i - s_j$. By using the matrix D, Eq. 5 can be written as:

$$AP_{\mathbf{q}} = \frac{1}{|\mathcal{S}^{\mathbf{q}}_{\mathcal{P}}|} \sum_{i \in \mathcal{S}^{\mathbf{q}}_{\mathcal{P}}} \frac{1 + \sum_{j \in \mathcal{S}_{\mathcal{P}}, j \neq i} \mathbb{1}\{D_{ij} > 0\}}{1 + \sum_{j \in \mathcal{S}^{\mathbf{q}}_{\mathcal{P}}, j \neq i} \mathbb{1}\{D_{ij} > 0\} + \sum_{j \in \mathcal{S}^{\mathbf{q}}_{\mathcal{N}}} \mathbb{1}\{D_{ij} > 0\}}.$$

The indicator function $\mathbb{1}\{\cdot\}$ is non-differentiable. To overcome this problem, the indicator function can be replaced by a sigmoid function:

$$\mathcal{G}(x; \tau) = \frac{1}{1 + e^{\frac{-x}{\tau}}}. \tag{7}$$

By replacing the indicator function $\mathbb{1}\{\cdot\}$ by $\mathcal{G}$, the Average Precision metric can be approximated with a smooth function:

$$AP_{\mathbf{q}} \approx \frac{1}{|\mathcal{S}_{\mathcal{P}}^{q}|} \sum_{i \in \mathcal{S}_{\mathcal{P}}^{\mathbf{q}}} \frac{1+\sum_{j \in \mathcal{S}_{\mathcal{P}}^{\mathbf{q}}, j \neq i} \mathcal{G}(D_{ij};\tau)}{1+\sum_{j \in \mathcal{S}_{\mathcal{P}}^{\mathbf{q}}, j \neq i} \mathcal{G}(D_{ij};\tau)+\sum_{j \in \mathcal{S}_{\mathcal{N}}^{\mathbf{q}}} \mathcal{G}(D_{ij};\tau)}.$$

This loss function is called *SmoothAP* and has been introduced in the context of image retrieval [1], following similar proposals in document retrieval and learning to rank [2, 3, 34, 41]. The total loss over a batch $\mathcal{B}$ can then be formulated as follows:

$$\mathcal{L}_{AP} = \frac{1}{|\mathcal{B}|} \sum_{\mathbf{q} \in \mathcal{B}} (1 - AP_{\mathbf{q}}). \tag{8}$$

In the online appendix,[1] we provide an extended explanation of SmoothAP.

3 Do Findings from Metric Learning Extend to ICR?

In representation learning it was found that NT-Xent loss outperforms the Triplet loss and Triplet loss SH [7]. For both the image retrieval and representation learning task, results show that SmoothAP outperforms both the Triplet loss SH and the NT-Xent loss [1, 38]. We examine whether these findings generalize to ICR.

Experimental setup. We focus on two benchmark datasets for the ICR task: the Flickr30k [42] and MS-COCO [27] datasets. Similar to [12, 25], we use the split provided by Karpathy and Fei-Fei [17] for MS-COCO and the Flickr30k. For details of the specific implementations of VSE++ [12][2] and [25][3] we refer to the papers and online implementations. Each method is trained for 30 epochs with a batch size of 128. We start with a learning rate of 0.0002 and after 15 epochs we lower the learning rate to 0.00002.

For VSE++, we do not apply additional fine-tuning of the image encoder after 30 epochs. Our main goal is to have a fair comparison across methods, datasets, and loss functions, not to have the highest overall evaluation scores. For VSE++, we use ResNet50 [15] as image-encoder instead of ResNet152 [15] or VGG [37]. ResNet50 is faster to optimize and the performance differences between ResNet50 and ResNet152 are relatively small.

The VSRN method comes with an additional caption decoder, to decode the original input caption from the latent image representation, this to add additional supervision to the optimization process. We remove the additional image-captioning module, so as to exclude performance gains on the retrieval tasks due to this extra supervision. In [25], the similarity score for a query candidate pair, during evaluation, is based on averaging the predicted similarity scores of (an ensemble of) two trained models. We only take the predicted relevance score of one model. The reason for this is that the evaluation score improvements are marginal when using the scores of two models (instead of one) but optimizing the methods takes twice as long. Therefore, our results are lower than the results published in [25]. For all the remaining details, we refer to our repository.[4] When optimizing with SmoothAP, we take all the k captions into account when sampling a batch, instead of one positive candidate. For this reason, we have to increase the amount of training epochs k times as well to have a fair comparison. For each loss function, we select the best performing hyper-parameter according to its original work.

Experiments. We evaluate each loss function we described in Section 2 given a dataset and method. For ease of reference, we refer to each individual evaluation with an

[2] https://github.com/fartashf/vsepp [3] https://github.com/KunpengLi1994/VSRN
[4] https://github.com/MauritsBleeker/ecir-2022-reproducibility-bleeker

Table 1: Evaluation scores for the Flickr30k and MS-COCO, for the VSE++ and VSRN.

Loss function	#	hyper param	i2t R@1	i2t R@5	i2t R@10	i2t average recall	i2t mAP@5	t2i R@1	t2i R@5	t2i R@10	t2i average recall	rsum
Flickr30k												
VSE++												
Triplet loss	1.1	α=0.2	30.8±.7	62.6±.3	74.1±.8	55.9±.3	0.41±.00	23.4±.3	52.8±.1	65.7±.3	47.3±.1	309.4±0.9
Triplet loss SH	1.2	α=0.2	**42.4**±.5	**71.2**±.7	**80.7**±.7	64.8±.6	0.50±.01	**30.0**±.3	**59.0**±.2	**70.4**±.4	53.1±.2	**353.8**±1.6
NT-Xent	1.3	τ=0.1	37.5±.6	68.4±.6	77.8±.5	61.2±.3	0.47±.00	27.0±.3	57.3±.3	69.1±.2	51.1±.2	337.1±1.3
SmoothAP	1.4	τ=0.01	**42.1**±.8	**70.8**±.6	**80.6**±.8	64.5±.4	0.50±.00	29.1±.3	58.1±.1	69.7±.2	52.3±.2	350.4±1.7
VSRN												
Triplet loss	1.5	α=0.2	56.4±.7	83.6±.6	90.1±.2	76.7±.5	0.63±.01	43.1±.3	74.4±.3	83.1±.4	66.9±.3	430.7±1.8
Triplet loss SH	1.6	α=0.2	**68.3**±1.3	**89.6**±.7	**94.0**±.5	84.0±.5	0.73±.01	**51.2**±.9	**78.0**±.6	**85.6**±.5	71.6±.6	**466.6**±3.3
NT-Xent	1.7	τ=0.1	50.9 ±.5	78.9±.7	86.6±.4	72.2±.4	0.59±.00	40.6±.6	71.9±.2	81.7±.3	64.7±.2	410.6±1.5
SmoothAP	1.8	τ=0.01	63.1±1.0	86.6±.8	92.4±.5	80.7±.7	0.69±.00	45.8±.2	73.7±.3	82.3±.2	67.3±.1	444.0±2.1
MS-COCO												
VSE++												
Triplet loss	2.1	α=0.2	22.1±.5	48.2±.3	61.7±.3	44.0±.3	0.30±.00	15.4±.1	39.5±.1	53.2±.1	36.0±.1	240.0±0.9
Triplet loss SH	2.2	α=0.2	**32.5**±.2	**61.6**±.3	**73.8**±.3	56.0±.2	0.41±.00	**21.3**±.1	**48.1**±.1	**61.5**±.0	43.6±.1	**298.8**±0.8
NT-Xent	2.3	τ=0.1	25.8±.5	53.6±.5	66.1±.2	48.5±.3	0.34±.00	18.0±.1	43.0±.1	56.6±.2	39.2±.1	263.0±0.9
SmoothAP	2.4	τ=0.01	30.8±.3	60.3±.2	**73.6**±.5	54.9±.3	0.40±.00	20.3±.2	46.5±.2	60.1±.2	42.3±.2	291.5±1.4
VSRN												
Triplet loss	2.5	α=0.2	42.9±.4	74.3±.3	84.9±.4	67.4±.3	0.52±.00	33.5±.1	65.1±.1	77.1±.2	58.6±.1	377.8±1.2
Triplet loss SH	2.6	α=0.2	**48.9**±.6	**78.1**±.5	**87.4**±.2	71.4±.4	0.57±.01	**37.8**±.5	**68.1**±.5	**78.9**±.3	61.6±.4	**399.0**±2.3
NT-Xent	2.7	τ=0.1	37.9±.4	69.2±.2	80.7±.3	62.6±.1	0.47±.00	29.5±.1	61.0±.2	74.0±.2	54.6±.1	352.3±0.5
SmoothAP	2.8	τ=0.01	46.0±.6	76.1±.3	85.9±.3	69.4±.3	0.54±.00	33.8±.3	64.1±.1	76.0±.2	58.0±.2	382.0±1.1

experiment number (#) (see Table 1). To reduce the variance in the results we run each experiment five times and report the average score and standard deviation. Similar to [12, 25], we evaluate using Recall@k with k={1,5,10}, for both the image-to text (i2t) and text-to-image (t2i) task. We also report the sum of all the recall scores (rsum) and the average recall value. For the i2t task, we also report the mean average precision at 5 (mAP@5) due to the fact we have k positive captions per image query.

Results. Based on the scores reported in Table 1, we have the following observations:

(1) Given a fixed method and default hyper-parameters for each loss function, the Triplet loss SH results in the best evaluation scores, regardless of dataset, method or task.
(2) Similar to [12], we find that the Triplet loss SH consistently outperforms the general Triplet loss, which takes all the negative triplets in the batch into account that violate the margin constraint.
(3) The NT-Xent loss consistently underperforms compared to the Triplet loss SH. This is in contrast with findings in [7], where the NT-Xent loss results in better downstream evaluation performance on a (augmented image-to-image) representation learning task than the Triplet loss SH. Although the ICR task has different (input) data modalities, the underlying learning object is the same for ICR and augmented image-to-image representation learning (i.e., contrasting positive and negative pairs).
(4) Only for the VSE++ method on the i2t task, SmoothAP performs similar to the Triplet-loss SH.
(5) SmoothAP does not outperform the Triplet loss SH. This is in contrast with the findings in [1], where SmoothAP does outperform Triplet-loss SH and other metric learning functions.
(6) The method with the best Recall@k score also has the highest mAP@k score.

Upshot. Based on our observations concerning Table 1, we conclude the following: (1) The Triplet loss SH should still be the *de facto* choice for optimizing ICR methods. (2) The promising results from the representation learning field that were obtained by using the NT-Xent loss [7], do not generalize to the ICR task. (3) Optimizing an ICR method with a smooth approximation of a ranking metric (SmoothAP) does not result in better Recall@k scores. (4) Optimizing an ICR method by using a pair-wise distance loss between the positive triplet and a semi-hard negative triplet still yields the best evaluation performance. For both methods VSE++ and VSRN, i2t and t2i and for both datasets.

4 A Method for Analyzing the Behavior of Loss Functions

Next, we propose a method for analyzing the behavior of loss functions for ICR. The purpose is to compare loss functions, and explain the difference in performance. If we compare the gradient w.r.t. $\mathbf{q}$ for the Triplet loss and the Triplet loss SH, the only difference is the number of triplets that the two loss functions take into account. If two models are optimized in exactly the same manner, except one model uses the Triplet loss and the other uses Triplet loss SH, the difference in performance can only be explained by the fact that the Triplet loss takes all violating triplets into account. This means that the number of triplets (i.e., candidates) that contribute to the gradient directly relates to the evaluation performance of the model. The same reasoning applies for the NT-Xent and the SmoothAP loss. For example, the gradient w.r.t. $\mathbf{q}$ for the NT-Xent loss also has the form $\mathbf{v}^+ - \mathbf{v}^-$. The major difference between the two functions is that, for the negative candidate the NT-Xent loss computes a weighted sum over all negative to compute a representation of $\mathbf{v}^-$. Therefore, the difference in evaluation performance between the Triplet loss SH and NT-Xent can only be explained by this weighted sum over all negatives. This sum can be turned into a count of negatives, i.e., how many negative approximately contribute to this weighted sum, which can be related to the other losses. By counting the number of candidates that contribute to the gradient, we aim to get a better understanding of why a certain loss function performs better than others. The method we propose is called *counting contributing samples* (COCOS).

First, we provide the form of the derivative of each loss function w.r.t. query $\mathbf{q}$. For each loss function the derivative is a sum over $\mathbf{v}^+ - \mathbf{v}^-$. Loss functions may weight the positive and negative candidate(s) differently, and the number of candidates or triplets that are weighted may differ across loss functions.

Triplet loss and Triplet loss SH. The gradient w.r.t. $\mathbf{q}$ for the Triplet loss SH, $\mathcal{L}^{\mathbf{q}}_{TripletSH}$ is the difference between the representation of the positive and negative candidate:

$$\frac{\partial \mathcal{L}^{\mathbf{q}}_{TripletSH}}{\partial \mathbf{q}} = \begin{cases} \mathbf{v}^+ - \mathbf{v}^-, \text{ if } s^+ - s^- < \alpha \\ 0, \qquad \text{otherwise.} \end{cases} \tag{9a}$$

$$\frac{\partial \mathcal{L}^{\mathbf{q}}_{Triplet}}{\partial \mathbf{q}} = \sum_{\mathbf{v}^- \in \mathcal{N}_{\mathbf{q}}} \mathbb{1}\{s^+ - s^- < \alpha\}(\mathbf{v}^+ - \mathbf{v}^-). \tag{9b}$$

The gradient of Triplet loss $\mathcal{L}^{\mathbf{q}}_{Triplet}$ (Eq. 9b) w.r.t. $\mathbf{q}$ has a similar form. However, there the gradient is a sum over all triplets that violate $s^+ + s^- < \alpha$, and not only the maximum violating one. Based on Eq. 9a we can see that a query $\mathbf{q}$ only has a non-zero gradient when $s^+ - s^- < \alpha$. If this is the case, the gradient always has the form $\mathbf{v}^+ - \mathbf{v}^-$, and this value is independent of the magnitude $s^+ - s^-$. For this reason,

given a batch $\mathcal{B}$, the number of queries $\mathbf{q}$ that have a non-zero gradient is defined by:

$$C^{\mathcal{B}}_{TripletSH} = \sum_{\mathbf{q}\in\mathcal{B}} \mathbb{1}\{s^+ - s^- < \alpha\}, \tag{10}$$

where $s^+ = s^0 \in \mathcal{S}^{\mathbf{q}}_{\mathcal{P}}$ and $s^- = \max(\mathcal{S}^{\mathbf{q}}_{\mathcal{N}})$. We define $C^{\mathcal{B}}_{TripletSH}$ to be *the number of queries* $\mathbf{q}$ *that have a non-zero gradient given batch* $\mathcal{B}$.

As the Triplet loss takes all the triplets into account that violate the distance margin α, we can count three things: (1) Per query $\mathbf{q}$, we can count how many triplets $\mathbf{v}^+ - \mathbf{v}^-$ contribute to the gradient of $\mathbf{q}$. We define this as $C^{\mathbf{q}}_{Triplet} = \sum_{s^-\in\mathcal{S}^{\mathbf{q}}_{\mathcal{N}}} \mathbb{1}\{s^+ - s^- < \alpha\}$. (2) Given the batch $\mathcal{B}$, we can count how many triplets contribute to the gradient over the entire training batch $\mathcal{B}$. We define this number as $C^{\mathcal{B}}_{Triplet} = \sum_{\mathbf{q}\in\mathcal{B}} C^{\mathbf{q}}_{Triplet}$. (3) Given the entire batch $\mathcal{B}$, we can count how many queries have a gradient value of zero (i.e., no violating triplets). This number is $C^0_{Triplet} = \sum_{\mathbf{q}\in\mathcal{B}} \mathbb{1}\{C^{\mathbf{q}}_{Triplet} = 0\}$.

NT-Xent loss. The gradient w.r.t. $\mathbf{q}$ for the NT-Xent loss is defined as [7]:

$$\frac{\partial \mathcal{L}^{\mathbf{q}}_{NT\text{-}Xent}}{\partial \mathbf{q}} = \left(1 - \frac{\exp(s^+/\tau)}{Z(\mathbf{q})}\right)\tau^{-1}\mathbf{v}^+ - \sum_{s^-\in S^q_{\mathcal{N}}} \left(\frac{\exp(s^-/\tau)}{Z(\mathbf{q})}\right)\tau^{-1}\mathbf{v}^-, \tag{11}$$

where $Z(\mathbf{q}) = \sum_{s_i\in\mathcal{S}^{\mathbf{q}}_{\Omega}} \exp(s_i/\tau)$, a normalization constant depending on $\mathbf{q}$. The gradient w.r.t. $\mathbf{q}$ is the weighted difference of the positive candidate $\mathbf{v}^+$ and the weighted sum over all the negative candidates. The weight for each candidate is based on the similarity with the query, normalized by the sum of the similarities of all candidates. In contrast, for the Triplet-loss (Eq. 9b) all candidates are weighted equally when they violate the margin constraint. The NT-Xent loss performs a natural form of (hard) negative weighting [7]. The more similar a negative sample is to the query, the higher the weight of this negative in the gradient computation. In principle, all the negatives and the positive candidate contribute to the gradient w.r.t. $\mathbf{q}$. In practice, most similarity scores $s^- \in \mathcal{S}^{\mathbf{q}}_{\mathcal{N}}$ have a low value; so the weight of this negative candidate in the gradient computation will be close to 0.

To count the number of negative candidates that contribute to the gradient, we define a threshold value ϵ. If the weight of a negative candidate $\mathbf{v}^-$ is below ϵ, we assume that its contribution is negligible. All candidate vectors are normalized. Hence, there is no additional weighting effect by the magnitude of the vector. For the NT-Xent loss we define three terms: $C^{\mathbf{qv}^-}_{NTXent}$, $W^{\mathbf{qv}^-}_{NTXent}$ and $W^{\mathbf{qv}^+}_{NT-Xent}$: (1) Given a query $\mathbf{q}$, $C^{\mathbf{qv}^-}_{NT-Xent}$ is the number of negative candidates $\mathbf{v}^-$ that contribute to the gradient w.r.t. $\mathbf{q}$: $C^{\mathbf{qv}^-}_{NT-Xent} = \sum_{s^-\in\mathcal{S}^{\mathbf{q}}_{\mathcal{N}}} \mathbb{1}\{\exp(s^-/\tau)Z(\mathbf{q})^{-1} > \epsilon\}$. (2) Given $C^{\mathbf{qv}^-}_{NT-Xent}$, we compute the sum of the weight values of the contributing negative canidates $\mathbf{v}^-$ as $W^{\mathbf{qv}^-}_{NT-Xent} = \sum_{s^-\in\mathcal{S}^{\mathbf{q}}_{\mathcal{N}}} \mathbb{1}\{\exp(s^-/\tau)Z(\mathbf{q})^{-1} > \epsilon\}\exp(s^-/\tau)Z(\mathbf{q})^{-1}$. (3) We define $W^{\mathbf{qv}^+}_{NT-Xent} = \frac{1}{N}\sum_{\mathbf{q}\in\mathcal{B}}(1 - \exp(s^+/\tau)Z(\mathbf{q})^{-1})$, as the mean weight value of the positive candidates in batch $\mathcal{B}$.

We define the two extra terms, $W^{\mathbf{qv}^-}_{NT-Xent}$ and $W^{\mathbf{qv}^+}_{NT-Xent}$ because for the NT-Xent function we have to count the candidates with a weight value above the threshold ϵ. This count on its own does not provide a good picture of the contribution of these candidates to the gradient. Therefore, we compute a mean value of those weight values as well, to provide insight into the number of the samples on which the gradient w.r.t. $\mathbf{q}$ is based.

SmoothAP loss. A full derivation of the gradient of SmoothAP w.r.t. $\mathbf{q}$ is provided with the implementation of our methods.[1] We introduce $sim(D_{ij})$, the derivative of (7):

$$\frac{\partial AP_{\mathbf{q}}}{\partial \mathbf{q}} = \frac{1}{|\mathcal{S}_{\mathcal{P}}^{\mathbf{q}}|}\sum_{i\in\mathcal{S}_{\mathcal{P}}^{\mathbf{q}}}\mathcal{R}(i,\mathcal{S}_{\Omega}^{\mathbf{q}})^{-2}\Big(\mathcal{R}(i,\mathcal{S}_{\mathcal{P}}^{\mathbf{q}})\Big(\sum_{j\in\mathcal{S}_{\mathcal{N}}^{\mathbf{q}}} sim(D_{ij})(\mathbf{v}_i-\mathbf{v}_j)\Big) - (\mathcal{R}(i,\mathcal{S}_{\mathcal{N}}^{\mathbf{q}})-1)\Big(\sum_{j\in\mathcal{S}_{\mathcal{P}}^{\mathbf{q}},j\neq i} sim(D_{ij})(\mathbf{v}_i-\mathbf{v}_j)\Big)\Big). \tag{12}$$

Given Eq. 12, it is less trivial to infer what the update w.r.t. $\mathbf{q}$ looks like in terms of positive candidates $\mathbf{v}_i$ and negative candidates $\mathbf{v}_j$. However, we can derive the following two properties: (1) The lower a positive candidate $\mathbf{v}_i$ is in the total ranking, the less this candidate is taken into account for the gradient computation w.r.t. $\mathbf{q}$, due the inverse quadratic term $\mathcal{R}(i,\mathcal{S}_{\Omega}^{q})^{-2}$. This is in line with optimizing the AP as a metric; positive candidates that are ranked low contribute less to the total AP score, and therefore are less important to optimize. (2) Each triplet $\mathbf{v}_i-\mathbf{v}_j$ is weighted according to their difference in similarity score D_{ij}. If their difference in similarity score w.r.t. query $\mathbf{q}$ is relatively small (i.e., D_{ij} is close to zero), $sim(D_{ij})$ will have a high value due to the fact that $sim(D_{ij})$ is the derivative of the sigmoid function. Therefore, $sim(D_{ij})$ indicates how close the similarity score (with the query) of candidate $\mathbf{v}_i$ is compared to the similarity score of $\mathbf{v}_j$ This is in line with the SmoothAP loss because we use a sigmoid to approximate the step-function; only triplets of candidates that have a similar similarity score will contribute to the gradient.

We define a threshold value ϵ again. If the value of $sim(D_{ij})$ is lower than the threshold value, we consider the contribution of this triplet to be negligible. We have to take into account that all triplets are also weighted by $\mathcal{R}(i,\mathcal{S}_{\Omega}^{\mathbf{q}})^{-2}$, which is always lower than or equal to 1. We can define $C_{Smooth}^{\mathbf{q}}$, which *is the number of triplets* $\mathbf{v}^+-\mathbf{v}^-$ *that contribute to the gradient w.r.t.* $\mathbf{q}$, for SmoothAP as follows:

$$C_{Smooth}^{\mathbf{q}} = \frac{1}{|\mathcal{S}_{\mathcal{P}}^{\mathbf{q}}|}\sum_{i\in\mathcal{S}_{\mathcal{P}}^{\mathbf{q}}}\Big(\sum_{j\in\mathcal{S}_{\mathcal{N}}^{\mathbf{q}}}\mathbb{1}\Big\{\frac{sim(D_{ij})}{\mathcal{R}(i,\mathcal{S}_{\Omega}^{\mathbf{q}})^2}>\epsilon\Big\}+\sum_{j\in\mathcal{S}_{\mathcal{P}}^{\mathbf{q}},j\neq i}\mathbb{1}\Big\{\frac{sim(D_{ij})}{\mathcal{R}(i,\mathcal{S}_{\Omega}^{\mathbf{q}})^2}>\epsilon\Big\}\Big). \tag{13}$$

Similar to [1], we use $sim(D_{ij})$ in combination with a threshold value ϵ to indicate which samples have a non-zero gradient in the training batch. We ignore the terms $\mathcal{R}(i,\mathcal{S}_{\mathcal{P}}^{\mathbf{q}})$ and $1-\mathcal{R}(i,\mathcal{S}_{\mathcal{N}}^{\mathbf{q}})$ for this gradient computation. We also count all queries $\mathbf{q}$ within batch $\mathcal{B}$ that do not have a gradient value. We define this number as $C_{Smooth}^{0}=\sum_{\mathbf{q}\in\mathcal{B}}\mathbb{1}\{C_{Smooth}^{\mathbf{q}}=0\}$. This completes the definition of COCOS: for every loss function that we consider, it counts the number of candidates that contribute to the gradient w.r.t. $\mathbf{q}$.

5 Analyzing the Behavior of Loss Functions for ICR

Experimental setup. To use COCOS , we introduce the following experimental setup. For each loss function, we take the checkpoint of one of the five optimized models. We refer to this checkpoint as *the optimal convergence point* for this loss function. This is not the point with the lowest loss value, but the model checkpoint that results in the highest evaluation scores on the validation set. We freeze all model parameters and do not apply dropout. We iterate over the entire training set by sampling random batches $\mathcal{B}$ (with batch size $|\mathcal{B}|=128$, similar to the training set-up). For each batch we compute the COCOS and weight values defined in Section 4. We report the mean value and standard deviation over the entire training set for both VSE++ and VSRN, for both datasets and for each loss function. The only hyper-parameter for this experiment is ϵ. We use $\epsilon=0.01$ for both the NT-Xent and SmoothAP loss.

Experimental outcomes. For each of the loss functions that we consider, we analyze

Table 2: COCOS w.r.t. query $\mathbf{q}$, for the Triplet loss and the Triplet loss SH.

				i2t			t2i		
			#	$C^{\mathbf{q}}$	$C^{\mathcal{B}}$	C^{0}	$C^{\mathbf{q}}$	$C^{\mathcal{B}}$	C^{0}
Flickr30k	VSE++	Triplet loss	1.1	6.79±0.83	768.92±96.87	14.78±3.52	6.11±0.75	774.67±98.05	1.14±1.22
		Triplet loss SH	1.2	1±0.0	98.74±4.83	29.23±4.81	1±0.0	98.22±4.66	29.75±4.62
	VSRN	Triplet loss	1.5	1.39±0.12	60.96±10.30	84.29±5.80	1.28±0.10	61.21±10.01	80.15±6.35
		Triplet loss SH	1.6	1±0.0	45.59±5.93	82.39±5.92	1±0.0	44.98±5.70	82.99±5.70
MS-COCO	VSE++	Triplet loss	2.1	3.51±0.49	353.82±52.71	27.09±4.60	2.94±0.36	341.64±50.80	12.24±4.92
		Triplet loss SH	2.2	1 ±0.0	88.17±5.25	39.82±5.24	1±0.0	87.24±5.34	40.75±5.33
	VSRN	Triplet loss	2.5	1.21±0.13	29.88±7.46	103.33±5.22	1.15±0.10	30.25±7.49	101.70±5.58
		Triplet loss SH	2.6	1±0.0	33.24±5.39	94.73±5.45	1±0.0	32.90±5.35	95.08±5.4

its performance using COCOS.

Triplet loss. Our goal is not to show that the Triplet loss SH outperforms the Triplet loss, which has already been shown [12], but to explain this behavior based on COCOS w.r.t. $\mathbf{q}$ and also relate this to the NT-Xent and SmoothAP loss.

Based on Table 1 (row 1.1/1.2 and 1.5/1.6, row 2.1/2.2 and 2.5/2.6) it is clear that the Triplet loss SH always outperforms the general Triplet loss with a large margin. If we look at Table 2, row 1.1/1.2 and 2.1/2.2, respectively, there is a clear relation between $\mathcal{C}^{\mathbf{q}}$ and the final evaluation score for the VSE++ model for both sub-tasks i2t and t2i (Table 1). $\mathcal{C}^{\mathbf{q}}_{Triplet}$ and $\mathcal{C}^{\mathcal{B}}_{Triplet}$ are both much greater than $\mathcal{C}^{\mathbf{q}}_{TripletSH}$ and $\mathcal{C}^{\mathcal{B}}_{TripletSH}$, for both dataset and both the the i2t and t2i task. When multiple negatives with small margin violation are combined into a gradient, the gradient is dominated by easy or non-informative negative samples, which results in convergence of the model into a sub-optimal point [12]. Clearly, the loss function with the lowest evaluation score takes into account the most negatives when computing the gradient w.r.t. $\mathbf{q}$. Based on [12] and the COCOS results in Table 2 we conclude that, at the optimal convergence point, the Triplet loss takes too many negatives into account (i.e., too many triplets still violate the margin constraint), leading to lower evaluation scores.

For VSRN the relation between $\mathcal{C}^{\mathbf{q}}_{Triplet}$, $\mathcal{C}^{\mathbf{q}}_{TripletSH}$ and the final evaluation score is less clear. If we look at Table 2, row 1.5/1.6 and 2.5/2.6, respectively, we see that $C^{\mathbf{q}}_{Triplet} \approx C^{\mathbf{q}}_{TripletSH} = 1$. This means that at the optimal convergence point, for VSRN, the Triplet loss and the Triplet loss SH (approximately) are a similar to each other and both functions only take one negative triplet into account when computing the gradient w.r.t. $\mathbf{q}$. Thus, both functions should result in approximately the same gradient value while the Triplet loss SH still outperforms the Triplet loss with a large margin. This can be explained as follows: At the start of training, for each query $\mathbf{q}$ (almost) all triplets violate the margin constraint (because all candidate representations are random). Therefore, the gradient(s) computation w.r.t. $\mathbf{q}$ for the Triplet loss is based on all triplets in the batch and therefore this gradient is dominated by a majority of non-informative samples in the beginning of the training, which leads to convergence at a sub-optimal point.

NT-Xent. Based on Table 3, we can see that $C^{\mathbf{qv}^{-}}_{NT-Xent}$ is higher than 1 for both VSE++ and VSRN, for i2t and t2i, on both datasets. If we relate the evaluation performances of the NT-Xent loss (row 1.3, 1.7, 2.3, 2.7) to the Triplet loss SH (row 1.2, 1.6, 2.2, 2.6) in Table 1, we can see that the Triplet loss SH consistently outperforms the NT-Xent

Table 3: COCOS w.r.t. query $\mathbf{q}$, for the NT-Xent loss [7].

			i2t			t2i		
		#	$C^{\mathbf{qv}^-}_{NT-Xent}$	$W^{\mathbf{qv}^-}_{NT-Xent}$	$W^{\mathbf{qv}^+}_{NT-Xent}$	$C^{\mathbf{qv}^-}_{NT-Xent}$	$W^{\mathbf{qv}^-}_{NT-Xent}$	$W^{\mathbf{qv}^+}_{NT-Xent}$
Flickr30k	VSE++	1.3	9.88±0.51	0.42±0.02	0.56±0.02	9.65±0.51	0.42±0.02	0.56±0.02
	VSRN	1.7	2.45±0.23	0.13±0.02	0.20±0.02	2.46±0.23	0.13±0.02	0.20±0.02
MS-COCO	VSE++	2.3	5.59±0.40	0.36±0.02	0.46±0.02	5.33±0.38	0.36±0.02	0.46±0.02
	VSRN	2.7	1.10±0.14	0.10±0.02	0.14±0.02	1.11±0.14	0.09±0.02	0.14±0.02

Table 4: COCOS w.r.t. query $\mathbf{q}$, for the SmoothAP [1] loss.

			i2t		t2i	
		#	$C^{\mathbf{q}}_{SmoothAP}$	$C^{0}_{SmoothAP}$	$C^{\mathbf{q}}_{SmoothAP}$	$C^{0}_{SmoothAP}$
Flickr30k	VSE++	1.4	1.27±0.06	2.15±1.51	1.47±0.83	636.72±18.72
	VSRN	1.8	2.33±0.07	0.00±0.00	1.62±0.95	636.49±18.65
MS-COCO	VSE++	2.4	1.48±0.07	0.80±0.90	1.41±0.74	637.10±20.28
	VSRN	2.8	1.67±0.07	0.14±0.37	1.42±0.76	637.23±20.35

loss, regardless of the method, dataset or sub-task. We therefore can conclude that taking only the most violating negative into account when computing the gradient w.r.t. $\mathbf{q}$ results in better evaluation performances than computing a weighted sum over all negative candidates. We can apply the same reasoning used to explain the performance difference between the Triplet loss and Triplet loss SH. The gradient w.r.t. $\mathbf{q}$ for the NT-Xent is dominated by too many non-informative negatives, which have a weight value bigger than ϵ.

Looking at Table 1, we see that NT-Xent loss outperforms the Triplet loss for the VSE++ method (1.3/1.1 and 2.3/2.1), while taking more negative samples into account when computing the gradient (based on our definition of COCOS). This in contrast with the previous observation for the Triplet loss of the more (non-informative) samples a loss function takes into account when computing the gradient w.r.t. $\mathbf{q}$, the lower the evaluation score. Solely counting the number of negative examples that contribute to the gradient does not the provide the full picture for the NT-Xent loss; the weight value of each individual sample (including the positive) plays a more important role than initially was assumed. We have tried different values for ϵ, with little impact.

SmoothAP. The observations in Table 4 are in line with the observations in Table 2 and the evaluation performance in Table 1. At the optimal convergence point SmoothAP takes approximately one triplet into account when computing the gradient w.r.t. $\mathbf{q}$, which results in close-to or similar performances as the Triplet loss SH. We also observe the following: the only experiment where the Triplet loss SH outperforms SmoothAP with a large margin (Table 1, row 1.5 and 1.8), is also the experiment where the SmoothAP function takes the highest number of negatives into account when computing the gradient w.r.t. $\mathbf{q}$ (Table 4, row 1.8). This supports the general observation that the more samples that contribute to the gradient, the lower the final evaluation score.

For the t2i task, we also see that $C^{0}_{SmoothAP}$ is almost as big as the number of samples ($640{=}(k{=}5){\times}(|\mathcal{B}|{=}128)$) in the candidate set, for both datasets and methods. Hence, barely any query has a gradient value anymore at the optimal convergence point. However,

this is not the case for the i2t task. We conclude that optimizing a ranking metric (i.e., AP) with only one positive candidate (as is the case for the t2i task), might be too easy to optimize and could result in over-fitting. Therefore, it is not useful to optimize a ranking task like ICR with a ranking-based loss function when there is only one positive candidate per query, which is the case for the i2t task. For the i2t task, however, there are barely any queries without a gradient value; here we have k positive candidates per query.

Upshot. In summary, (1) it is important to focus on only one (or a limited) number of (hard) negatives per query during the entire training for the gradient computation, so as to prevent the gradient from being dominated by non-informative or easy negative samples. (2) Weighting each negative candidate by its score (as is done in NT-Xent) as opposed to weighting all negative equally (as is done in the Triplet loss) can beneficial for the gradient computation and therefore for the final evaluation score. However, this weighted sum of negatives does not result in the fact that the NT-Xent loss outperforms the Triplet loss SH, which implies that the gradient computation for the NT-Xent is still based on too many non-informative samples.

6 Conclusion

We have examined three loss functions from the metric learning field to question if the promising results obtained in metric learning generalize to the image-caption retrieval (ICR) task. In contrast with the findings from metric learning, we find that the Triplet loss with semi-hard negative mining still outperforms the NT-Xent and SmoothAP loss. Hence, the Triplet loss should still be the de facto choice as a loss function for ICR; results from metric learning do not generalize directly to ICR. To gain a better understanding of why a loss function results in better performance than others, we have introduced the notion of counting contributing samples (COCOS). We have shown that the best performing loss function only focuses on one (hard) negative sample when computing the gradient w.r.t. the query and therefore results in the most informative gradient. COCOS suggests that the underperforming loss functions take too many (non-informative) negatives into account, and therefore converge to a sub-optimal point.

The definition of COCOS uses a threshold value. The idea that a candidate contributes to the gradient if its weight value is above a certain threshold is insightful but does not provide the complete picture of how strong the influence of this sample is. We encourage two directions for future work: (1) Work on more sophisticated methods to determine the influence of (the number of) samples on the gradient w.r.t. a query. (2) Design new loss functions for the ICR task by taking the lessons from COCOS into account, i.e., loss functions that only take one, or a limited number of, hard negative(s) into account. Additionally, we want to investigate if our findings generalize to fields such as Dense Passage Retrieval (DPR) [18]. DPR methods are also mainly optimized by using two data encoders [18, 19], for the query and for documents, and the main learning objective is contrasting positive and negative candidates with a query [8, 13, 14, 18, 19, 44], similar to ICR.

Acknowledgements. We thank Gabriel Benedict, Mariya Hendriksen, Maria Heuss, Sarah Ibrahimi, and Ana Lucic for feedback and discussions. This research was supported by the Nationale Politie and the Hybrid Intelligence Center through the Netherlands Organisation for Scientific Research. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

Bibliography

[1] Brown A, Xie W, Kalogeiton V, Zisserman A (2020) Smooth-AP: Smoothing the path towards large-scale image retrieval. In: European Conference on Computer Vision (ECCV), Springer, pp 677–694

[2] Bruch S, Wang X, Bendersky M, Najork M (2019) An analysis of the softmax cross entropy loss for learning-to-rank with binary relevance. In: International Conference on Theory of Information Retrieval (ICTIR), ACM, pp 75–78

[3] Bruch S, Zoghi M, Bendersky M, Najork M (2019) Revisiting approximate metric optimization in the age of deep neural networks. In: SIGIR Conference on Research and Development in Information Retrieval (SIGIR), ACM, pp 1241–1244

[4] Chen H, Ding G, Liu X, Lin Z, Liu J, Han J (2020) Imram: Iterative matching with recurrent attention memory for cross-modal image-text retrieval. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 12655–12663

[5] Chen T, Li L (2020) Intriguing properties of contrastive losses. arXiv preprint arXiv:201102803

[6] Chen T, Deng J, Luo J (2020) Adaptive offline quintuplet loss for image-text matching. In: European Conference on Computer Vision (ECCV), Springer, pp 549–565

[7] Chen T, Kornblith S, Norouzi M, Hinton G (2020) A simple framework for contrastive learning of visual representations. In: International conference on machine learning, PMLR, pp 1597–1607

[8] Chen X, He B, Hui K, Sun L, Sun Y (2021) Simplified TinyBERT: Knowledge distillation for document retrieval. In: Advances in Information Retrieval, Springer International Publishing, pp 241–248

[9] Chen YC, Li L, Yu L, El Kholy A, Ahmed F, Gan Z, Cheng Y, Liu J (2020) Uniter: Universal image-text representation learning. In: European Conference on Computer Vision (ECCV), Springer, pp 104–120

[10] Cho K, Van Merriënboer B, Bahdanau D, Bengio Y (2014) On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:14091259

[11] Diao H, Zhang Y, Ma L, Lu H (2021) Similarity reasoning and filtration for image-text matching. arXiv preprint arXiv:210101368

[12] Faghri F, Fleet DJ, Kiros JR, Fidler S (2018) VSE++: Improving visual-semantic embeddings with hard negatives. In: Proceedings of the British Machine Vision Conference (BMVC)

[13] Formal T, Piwowarski B, Clinchant S (2021) Splade: Sparse lexical and expansion model for first stage ranking. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 2288–2292

[14] Gao L, Dai Z, Callan J (2021) Rethink training of BERT rerankers in multi-stage retrieval pipeline. In: Advances in Information Retrieval, Springer International Publishing, pp 280–286

[15] He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 770–778

[16] Jia C, Yang Y, Xia Y, Chen YT, Parekh Z, Pham H, Le QV, Sung Y, Li Z, Duerig T (2021) Scaling up visual and vision-language representation learning with noisy text supervision. arXiv preprint arXiv:210205918

[17] Karpathy A, Fei-Fei L (2015) Deep visual-semantic alignments for generating image descriptions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 3128–3137

[18] Karpukhin V, Oguz B, Min S, Lewis P, Wu L, Edunov S, Chen D, Yih Wt (2020) Dense passage retrieval for open-domain question answering. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, pp 6769–6781

[19] Khattab O, Zaharia M (2020) Colbert: Efficient and effective passage search via contextualized late interaction over bert. In: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, pp 39–48

[20] Kim W, Son B, Kim I (2021) Vilt: Vision-and-language transformer without convolution or region supervision. arXiv preprint arXiv:210203334

[21] Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:160902907

[22] Kiros R, Salakhutdinov R, Zemel RS (2014) Unifying visual-semantic embeddings with multimodal neural language models. arXiv preprint arXiv:14112539

[23] Krishna R, Zhu Y, Groth O, Johnson J, Hata K, Kravitz J, Chen S, Kalantidis Y, Li LJ, Shamma DA, et al. (2016) Visual genome: Connecting language and vision using crowdsourced dense image annotations. arXiv preprint arXiv:160207332

[24] Lee KH, Chen X, Hua G, Hu H, He X (2018) Stacked cross attention for image-text matching. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 201–216

[25] Li K, Zhang Y, Li K, Li Y, Fu Y (2019) Visual semantic reasoning for image-text matching. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp 4654–4662

[26] Li X, Yin X, Li C, Hu X, Zhang P, Zhang L, Wang L, Hu H, Dong L, Wei F, Choi Y, Gao J (2020) Oscar: Object-semantics aligned pre-training for vision-language tasks. In: European Conference on Computer Vision (ECCV), Springer, pp 121–137

[27] Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft COCO: Common objects in context. In: European Conference on Computer Vision, Springer, pp 740–755

[28] Lu J, Batra D, Parikh D, Lee S (2019) Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. arXiv preprint arXiv:190802265

[29] Lu X, Zhao T, Lee K (2021) VisualSparta: An embarrassingly simple approach to large-scale text-to-image search with weighted bag-of-words. arXiv preprint arXiv:210100265

[30] Messina N, Amato G, Esuli A, Falchi F, Gennaro C, Marchand-Maillet S (2020) Fine-grained visual textual alignment for cross-modal retrieval using transformer encoders. arXiv preprint arXiv:200805231

[31] Messina N, Falchi F, Esuli A, Amato G (2021) Transformer reasoning network for image-text matching and retrieval. In: International Conference on Pattern Recognition (ICPR), IEEE, pp 5222–5229

[32] Musgrave K, Belongie S, Lim SN (2020) A metric learning reality check. In: European Conference on Computer Vision, Springer, pp 681–699
[33] van den Oord A, Li Y, Vinyals O (2018) Representation learning with contrastive predictive coding. arXiv preprint arXiv:180703748
[34] Oosterhuis H, de Rijke M (2018) Differentiable unbiased online learning to rank. In: International Conference on Information and Knowledge Management (CIKM), ACM, pp 1293–1302
[35] Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J, et al. (2021) Learning transferable visual models from natural language supervision. arXiv preprint arXiv:210300020
[36] Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, vol 28, pp 91–99
[37] Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:14091556
[38] Varamesh A, Diba A, Tuytelaars T, Van Gool L (2020) Self-supervised ranking for representation learning. arXiv preprint arXiv:201007258
[39] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, pp 5998–6008
[40] Verma G, Vinay V, Bansal S, Oberoi S, Sharma M, Gupta P (2020) Using image captions and multitask learning for recommending query reformulations. In: Advances in Information Retrieval (ECIR), Springer, pp 681–696
[41] Wang X, Li C, Golbandi N, Bendersky M, Najork M (2018) The lambdaloss framework for ranking metric optimization. In: Conference on Information and Knowledge Management (CIKM), ACM, pp 1313–1322
[42] Young P, Lai A, Hodosh M, Hockenmaier J (2014) From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. Transactions of the Association for Computational Linguistics 2:67–78
[43] Zeng D, Yu Y, Oyama K (2020) Deep triplet neural networks with cluster-cca for audio-visual cross-modal retrieval. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 16(3):1–23
[44] Zhan J, Mao J, Liu Y, Guo J, Zhang M, Ma S (2021) Optimizing dense retrieval model training with hard negatives. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), Association for Computing Machinery, pp 1503–1512
[45] Zhang Y, Lu H (2018) Deep cross-modal projection learning for image-text matching. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 686–701

This appendix has two sections, one devoted to a derivation of the gradient of SmoothAP w.r.t. q (Appendix A), and one devoted to reproducibility (Appendix B).

A Derivative of the gradient of SmoothAP w.r.t. q

A.1 Explanation of SmoothAP

The Average Precision metric represents the area under the precision-recall curve. Average Precision is a discrete metric and therefore can not be used directly as loss function for optimizing retrieval methods. The main intuition behind the SmoothAP [1] is to have a smooth, and therefore differentiable, approximation of the Average Precision metric. Using the notation introduced in Section 2, the Average Precision metric is defined as follows:

$$AP_{\mathbf{q}} = \frac{1}{|\mathcal{S}_{\mathcal{P}}^{\mathbf{q}}|} \sum_{i \in \mathcal{S}_{\mathcal{P}}^{\mathbf{q}}} \frac{\mathcal{R}(i, \mathcal{S}_{\mathcal{P}}^{\mathbf{q}})}{\mathcal{R}(i, \mathcal{S}_{\Omega}^{\mathbf{q}})}, \tag{14}$$

where $\mathcal{R}(i,\mathcal{S})$ is defined as:

$$\mathcal{R}(i,\mathcal{S}) = 1 + \sum_{j \in \mathcal{S}, i \neq j} \mathbb{1}\{s_i - s_j < 0\}. \tag{15}$$

$\mathcal{R}(i,\mathcal{S})$ returns the rank of candidate i in a ranking over a set with candidates $\mathcal{S}$, given query $\mathbf{q}$. s_i is the similarity score between the query $\mathbf{q}$ and candidate $\mathbf{v}_i$. Similarly, s_j is similarity score between the query $\mathbf{q}$ and candidate $\mathbf{v}_j$. If $s_i - s_j$ is lower than 0, this indicates that candidate j is ranked higher than candidate i. By counting how many times $\mathbb{1}\{s_i - s_j < 0\}$ is true, the ranking of candidate i can be determined.

To simplify the computation and notation, we can introduce a matrix D, where D is defined as:

$$D = \begin{bmatrix} s_1 & \dots & s_m \\ \vdots & \ddots & \vdots \\ s_1 & \dots & s_m \end{bmatrix} - \begin{bmatrix} s_1 & \dots & s_1 \\ \vdots & \ddots & \vdots \\ s_m & \dots & s_m \end{bmatrix}, \tag{16}$$

where $D_{ij} = s_i - s_j$. In this case we have a candidate set $\mathcal{S}$ with m candidates. By using matrix D, we can rewrite Eq. 15 as follows:

$$\mathcal{R}(i,\mathcal{S}) = 1 + \sum_{j \in \mathcal{S}, i \neq j} \mathbb{1}\{D_{ij} > 0\}. \tag{17}$$

To make $\mathcal{R}(i,\mathcal{S})$, and thereby $AP_{\mathbf{q}}$, differentiable, the indicator function $\mathbb{1}$ is replaced by the smooth sigmoid function $\mathcal{G}(\cdot,\tau)$.

A.2 Derivative of the gradient of SmoothAP w.r.t. q

In this section, we give an analyses and derivation of the gradient of SmoothAP [1] w.r.t. query $\mathbf{q}$. We start with Eq. 18, the definition of SmoothAP:

$$AP_{\mathbf{q}} = \frac{1}{|\mathcal{S}_{\mathcal{P}}^{\mathbf{q}}|} \sum_{i \in \mathcal{S}_{\mathcal{P}}^{\mathbf{q}}} \frac{1 + \sum_{j \in \mathcal{S}_{\mathcal{P}}^{\mathbf{q}}, j \neq i} \mathcal{G}(D_{ij};\tau)}{1 + \sum_{j \in \mathcal{S}_{\mathcal{P}}^{\mathbf{q}}, j \neq i} \mathcal{G}(D_{ij};\tau) + \sum_{j \in \mathcal{S}_{\mathcal{N}}^{\mathbf{q}}} \mathcal{G}(D_{ij};\tau)}. \tag{18}$$

Here, $\mathcal{G}$ is a smooth approximation of an indicator/step function:

$$\mathcal{G}(f(x);\tau)=\frac{1}{1+e^{-\frac{f(x)}{\tau}}}. \tag{19}$$

The derivative of $\mathcal{G}$ w.r.t. a function $f(x)$ has the following form:

$$\frac{\partial \mathcal{G}(f(x);\tau)}{\partial x}=\mathcal{G}(f(x);\tau)(1-\mathcal{G}(f(x);\tau))\frac{1}{\tau}\frac{\partial f(x)}{\partial x}. \tag{20}$$

Note that $f(x)$ in the case of SmoothAP is D_{ij}:

$$D_{ij}=s_i-s_j=\mathbf{q}\mathbf{v}_i-\mathbf{q}\mathbf{v}_j, \tag{21}$$

where both $\mathbf{v}_i$, $\mathbf{v}_j$ and $\mathbf{q}$ are normalized on the unit-sphere. The gradient of D_{ij} w.r.t. query $\mathbf{q}$ has the following form:

$$\frac{\partial D_{ij}}{\partial \mathbf{q}}=\mathbf{v}_i-\mathbf{v}_j. \tag{22}$$

If we plug in D_{ij} into Eq. 20 and take the gradient w.r.t. query $\mathbf{q}$, we get

$$\begin{aligned}\frac{\partial \mathcal{G}(D_{ij};\tau)}{\partial \mathbf{q}}&=\mathcal{G}(D_{ij};\tau)(1-\mathcal{G}(D_{ij};\tau))\frac{1}{\tau}(\mathbf{v}_i-\mathbf{v}_j)\\&=sim(D_{ij},\tau)(\mathbf{v}_i-\mathbf{v}_j),\end{aligned} \tag{23}$$

where $sim(D_{ij},\tau)$ is a function that gives an indication of how close the similarity scores are of candidate i and j are w.r.t. query $\mathbf{q}$, scaled by τ:

$$sim(D_{ij},\tau)=\mathcal{G}(D_{ij};\tau)(1-\mathcal{G}(D_{ij};\tau))\frac{1}{\tau}. \tag{24}$$

Now we define $\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\Omega})$ and $\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\mathcal{P}})$. $\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\Omega})$ gives the ranking of candidate i within the full candidate set $\mathcal{S}^{\mathbf{q}}_{\Omega}$. $\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\mathcal{P}})$. gives the rank of candidate i within the positive candidate set $\mathcal{S}^{\mathbf{q}}_{\mathcal{P}}$:

$$\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\Omega})=\left(\overbrace{1+\sum_{j\in\mathcal{S}^{\mathbf{q}}_{\mathcal{P}},j\neq i}\mathcal{G}(D_{ij};\tau)}^{A}+\overbrace{\sum_{j\in\mathcal{S}^{\mathbf{q}}_{\mathcal{N}}}\mathcal{G}(D_{ij};\tau)}^{C}\right) \tag{25}$$

$$\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\mathcal{P}})=\left(\overbrace{1+\sum_{j\in\mathcal{S}^{\mathbf{q}}_{\mathcal{P}},j\neq i}\mathcal{G}(D_{ij};\tau)}^{A}\right). \tag{26}$$

The gradient of $\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\Omega})$ w.r.t. to $\mathbf{q}$ has the following form:

$$\frac{\partial \mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\Omega})}{\partial \mathbf{q}}=\left(\overbrace{\sum_{j\in\mathcal{S}^{\mathbf{q}}_{\mathcal{P}},j\neq i}sim(D_{ij})(\mathbf{v}_i-\mathbf{v}_j)}^{B}+\overbrace{\sum_{j\in\mathcal{S}^{\mathbf{q}}_{\mathcal{N}}}sim(D_{ij})(\mathbf{v}_i-\mathbf{v}_j)}^{D}\right). \tag{27}$$

Using all the definitions above, we can write the full gradient of $AP_{\mathbf{q}}$ w.r.t. $\mathbf{q}$:

$$\begin{aligned}&\frac{\partial AP_{\mathbf{q}}}{\partial \mathbf{q}}\\&=\frac{1}{|\mathcal{S}^{\mathbf{q}}_{P}|}\sum_{i\in\mathcal{S}^{\mathbf{q}}_{\mathcal{P}}}\frac{\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\mathcal{P}})\frac{\partial \mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\Omega})}{\partial \mathbf{q}}-\mathcal{R}(i,\mathcal{S}^{q}_{\Omega})\left(\sum_{j\in\mathcal{S}^{\mathbf{q}}_{\mathcal{P}},j\neq i}sim(D_{ij})(\mathbf{v}_i-\mathbf{v}_j)\right)}{\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\Omega})^2}\end{aligned} \tag{28}$$

$$= \frac{1}{|\mathcal{S}^{\mathbf{q}}_{\mathcal{P}}|} \sum_{i \in \mathcal{S}^{\mathbf{q}}_{\mathcal{P}}} \frac{1}{\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\Omega})^2} \left(\left(\overbrace{\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\mathcal{P}})}^{A} \overbrace{\frac{\partial \mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\Omega})}{\partial \mathbf{q}}}^{B+D} \right) - \left(\overbrace{\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\Omega})}^{A+C} \left(\overbrace{\sum_{j \in \mathcal{S}^{\mathbf{q}}_{\mathcal{P}}, j \neq i} sim(D_{ij})(\mathbf{v}_i - \mathbf{v}_j)}^{D} \right) \right) \right). \quad (29)$$

When looking at Eq. 29 it becomes clear that we have a function in the following form $A(B+D)-(A+C)B$. This can be rewritten to: $AB+AD-AB-CB=AD-CB$. If we apply this to Eq. 29, we end up with the following form:

$$\frac{\partial AP_{\mathbf{q}}}{\partial \mathbf{q}}$$

$$= \frac{1}{|\mathcal{S}^{\mathbf{q}}_{\mathcal{P}}|} \sum_{i \in \mathcal{S}^{\mathbf{q}}_{\mathcal{P}}} \frac{1}{\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\Omega})^2} \left(\overbrace{\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\mathcal{P}})}^{A} \left(\overbrace{\sum_{j \in \mathcal{S}^{\mathbf{q}}_{\mathcal{N}}} sim(D_{ij})(\mathbf{v}_i - \mathbf{v}_j)}^{D} \right) - \overbrace{\sum_{j \in \mathcal{S}^{\mathbf{q}}_{\mathcal{N}}} \mathcal{G}(D_{ij};\tau)}^{C} \left(\overbrace{\sum_{j \in \mathcal{S}^{\mathbf{q}}_{\mathcal{P}}, j \neq i} sim(D_{ij})(\mathbf{v}_i - \mathbf{v}_j)}^{B} \right) \right) \quad (30)$$

$$= \frac{1}{|\mathcal{S}^{\mathbf{q}}_{P}|} \sum_{i \in \mathcal{S}^{\mathbf{q}}_{\mathcal{P}}} \frac{1}{\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\Omega})^2} \left(\mathcal{R}(i,\mathcal{S}^{\mathbf{q}}_{\mathcal{P}}) \left(\sum_{j \in \mathcal{S}^{q}_{\mathcal{N}}} sim(D_{ij})(\mathbf{v}_i - \mathbf{v}_j) \right) - (\mathcal{R}(i,\mathcal{S}^{q}_{\mathcal{N}}) - 1) \left(\sum_{j \in \mathcal{S}^{\mathbf{q}}_{P}, j \neq i} sim(D_{ij})(\mathbf{v}_i - \mathbf{v}_j) \right) \right). \quad (31)$$

B Reproducibility

B.1 VSE++

The VSE++ [12] is an ICR method that uses two encoders that do not share parameters: an image and a caption encoder. For the image encoder, two CNN networks have been used in [12]: ResNet-152 [15] and VGG19 [37], where ResNet-152 yields the best evaluation performances. To reduce the computation time of the training process, we have decided to use ResNet-50 instead of ResNet-152. Some preliminary experiments have shown that the differences in evaluation score between ResNet-152 and 50 is relatively small, while ResNet-50 is faster to optimize. The ResNet network functions as a so-called backbone or feature extractor. On top of the ResNet network, a fully-connected layer is placed to map the extracted features to a multi-modal latent space. Only the weights of this fully-connected layer are optimized during training.

The caption encoder consists of a unidirectional GRU [10] encoder. The word embeddings for the text encoder are trained end-to-end (from scratch) with the rest of

text encoder. The output of the last encoding step is the representation for the input caption. The output representations of both encoders are normalized on the unit sphere.

B.2 VSRN

VSRN [25] also consists of separate text and image encoder. For the image encoder VSRN uses pre-computed features as input. These features have been generated by a Faster R-CNN [36] model, which uses ResNet-101 [15] as backbone, trained on the Visual Genomes dataset [23]. The feature map of the last convolutional layer serves as input representation for the next layer, each vector in this feature map represents a region in the input image. Next, a GCN [21] is used to enhance the input feature vectors with relation information between each region in the input image. Finally, a GRU is used to compute the global representation of the different region vectors. This is done by feeding the region representations one by one into the GRU encoder as a sequence. VSRN uses the same text encoder as VSE++.

To generate an extra training signal, a caption generator is added to the training process. This generator is optimized to reconstruct the input caption based on the visual region feature representations. We have decided to remove this caption decoder from the learning algorithm. The reason for this is that we focus on ICR only and we want to exclude any additional learning signal from the training process.

B.3 Implementation and optimization details

Both VSE++ and VSRN are optimized for 30 epochs on the same datasets [27, 42]. For VSE++, after 30 epochs of training, 15 epochs of additional fine-tune are applied where the backbone of the image encoder is also optimized. We do not apply this additional fine-tuning step due to the following two reasons: (1) We want to optimize VSE++ and VSRN for the same number of epochs, and (2) Our goal is not to have the best performing model, but rather to evaluate the impact of a loss function. Therefore, the weights of the feature extractor for the VSE++ image encoder are frozen during the entire training process in this work. All the other remaining implementation details and hyper-parameters in this work are similar to [12, 25].