MSE * *: Multi-modal semantic embeddings for datasets with several positive matchings

Jérémie Huteau, Adrian Basarab, Florence Dupin de Saint-Cyr

MSE**: Multi-modal semantic embeddings for datasets with several positive matchings*

Jérémie Huteau† Adrian Basarab Florence Dupin de Saint-Cyr

IRIT, Toulouse University, France

Abstract

In the context where the user wants to retrieve an image corresponding to a sentence, deep learning frameworks have started to give very good results. More precisely contrastive learning can be used to learn good representations of the same object presented under different modalities (text, image, video, etc.). The common representation of the same object is called semantic embedding, and in the case of image and text modalities it becomes visual semantic embedding (VSE). In this paper, we propose an approach which extends a visual semantic embedding approach called VSE^{++} with the ability to handle multiple-modalities and to dispose of several positive items of the same modality for one object. We compare the two methods and we show that despite a better expressivity MSE** gives nearly the same results as VSE^{++} on several datasets. We show that the Loss function of MSE** is more accurate for some hard cases of our dataset. This work opens several perspectives: 1) use MSE** on other datasets having many examples of each class (e.g., a sentence that could be linked with several images), 2) use a VSE model to find new positive pairs and to eliminate false negatives of the dataset, 3) associate images with logical formulas. This last perspective could allow for post-process reasoning. It could also improve the accuracy by enabling us to incorporate the specificity of formulas when comparing the similarities of the images associated to them.

Keywords: Deep Learning; Contrastive Learning; Triplet Loss.

1 Introduction

Understanding an image or a text, even independently, is an extremely complex task, for which handcrafted methods are both difficult to develop and not very robust. Advances in machine learning have nevertheless made it possible to obtain results about the similarity between two images, or between two sentences. These learning methods are part of the deep learning framework: deep convolutional neural networks (CNN [Krizhevsky et al., 2012]) for images, and deep recurrent neural networks (LSTM-RNN [Hochreiter and Schmidhuber, 1997]) for texts. To perform a task, a neural network with hidden layers will use these hidden layers to compute a useful representation, which is a latent vector that translates the properties of interest for the task the network was trained on. For example, if one wants to classify images into dogs and cars, a useful representation of the image would be whether or not there are ears, eyes, wheels or glass panes, from which it is easy to learn which class the image corresponds to. However, the representation that would be learned from such a supervised task would be specialized, and might not be useful for other tasks. This is a problem when the task one wants a model for does not have abundant data or

*This article is a draft version of the chapter situated at pages 91-110 in the book "Semantic AI in Knowledge Graphs", CRC Press, 2023. The experiments presented in this paper were carried out using the OSIRIM platform (http://osirim.irit.fr/site/en) that is administered by the Institut de Recherche en Informatique de Toulouse (IRIT) and supported by the French National Center for Scientific Research (CNRS), the Occitanie Region, the French Government, and the European Regional Development Fund (ERDF). The work has benefited from a CISA-IRIT funding.

†CONTACT J. Huteau. Email: jeremie.huteau@gmail.com

when labels are rare and costly: the medical domain is full of such tasks due to data being hard to acquire and labels costly to make.

When the only information is whether two items are related or not (e.g. this image corresponds to this caption, this video sequence follows this sequence), contrastive learning can be used to learn good representations (generic and rich enough to be good starting points) before fine-tuning on a task where labels are sparse/few. In essence, contrastive learning aims to have representations of related elements be close to one another and representations of unrelated elements be far from each other (see [Weng, 2021] for an introduction to the different methods of contrastive learning). Contrastive learning is particularly useful when dealing with data with several modalities (e.g. an image associated with a caption, a video associated with an audio track [Rohrbach et al., 2013]). In this domain, it may be desirable to obtain similar representations for the different modalities of the same object. This common representation can be viewed as a way to capture the meaning of the object represented: this is called semantic embedding. In the case of image and text modalities, the VSE neural architecture [Kiros et al., 2014] uses a CNN and a RNN in parallel in order to obtain a common vector representation for both the image and its associated legend. Once trained on a set of images and legends, the architecture can be used as an encoding-decoding machine in order to produce new legends for unknown images: the legend is generated (decoded) from the vector that encodes the image, or to select images that best suits a new sentence: e.g. select the image from a dataset whose encoding best corresponds to the encoding of the sentence. The idea to use this kind of pipeline comes from the work done in the domain of machine translation [Cho et al., 2014] (with two RNNs), indeed legend generation is a way to translate an image into a description.

Note that obtaining a common representation coming from the two neural networks, i.e. a cross-modality representation, is only possible if the parallels networks are guided with loss functions that aims at making more similar the data referring to the same object, and non-similar the data referring to a different object. This leads to consider the classical method for contrastive learning which aims at minimizing a triplet loss [Schroff et al., 2015] where triplets are composed of an element which can be considered a query, and an element which can be considered a relevant/matching element (w.r.t. the query) and an element which can be considered irrelevant/non-matching. The goal is then to extract a representation from all these elements such that the query and matching elements are more similar to one another than the query and the non matching elements are. The representations are extracted with neural networks, whose parameters are learned through the minimization of the triplet loss. The need to define a very accurate loss function was shown by [Faghri et al., 2018] where these authors propose a new method VSE^{++} in which the architecture is the same as the one of VSE [Kiros et al., 2014], but the loss function is more elaborate. Indeed in VSE^{++}, the authors extend VSE with the explicit introduction of hard negatives in the loss for multi-modal embeddings. They show that their proposal improves the results already obtained on the same datasets: experiments were done on Microsoft COCO dataset [Lin et al., 2015] with an out-performance "on the best reported result of almost 9%" (according to the authors).

The two drawbacks of VSE^{++} are the following: on the one hand, only two modalities are considered namely image and text, while it would be interesting to offer the possibility to express information with other modes like e.g. text in another language, logical formulas, diagrams, etc. On the second hand, we have encountered datasets where the same image is associated with several texts (in COCO dataset, each image is associated with five legends). Due to the way the loss function is defined, VSE^{++} is not able to handle such dataset in its integrality. Indeed, let us consider two texts t_1 and t_2 that are both well adapted with an image i. VSE^{++} can only take into account one of them for being the positive representative of this image, it means that some maybe useful and complementary information is lost.

Several approaches have been proposed for improving the learning of joint representations of vision and language. In [Li et al., 2019], the authors propose to first transform the images into a more structured representation that relates image regions using Graph Convolutional Networks [Welling and Kipf, 2016] to generate features with semantic relationships and then use the triplet

ranking loss of VSE^{++}. In the same vein, Oscar approach [Li et al., 2020b] performs a pre-training in order to obtain a dataset made of triples [word tokens, object tags, region features] by focusing on some salient elements in the images using Faster R-CNN [Ren et al., 2015]. Unicoder-VL[Li et al., 2020a] extends this idea by using three different pre-trainings in order to align the visual and textual modalities: a pre-training using an attention mechanism [Vaswani et al., 2017] is performed in order to learn a "cross-modality contextualized embedding" between regions and word tokens. This pre-training requires first to dispose of the linguistic embeddings (it is done with BERT [Kenton and Toutanova, 2019]: a pre-trained model for language prediction based on attention mechanism) and image embeddings which are obtained by using Faster R-CNN. In more recent approaches the pre-training is done with self attention modules that are used for capturing distant dependencies or heterogeneous interactions between regions [Xue et al., 2021], see [Dou et al., 2022] for an extensive analyse of the different pre-training models on several image-caption datasets. As we can see all these improvements concern the pre-processing of the dataset in order to focus on salient regions or words and these approach only deal with the two modalities image and text. Note that the object tags used in Oscar approach could be viewed as a third modality, but it is obtained by a pre-processing on the initial dataset, hence obtaining this modality is already a part of the learning process. To sum up, as far as we know there are no approach that proposes both to extend the triple loss notion with multiple positives and to accept multiple modalities in the dataset.

In this paper, we propose an approach called MSE** which extends VSE^{++} to many modalities and to the possibility to handle a set of positive items of the same modality for one object. We compare the two methods and we show that despite a better expressivity MSE** gives nearly the same results as VSE^{++} on several datasets. We show that the Loss function of MSE^{**} is more accurate for some hard cases of our dataset.

The rest of the paper is organized as follows, section 2 presents the new MSE** model, section 3 discusses the particular issues raised in this new framework, section 4 gives the implementation details and analyzes the results obtained on the MS-COCO dataset. The last section summarizes the approach and evokes several perspectives.

2 Proposed model: MSE**

In this section, we describe a new model called MSE** that extends the VSE^{++} model. The main feature of these two models is the use of triples consisting of an anchor, a positive element and a negative element and a similarity function between the elements. The goal of the learning process is to learn a representation of each of its elements such that the anchor and the positive element are more similar to each other than the anchor and the negative. We first define the triples and then the loss function used to guide the learning process.

2.1 Definitions and Notations

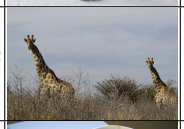
Let us denote by K the number of different modalities contained by a given dataset. For example, for a dataset containing images and text legends, K is equal to 2. We denote by X such a dataset, consisting in I tuples denoted by X_i. Each tuple X_i is formed by K sets denoted by $X_i^k, k = 1..K$, *i.e.*, one set by modality: $X = \left\{X_i \mid X_i = \left(X_i^1, ..., X_i^K\right), i \in \{1...I\}\right\}$

Note that within the ith tuple X_i, a set X_i^k contains one or several elements from the modality k, *i.e.*, $X_i^k = \{x_{i,1}^k, \dots x_{i,|X_i^k|}^k\}$, where $|X_i^k|$ stands for the cardinal of X_i^k.

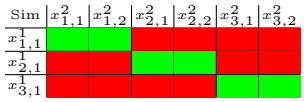
To illustrate these notations, Figure 1.a shows a small example consisting in three images and six legends. In this example, K is equal to 2 (two modalities). One can observe that this dataset contains three tuples, among which the first consists in a set of images $X_1^1 = \{x_{1,1}^1\}$ (in this example only one image corresponds to one tuple) and a set of two captions $X_1^2 = \{x_{1,1}^2, x_{1,2}^2\}$ (the two legends corresponding to the first image).

Figure 1: a) Toy dataset containing three tuples and two modalities. Each tuple contains one image and two legends. For a given element, index i defines the tuple, k its modality, and j its rank in the modality set.
b) Positive (green) and negative (red) legends for each image of this toy example.

a)

$x^k_{i,j}$	$k=1$ $j=1$	$k=2$ $j=1$	$k=2$ $j=2$
$i=1$		A woman in a red jacket skiing down a slope	A woman posing for the camera standing on skis.
$i=2$		Two giraffes during day in field of grasses.	A couple giraffes staring in the same direction
$i=3$		A close up of a cell phone, scissors and a cup.	A cup of coffee, cell phone and scissor sitting on a desk.

b)



In the following, we introduce the notion of triplet, that will be further used to derive the proposed training process.

Definition 2.1 (triplet) *A triplet is composed of:*

- *an anchor $x^=$: any element from any set of any tuple X_i of dataset X;*
- *a positive x^+: any element extracted from a set belonging to the the same tuple X_i as the anchor but from a different modality;*
- *a negative x^-: any element from any set of a different tuple than the anchor, $X_{i'}$ with $i \neq i'$, from a different modality than the anchor.*

Reconsidering the toy example in Figure 1.a, one valid triplet respecting the definition above is $(x^1_{1,1}, x^2_{1,1}, x^2_{3,2})$, containing image $x^1_{1,1}$ as anchor, a legend associated to it as positive and a legend associated to another image as negative. Furthermore, Figure 1.b illustrates how triplets can be formed from this small dataset, by selecting an element of a row as the anchor, an element corresponding to a green cell from the same row as the positive and an element corresponding to any red cell as the negative. Note that by transposing the matrix, the anchor can correspond to a legend instead of an image.

2.2 Loss function

The triplets defined in the previous subsection contain two pairs of interest: the anchor and the positive, hereafter referred to as a positive pair, and the anchor and the negative, called negative pair. In general, one can evaluate how close two representations a and b are using a similarity function Sim, which maps the two representations to a scalar ($Sim(a,b) \in \mathbb{R}$), where *a large value corresponds to a high similarity.* One example of such a function is the cosine similarity, which is the dot product of two unit-vectors. It is worth mentioning that one can also use a distance instead of a similarity function, for which the signs and comparison operators are swapped.

Given a triplet, the objective herein is to have a representation of each of its elements such that *the anchor and the positive are more similar to each other than the anchor and the negative, i.e.*, $Sim(x^=, x^-) < Sim(x^=, x^+)$. Furthemore, this inequality will be stregthen using a margin parameter α such that $Sim(x^=, x^-) + \alpha < Sim(x^=, x^+)$.

The objective of the learning process is therefore to learn a representation function which leads to the least number of violation of this constraint. Unfortunately this constraint is not differentiable (because of the comparison operator) and is therefore not suitable for learning a model through gradient descent. In order to facilitate the training of a neural network, one therefore needs to reformulate this inequality into a differentiable loss function as follows:

$$\mathcal{H}(x^{=}, x^{+}, x^{-}) = max(0, Sim(x^{=}, x^{-}) + \alpha - Sim(x^{=}, x^{+})). \tag{1}$$

This function, called the triplet $\mathcal{H}$inge loss in the related literature, corresponds to the violation of the constraint on pair similarities. In the case where the positive pair is more similar by a margin of α than the negative pair, the loss is 0. Note that with this loss function the constraint is only imposed to be satisfied, *i.e.*, the degree of constraint satisfaction is not quantified. Otherwise, the loss function becomes strictly positive. Starting from this definition of the loss for a triplet, let us define the loss for a given anchor, from which, as explained previously, one can form several triplets by assigning a positive and a negative.

Definition 2.2 *For a given item $x^{=}$ in a set of modality k, X_i^k, of a tuple X_i, we denote by X^+ the set of its positive elements and by X^- the set of its negative elements (with positive and negative elements defined as in Definition 2.1. With these notations, the loss function for a given anchor, $\mathcal{L}(x^{=})$, is given by:*

$$\mathcal{L}(x^{=}) = \mathbb{E}[\mathcal{R}^{+}_{x^{+} \in X^{+}} \mathcal{R}^{-}_{x^{-} \in X^{-}} \mathcal{H}(x^{=}, x^{+}, x^{-})], \tag{2}$$

where R^+ and R^- are functions mapping a set of elements to one of its subset, called reduction functions and $\mathbb{E}$ is the mathematical expectation.

In our case, these reduction functions select elements based on $Sim(x^{=}, x)$, their similarity with the anchor. In the case of selecting all elements (no reduction function), this is equivalent to considering the mean of the triplet losses (the traditional triplet loss). In the case of selecting only the hardest positive and negative (respectively the ones with the lowest and highest similarities to the anchor) this is equivalent to taking the hardest triplet as VSE^{++} loss does.

From the definition of the loss for a given anchor, one can define the loss function for a given modality. The loss for the k-th modality is defined as the mean of the losses of the elements of this modality, normalized by the margin:

$$\mathcal{L}(X^k) = \frac{1}{\alpha} \mathbb{E}_{x^{=} \in X^k}[\mathcal{L}(x^{=})]. \tag{3}$$

Note that the interest of the normalization by the margin is to obtain a (soft) loss bounded between 0 and 1. In practice, the worst case is when the representations of all elements collapse to the same vector, leading to a loss of α for all triplets. Finally, the global loss is the mean of the losses of the k-th sets for each k:

$$\mathcal{L} = \mathbb{E}_k[\mathcal{L}(X^k)] \tag{4}$$

Let us note also that VSE^{++} defines the loss for only one modality and that the global loss is not using the mean of individual losses but rather the sum. The two are are equivalent from an optimization point of view, given that the mean is equal to the sum scaled by a constant factor ($\frac{1}{|X^k|}$ for the loss of the k-th modality $\mathcal{L}(X^k)$, $\frac{1}{K}$ for the global loss $\mathcal{L}$)).

In the general formalism proposed above, we can highlight that VSE^{++} is a particular case where:

- the modality of the anchor and the modality of the positive and negative are different, i.e., cross-modality is imposed;
- there is only one positive by modality i.e. for each tuple i, for each modality k, $|X_i^k| = 1$

- the reduction function uses a maximum: it selects the hardest negative taking $\mathcal{R}^-{}_{x^- \in X^-} = \arg\max_{x^- \in X^-}(Sim(x^=, x^-))$.

The main contribution of VSE^{++} was to compute the loss on the hardest triplet, which in the absence of multiple positives turns to having $\mathcal{R}^- = \arg\max$. The authors of VSE^{++} argued that most negative triplets do not contribute to learning good representations (because they are too easy, i.e., too different from the anchor) while forcing the loss function to local minima in which improving the hard negatives would not help to improve the loss. Therefore, using only the hardest negative example led to better representations.

The main contribution herein is the extension to several positive elements. If the reduction function over positives ($\mathcal{R}^+$) is linear (e.g., a sum), considering multiple positives is equivalent to computing the anchor loss with a different positive each time, *i.e.*, increasing the batch size with same anchor associated to different positive samples. However, similarly to how selecting the hardest negative improves the representations learned by VSE^{++}, we show that selecting the hardest positive has also a positive impact on the representations learned, in particular forcing the representations of the positive to be close to the anchor even in the worst case.

3 Particular issues raised within MSE** framework

In this section, we describe how we dealt with two problems raised by the model: the first is due to the fact that we allow an anchor to be associated with a set of negative elements instead of being associated with a single element, the second concerns the constraints induced by the use of different modalities on the data augmentation.

3.1 Optimizing the hardest negative selection process

Although selecting the hardest negative ultimately leads to better learned representations, the optimization process is more difficult when considering all negatives. This is especially the case at the beginning of the training, where the representations are weak and a collapse to the same representation for all elements can occur, which would be a failed training.

The authors of VSE^{++} identified this issue and proposed to train the model by using the mean reduction for a few epochs before switching to the max reduction. It is however not trivial to choose the number of epochs before switching, and as it is a form of pretraining it could lead to a worse optima at the end of the training compared to using the max reduction from the beginning.

To meet this challenge, we propose to use the top-f operation as reduction function: it consists in selecting the hardest fraction f of the elements, where "hardest" stands for "most similar to the anchor" for the negatives and "least similar" for the positives. When f equals 1, this function is equivalent to the mean, and when f equals 0 it is equivalent to the max (since we always select at least 1 element). For example, for 100 elements, top-f with $f = 0.3$ will return the 30 elements with the largest values.

By decreasing f from 1 to 0 over the course of the training, one can start the training with a loss easy to optimize and gradually progress to a harder but more productive loss, therefore avoiding a collapse into the same representation for all elements. This method is progressively switching the loss function from mean to max. But sometimes the "pretraining" (with the mean reduction) has a long duration before enabling to switch to the max reduction.

While decreasing f linearly over a small (≈ 10000, out of ≈ 250000) number of steps, we noticed that the loss decreased until $f \approx 0.2$, after which it increased to reach 1 when $f = 0$. Therefore we decided to give priority to a decay rule which spends as few steps as possible in the "easy" regime, in order to do as little "pretraining" as possible. Such a behaviour is ensured by the following

hyperbola expression scaled between 1 and 0, with the main "inflection point" at $y = 0.2$:

$$\frac{1 - \text{step\_fraction}}{1 + k \times \text{step\_fraction}} \tag{5}$$

where step_fraction is the fraction of the decay steps (e.g. 10000) done at this point, and $k = 16$.

Note that this approach can be linked to some forms of curriculum learning. Indeed curriculum learning is a form of learning in which a model is trained on tasks in a particular order. The tasks can either be the specific examples that the model has to learn: in that case curriculum means to first learn the easy examples then the hard ones, or qualitatively different tasks [Pentina et al., 2014]: in that case curriculum learning consists in first classifying an object either in animal or machine, then in classifying it either in cat, dog, boat and plane. In our proposal, we do not alter the order in which the samples are presented to the model, but we start with an "easy" task (the mean reduction) and progressively evolve towards the "hard" task (the max reduction).

The way the transition from easy to hard tasks is done using the top-f reduction can be seen as applying binary weights over the elements, and assigning 0 to the weights of the easiest elements as the training goes on. We could instead have used weights between 0 and 1, and have shifted the distribution of these weights towards the hardest elements as the training was going on. This last approach seems more appealing (it avoids to directly take into account the difficulty of the task), but we do not expect it to have much (if any) impact on the accuracy. Therefore, this is left for perspective study.

Even though this careful increase of the difficulty of the task helps with the optimization and makes it possible to train the network, it should be noted that the length of the decay period (how many steps until we reach $f = 0$) is still important: indeed a too short period will still lead to a failed training while a too long one will lead to worse results.

3.2 Data augmentations for multi-modal tuples of sets

Deep neural networks often suffer from overfitting, and one way to reduce it is to use data augmentation. This consist in applying label conservative transformations to the data, which will artificially increase the amount of data. Training on this augmented data can force the network to encounter some conditions which it would not have met if augmentations had not taken place, this generally helps to learn more robust representations. Examples of such transformations for images are horizontal flip (mirror symmetry), cropping some part of the image, rotations, modifying the color, etc. Concerning texts, one can remove some words or replace them with synonyms.

When working with multi-modal data, some transformations need to be applied to all modalities in order to be label conservative. For example, when applying an horizontal flip to an image, any text describing it must have the words "left" and "right" swapped ([Desai and Johnson, 2021]).

When the multi-modal data is made of sets of elements under the same modality, such a transformation should be applied with the same parameters to all elements of each relevant sets. Flipping an image and one of its legend necessarily requires us to flip all the images and legends of the tuple, otherwise we would not preserve left-right positions: a flipped image would not be completely similar to an "unflipped" legend, and trying to match them through the contrastive loss would lead to a network which would be unable to recognize left from right (assuming the flip is done 50% of the time). However, when a transformation is not applied to multiple modalities, it is preferable to apply it with different parameters to the different element of a set. For example, with two images of a tennis ball, modifying the color of the image in the exact same way (say yellow to orange) will result in the tennis ball being the same color in both images. Even though this will still lead to the network having to recognize tennis ball by something else than their color over the course of the training process (and multiple applications of the transformation with different parameters), having the two tennis balls be of different colors will make an update on

the parameters which will concern other properties than its specific color. We argue that this is desirable and we implemented it in our experiments. However we did not run any experiment to confirm it (one would simply need to disable this augmentation parameter difference for the elements of a set) and we think that it will not make any difference on our datasets given the results we obtained.

4 Implementation

In order to test our main hypothesis that *using multiple positives in the triplet loss improves the learned representations*, we had to implement the VSE^{++} model, apply the necessary changes to it and run the experiment on suitable data. This implementation is available via the Github repository https://github.com/JeremieHuteau/adria_internship.

4.1 Dataset

The triplet loss minimization method can be applied to any dataset if we consider an element as the anchor, an augmentation of this element as the positive and a distinct element as the negative. But as we were originally in the context of cross-modal representations, we stuck with the classic dataset for this task: MS-COCO [Lin et al., 2015]. This dataset is suitable for many tasks, but we are mostly interested in the captioning part of the dataset, which contains in its training set about 113 000 images, each described by 5 different captions generated by humans. The validation/test set contains 5000 images, again with 5 captions each. Therefore, only the image modality will have multiple positives.

Due to lack of time to run the experiments, we did not experiment on other datasets. The most suitable one would have been Flicker30K [Young et al., 2014] (similar to MS-COCO but with only 30000 images). As for the medical domain, a dataset can be built from the PEIR library [Jones et al., 2001], which is way smaller with about 5000 images but in which the text modality is having multiple positives (a caption can describe a variable number of images). It should be noted that training a complete visual model for medical images with only 5000 images would not be easy, and using a model trained on natural images as a pretrained base might not be perfect (see [Raghu et al., 2019] for a work on transfer learning for medical imaging).

Let us remark that the MS-COCO dataset is not perfect for our task, as some captions can describe multiple images while only being linked to a single image. For example, a caption stating "A tennis player ready to hit the ball." fits to all images of tennis players in action, but will only be linked to one of them in the dataset. Therefore, many captions which we will consider as negatives are actually false negatives. From a summary exploration of the dataset, we estimate that about one percent of the captions could describe multiple images.

4.2 Model

Even if the theoretical model may use more than two modalities and even identical ones, for comparison purposes, we use the same model architecture as the one in VSE^{++}. The model has to be able to process both images and texts.

For the image modality, a Convolutional Neural Network (CNN) is used to extract a representation. The architecture of this CNN is the one of the ResNet family [He et al., 2015]. Material constraints did not allow us to use the largest model in the ResNet family (ResNet-152) due to memory requirements, so we ran the biggest network that fits on a single NVidia 1080 Ti GPU: namely, ResNet-34. The model we use is pretrained on the ImageNet dataset [Russakovsky et al., 2015].

For the text modality, token embeddings of dimensionality 300 are learned for all the words appearing at least 4 times in the training captions. The captions are then summarized using a Gated Recurrent Unit (GRU)[Cho et al., 2014], which outputs a 1024 dimensional vector. The

embedding vectors are initialized from a normal distribution with mean 0 and variance $\frac{1}{300}$. The exact distribution does not seem to matter, but if the initial values are too large (like they are with the default PyTorch embedding initialization, which uses a variance of 1) the results will be hampered. There are some work related with the importance of embedding initialization [Kocmi and Bojar, 2017] which further explore and explain this behaviour.

When training the model, the image encoding network is frozen (not updated) until some epochs have passed. Therefore, to allow for some flexibility of the image representation, and to help align it with the text representation, and more importantly to make it of the same dimension as the output of the text encoder, a linear projection is appended to the image encoder network. As the text encoder is defined and trained from scratch, it does not need a projection. Before computing the similarities of the representations of images and texts, the vectors are normalized to unit vectors. This, combined with using a dot product as the similarity function, makes it such that the similarity scores are bounded by -1 and 1: the similarity is the cosine similarity. Once the similarity scores are computed, the triplet loss can be computed according to the formulas in 2.1.

4.3 Triplet loss

Regarding the triplet loss using top-f as the reduction function, naive implementations were responsible for about 50% of the total training time, most likely due to transfers between CPU and GPU as well as being inefficient (not vectorized). Vectorizing this function is not trivial due to anchors having potentially different number of positives (and therefore negatives) in a batch. A user from the PyTorch forums came up with a way to vectorize this function [1]. However, this implementation has a cubic time and space complexity with respect to the number of elements in the batch (the time complexity is acceptable due to running on a GPU, but as we were already memory constrained, this space complexity is not acceptable). We ended up using an implementation which computes the loss on groups of anchors having the same number of positives, which has a quadratic complexity both in time and space, and made the time necessary for the computation of the loss negligible compared to the forward and backward crossing of the network (as it should be). An implementation for the max reduction function was created, using the same techniques as the cubic complexity implementations (masking the irrelevant elements to still allow for vectorization), but it ended up by not being used due to not being always applicable and not providing any run-time benefits (the implementation on groups is already fast enough).

4.4 Libraries used

As with most deep learning projects, Python is the language we use. We chose to use PyTorch [Paszke et al., 2019] as the deep learning framework, combined with PyTorch Lightning [Falcon, 2019]. PyTorch is responsible of handling the computations, while PyTorch Lightning is there to provide a standardized way to organize the deep learning code (how to define a model, load data, compute metrics, log, etc). To configure our application, we use Hydra [Yadan, 2019] which makes modifying the configuration of the application easy due to the way individual configurations (e.g. model architecture, data augmentations) can be composed to form the complete application configuration. Finally, we use GNU Make [Stallman et al., 2004] to link the various scripts (environment configuration generation, data preprocessing, model training).

4.5 Model training

We use the training process from the VSE^{++} paper. As a preprocessing step, all images are resized such that their smallest side has a length of 256 pixels. This is done in order to avoid loading full resolution images (that will not be fed to the network anyway). We also create a dictionary that

[1] https://discuss.pytorch.org/t/how-can-i-vectorize-this-for-loop-loss-function/122407/2

Table 1: Validation scores of VSE^{++} and our method (MSE**) on MS-COCO. R@K is the retrieval at K metric: proportion of anchors for which we found positives in the K most similar elements

		Image to Text			Text to Image		
Model	Method	R@1	R@5	R@10	R@1	R@5	R@10
ResNet-34	VSE^{++}	0.31	0.61	0.74	0.23	0.51	0.65
ResNet-34	MSE**	0.31	0.60	0.72	0.21	0.50	0.63
ResNet-101	VSE^{++}	0.37	0.67	0.80	0.27	0.57	0.70
ResNet-101	MSE**	0.36	0.67	0.79	0.26	0.55	0.68

Table 2: Validation scores of VSE^{++} and our method (MSE**) on MS-COCO. Loss is the triplet loss on the validation set, MeanR the mean rank of the positives, and MeanWorstR the mean rank of the hardest positive (only computed for "Image To Text" as captions only have 1 positive).

			Image to Text		Text to Image
Model	Method	Loss	MeanR	MeanWorstR	MeanR
ResNet-34	VSE^{++}	0.98	163	244	35
ResNet-34	MSE**	0.91	177	247	35
ResNet-101	VSE^{++}	0.90	134	470	28
ResNet-101	MSE**	0.85	121	393	27

maps all the words appearing at least 4 times in the captions to an integer (which is used to index the embedding table).

When loading the data, the following transformations are applied:

- Text is put in lower case
- Image is cropped at a random position to a size 224 by 224 pixels
- Image and text are randomly flipped horizontally (mirror symmetry for image, "left"-"right" swap for text)
- Image is normalized according to ImageNet mean and variance of each channel
- Text is tokenized (split into individual words) using NLTK's word tokenizer
- Text is padded with start/end of sentence tokens
- Text tokens are converted to indices using the dictionary made during preprocessing

As described in section 3.2, if an image is (not) flipped then all the captions that describe it are also (not) flipped.

We do not apply data augmentations to texts. Simple augmentations which would most likely improve performance exist [Wei and Zou, 2019]. We could also apply more powerful augmentations to the images, but did not do so to keep our results comparable to the ones from VSE^{++}.

We use a batch size of 128 tuples, which for COCO corresponds to 128 images, and either 128 captions when using one single positive or 640 captions when using all the available positives. We use the Adam [Kingma and Ba, 2015] optimizer. The learning rate is initialized to $2e^{-4}$ and is set to $2e^{-5}$ after 75 epochs ($\approx$ 70000 steps for COCO) have passed. The image encoder (but not the linear projection following it) is frozen (not updated) until 150 epochs have passed, to avoid to "forget" the pretrained knowledge while the network still performs poorly on the task.

4.6 Results and analysis

We are interested in two metrics:

Table 3: Similarities between images 2.a and 2.b with their legends.

Legend	Image 2.a		Image 2.b	
	VSE^{++}	MSE**	VSE^{++}	MSE**
1	0.17	0.57	0.12	0.52
2	0.17	0.53	-0.02	-0.07
3	0.15	0.51	0.10	0.19
4	0.16	0.45	0.13	0.56
5	-0.01	0.15	0.11	0.48
Hardest negative	0.20	0.57	0.17	0.59

- Retrieval at K: it is the fraction of the anchors which have one of their positives in their K most similar elements
- Mean rank: it is the average rank of the positives among the elements of the anchors

We compute the metrics when using images as anchors and texts as anchors, and present the values in Table 1.

Even though the validation loss is lower using multiple positives, we do not observe improvement on the retrieval metrics. We also computed the mean rank of the positive that is returned last, which is what our method is optimized for, and did not find significant improvements there either (244 vs 247 for VSE^{++} vs MSE**).

We suspect that selecting the hardest positive does not improve the performance due to the low number of positive captions (5 on MS-COCO) as well as the simplicity of the captions, which makes it unlikely to find a truly difficult positive (there aren't many ways to describe an image with a short sentence). We expect this method to perform better (relatively to using a single positive / the mean over positives) on datasets containing a lot of variety of positives. One such dataset would be ImageNet, which contains images of concepts in varied scenes, postures, orientations, etc.

Should our method perform better on some other datasets, ablations to identify the important parts would be necessary. The interesting questions to explore would concern the results that we could obtain if we:

- increase the batch size for the single positive method to match the number of anchors of our method (using a single positive, we have 1 anchor per tuple, while the multiple positives ends up using every positive as an anchor).
- use the mean instead of the max for the reduction function over positives (R^+).
- do not use different data augmentations parameters for all the elements of a set.
- use augmentations of an anchor as its positives, instead of the native positives (elements in the dataset tuple of this anchor).

Even if our method is only equivalent in results with VSE^{++}, it seems to produce a better similarity function, as shown on the example of Figure 2:

We notice that MSE** seems to perform better when one legend has typos (Fig.2.a's fifth legend ("kitcken","refigeratator")) and has words absent from the vocabulary or is plain wrong (Fig.2.b's second legend ("horses")). There is a difference in the magnitude of similarities between VSE^{++} and MSE**.

Figure 2: Two images and their five captions from MS-COCO dataset

Image a.

1 There are two refrigerators in this dirty, run-down kitchen.
2 Two refrigerators standing side by side in a room.
3 An olive green refrigerator next to a white refrigerator in an old kitchen.
4 A small refrigerator in a small kitchen with a window.
5 A kitcken that has two different refigeratator in it.

Image b.

1 A herd of elephants standing on top of a field.
2 a number of horses standing near one another
3 Two elephants that are pressing their heads together.
4 A couple of elephants standing in the grass.
5 a couple of elephants out in a large field

5 Conclusion

In this paper we have extended the triplet learning method to the case where there are multiple positive associated to an object, these extension does not improve the accuracy of the representations learned wrt the one obtained by the existing method VSE^{++} in the particular case of the dataset MS-COCO. Nevertheless the implementation is done to improve the development of this method in future works on different datasets.

This work opens several perspectives: 1) use the triplets method with multiple positives on datasets having many examples of each class; 2) use a VSE model to find new positive pairs, and to eliminate false negatives present in some datasets; 3) associate the images not with texts but with logical formulas (in addition to allowing reasoning in post-processing, this would make it possible to introduce degrees of positivity/specificity: if a formula describing an image is more specific than another one, then we will be a priori more intransigent on the respect of the similarity/dissimilarity with this image/another image); generate the most similar captions to the images: is it possible to obtain more exhaustive descriptions of the scenes than those provided by humans ?

References

[Cho et al., 2014] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation.

[Desai and Johnson, 2021] Desai, K. and Johnson, J. (2021). Virtex: Learning visual representations from textual annotations.

[Dou et al., 2022] Dou, Z.-Y., Xu, Y., Gan, Z., Wang, J., Wang, S., Wang, L., Zhu, C., Zhang, P., Yuan, L., Peng, N., et al. (2022). An empirical study of training end-to-end vision-and-language transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18166–18176.

[Faghri et al., 2018] Faghri, F., Fleet, D. J., Kiros, J. R., and Fidler, S. (2018). VSE++: Improving visual-semantic embeddings with hard negatives.

[Falcon, 2019] Falcon, WA, e. a. (2019). Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Jones et al., 2001] Jones, K. N., Woode, D. E., Panizzi, K., and Anderson, P. G. (2001). Peir digital library: Online resources and authoring system. In *AMIA*. AMIA.

[Kenton and Toutanova, 2019] Kenton, J. D. M.-W. C. and Toutanova, L. K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, pages 4171–4186.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Kiros et al., 2014] Kiros, R., Salakhutdinov, R., and Zemel, R. S. (2014). Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.

[Kocmi and Bojar, 2017] Kocmi, T. and Bojar, O. (2017). An exploration of word embedding initialization in deep-learning tasks.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

[Li et al., 2020a] Li, G., Duan, N., Fang, Y., Gong, M., and Jiang, D. (2020a). Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11336–11344.

[Li et al., 2019] Li, K., Zhang, Y., Li, K., Li, Y., and Fu, Y. (2019). Visual semantic reasoning for image-text matching. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4654–4662.

[Li et al., 2020b] Li, X., Yin, X., Li, C., Zhang, P., Hu, X., Zhang, L., Wang, L., Hu, H., Dong, L., Wei, F., et al. (2020b). Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, pages 121–137. Springer.

[Lin et al., 2015] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2015). Microsoft coco: Common objects in context.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle,

H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

[Pentina et al., 2014] Pentina, A., Sharmanska, V., and Lampert, C. H. (2014). Curriculum learning of multiple tasks.

[Raghu et al., 2019] Raghu, M., Zhang, C., Kleinberg, J., and Bengio, S. (2019). Transfusion: Understanding transfer learning for medical imaging.

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

[Rohrbach et al., 2013] Rohrbach, M., Qiu, W., Titov, I., Thater, S., Pinkal, M., and Schiele, B. (2013). Translating video content to natural language descriptions. In *Proceedings of the IEEE international conference on computer vision*, pages 433–440.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge.

[Schroff et al., 2015] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Stallman et al., 2004] Stallman, R. M., McGrath, R., and Smith, P. D. (2004). *GNU Make: A Program for Directing Recompilation, for Version 3.81*. Free Software Foundation.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Wei and Zou, 2019] Wei, J. and Zou, K. (2019). Eda: Easy data augmentation techniques for boosting performance on text classification tasks.

[Welling and Kipf, 2016] Welling, M. and Kipf, T. N. (2016). Semi-supervised classification with graph convolutional networks. In *J. International Conference on Learning Representations (ICLR 2017)*.

[Weng, 2021] Weng, L. (2021). Contrastive representation learning. *lilianweng.github.io/lil-log*.

[Xue et al., 2021] Xue, H., Huang, Y., Liu, B., Peng, H., Fu, J., Li, H., and Luo, J. (2021). Probing inter-modality: Visual parsing with self-attention for vision-and-language pre-training. *Advances in Neural Information Processing Systems*, 34:4514–4528.

[Yadan, 2019] Yadan, O. (2019). Hydra - a framework for elegantly configuring complex applications. Github.

[Young et al., 2014] Young, P., Lai, A., Hodosh, M., and 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.