VSE++: Improving Visual-Semantic Embeddings with Hard Negatives
(2018)

Fartash Faghri, David J. Fleet, Jamie Ryan Kiros, Sanja Fidler
Notes

Contributions

The authors focus on visual-semantic embeddings for cross-modal retrieval; i.e. the retrieval of images given captions, or of captions for a query image. The performance is measured by R@k (recall at K); the fraction of queries for which the correct item is retrieved in the closest K points to the query in the embedding space (often K = 1).

The main contribution is to incorporate hard negatives in the loss function, which is a triplet loss, where we have an anchor, a positive, and a negative example, and the objective is to learn some embeddings such that and anchor and the positive example are close to each other, and the negative example is at learn at margin away. In such cases the choice of negative can be crucial, the harder it is to distinguish between the positive and negative example.

Method

Learning Visual-Semantic Embeddings

A training set contains a set of image-caption pairs $\{(i_n, C_n)\}_{n=1}^{N}$, where (i_n, C_n) are positive pairs and $(i_n, c_{m \neq n})$ are negative pairs. Given a similarity function $s(i,c) \in \mathbb{R}$, we should have higher similarity scores between positive pairs and lower scores between negative scores.

Let $\phi(i)$ be the features computed from image i, which are the features before logits of a convnet, and $\psi(c)$ be the features of a caption c computed using an RNN, these features are then mapped into a joint feature space of dimension D using two linear projections: $f(i) = W_f^T \phi(i)$ and $g(c) = W_g^T \psi(c)$.

The linear projections, and the two features extractors, the CNN and the RNN, are then trained jointly to minimize the loss between set of positive pairs: $\frac{1}{N}\sum_{n=1}^{N} \ell\left(i_n, c_n\right)$. One possible formulation of the loss function is to use hinge based triplet ranking loss:

$$\ell_{SH}(i,c) = \sum_{\hat{c}} [\alpha - s(i,c) + s(i,\hat{c})]_+ + \sum_{\hat{i}} [\alpha - s(i,c) + s(\hat{i},c)]_+$$

With $[x]_+ \equiv \max(x,0)$, the loss above equals zero only if the margin between a given image i and its negatives $\hat{c}$ and a given caption c and its negatives $\hat{i}$ are at least α away from each other. There is some problems with this loss, first of all its quadratic, and given that we try to push all the negatives away at the same time or when a number of negatives with small violations combine to dominate the SH loss, this may have the negative affect of also on the positives.

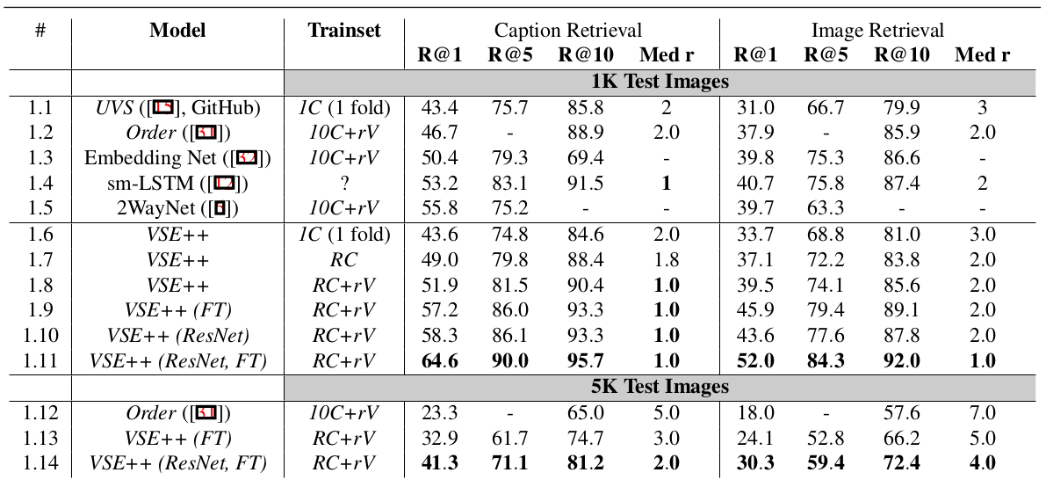
Hard Negatives

Instead of using all the negative in the loss function, only the hardest negative is used in the loss function, a winner takes all the gradients. Given a positive pair (i,c), the hardest negatives are given by $i' = \arg\max_{j \neq i} s(j,c)$ and $c' = \arg\max_{d \neq c} s(i,d)$ and the loss is defined as:

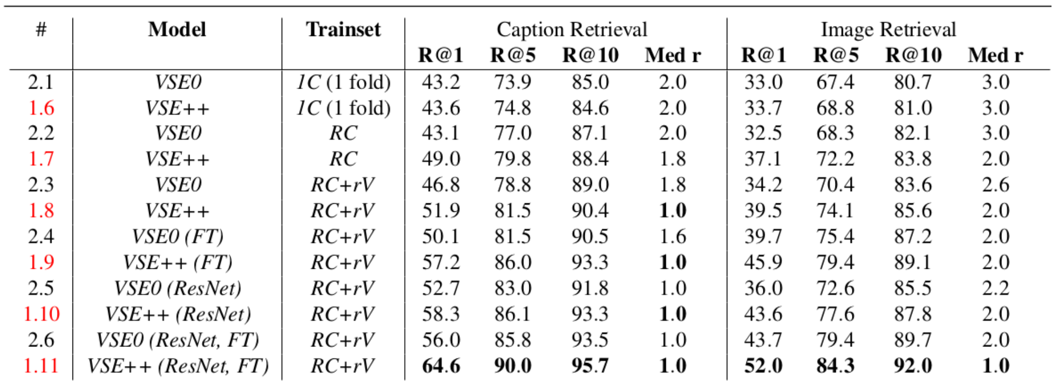
$$\ell_{MH}(i,c) = \max_{c'} \left[\alpha + s\left(i, c'\right) - s(i,c)\right]_+ + \max_{i'} \left[\alpha + s\left(i', c\right) - s(i,c)\right]_+$$

This loss is also quadratic, but the authors propose to find the hardest negatives only in the same mini-batch, instead of the whole dataset. And this approximation gives good results, the larger the batch size is, say we want to find the hardest negatives representing 90th percentile, the probability of not sampling such negatives is $0.9^{(M-1)}$ with M as the batch size. This probability falls bellow 0.1 for a batch of 44. and with a large batch such as 128, we can be confident that there is at least a negative that is harder than the 99.9% of the data points.

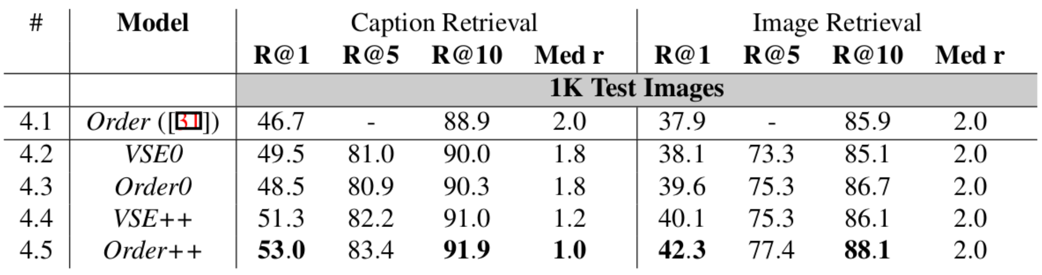
Results



#	Model	Trainset	Caption Retrieval				Image Retrieval			
			R@1	**R@5**	**R@10**	**Med r**	**R@1**	**R@5**	**R@10**	**Med r**
		1K Test Images								
1.1	*UVS* ([15], GitHub)	*1C* (1 fold)	43.4	75.7	85.8	2	31.0	66.7	79.9	3
1.2	*Order* ([31])	*10C+rV*	46.7	-	88.9	2.0	37.9	-	85.9	2.0
1.3	Embedding Net ([32])	*10C+rV*	50.4	79.3	69.4	-	39.8	75.3	86.6	-
1.4	sm-LSTM ([12])	?	53.2	83.1	91.5	**1**	40.7	75.8	87.4	2
1.5	2WayNet ([6])	*10C+rV*	55.8	75.2	-	-	39.7	63.3	-	-
1.6	*VSE++*	*1C* (1 fold)	43.6	74.8	84.6	2.0	33.7	68.8	81.0	3.0
1.7	*VSE++*	*RC*	49.0	79.8	88.4	1.8	37.1	72.2	83.8	2.0
1.8	*VSE++*	*RC+rV*	51.9	81.5	90.4	**1.0**	39.5	74.1	85.6	2.0
1.9	*VSE++ (FT)*	*RC+rV*	57.2	86.0	93.3	**1.0**	45.9	79.4	89.1	2.0
1.10	*VSE++ (ResNet)*	*RC+rV*	58.3	86.1	93.3	**1.0**	43.6	77.6	87.8	2.0
1.11	*VSE++ (ResNet, FT)*	*RC+rV*	**64.6**	**90.0**	**95.7**	**1.0**	**52.0**	**84.3**	**92.0**	**1.0**
		5K Test Images								
1.12	*Order* ([31])	*10C+rV*	23.3	-	65.0	5.0	18.0	-	57.6	7.0
1.13	*VSE++ (FT)*	*RC+rV*	32.9	61.7	74.7	3.0	24.1	52.8	66.2	5.0
1.14	*VSE++ (ResNet, FT)*	*RC+rV*	**41.3**	**71.1**	**81.2**	**2.0**	**30.3**	**59.4**	**72.4**	**4.0**



#	Model	Trainset	Caption Retrieval				Image Retrieval			
			R@1	**R@5**	**R@10**	**Med r**	**R@1**	**R@5**	**R@10**	**Med r**
2.1	*VSE0*	*1C* (1 fold)	43.2	73.9	85.0	2.0	33.0	67.4	80.7	3.0
1.6	*VSE++*	*1C* (1 fold)	43.6	74.8	84.6	2.0	33.7	68.8	81.0	3.0
2.2	*VSE0*	*RC*	43.1	77.0	87.1	2.0	32.5	68.3	82.1	3.0
1.7	*VSE++*	*RC*	49.0	79.8	88.4	1.8	37.1	72.2	83.8	2.0
2.3	*VSE0*	*RC+rV*	46.8	78.8	89.0	1.8	34.2	70.4	83.6	2.6
1.8	*VSE++*	*RC+rV*	51.9	81.5	90.4	**1.0**	39.5	74.1	85.6	2.0
2.4	*VSE0 (FT)*	*RC+rV*	50.1	81.5	90.5	1.6	39.7	75.4	87.2	2.0
1.9	*VSE++ (FT)*	*RC+rV*	57.2	86.0	93.3	**1.0**	45.9	79.4	89.1	2.0
2.5	*VSE0 (ResNet)*	*RC+rV*	52.7	83.0	91.8	1.0	36.0	72.6	85.5	2.2
1.10	*VSE++ (ResNet)*	*RC+rV*	58.3	86.1	93.3	**1.0**	43.6	77.6	87.8	2.0
2.6	*VSE0 (ResNet, FT)*	*RC+rV*	56.0	85.8	93.5	1.0	43.7	79.4	89.7	2.0
1.11	*VSE++ (ResNet, FT)*	*RC+rV*	**64.6**	**90.0**	**95.7**	**1.0**	**52.0**	**84.3**	**92.0**	**1.0**



#	Model	Caption Retrieval				Image Retrieval			
		R@1	**R@5**	**R@10**	**Med r**	**R@1**	**R@5**	**R@10**	**Med r**
		1K Test Images							
4.1	*Order* ([31])	46.7	-	88.9	2.0	37.9	-	85.9	2.0
4.2	*VSE0*	49.5	81.0	90.0	1.8	38.1	73.3	85.1	2.0
4.3	*Order0*	48.5	80.9	90.3	1.8	39.6	75.3	86.7	2.0
4.4	*VSE++*	51.3	82.2	91.0	1.2	40.1	75.3	86.1	2.0
4.5	*Order++*	**53.0**	83.4	**91.9**	**1.0**	**42.3**	77.4	**88.1**	2.0

#	**Model**	**Trainset**	Caption Retrieval				Image Retrieval			
			R@1	**R@5**	**R@10**	**Med r**	**R@1**	**R@5**	**R@10**	**Med r**
3.1	*UVS* ([15])	*1C*	23.0	50.7	62.9	5	16.8	42.0	56.5	8
3.2	*UVS* (GitHub)	*1C*	29.8	58.4	70.5	4	22.0	47.9	59.3	6
3.3	Embedding Net ([32])	*10C*	40.7	69.7	79.2	-	29.2	59.6	71.7	-
3.4	DAN ([24])	?	41.4	73.5	82.5	2	31.8	61.7	72.5	3
3.5	sm-LSTM ([12])	?	42.5	71.9	81.5	2	30.2	60.4	72.3	3
3.6	2WayNet ([6])	*10C*	49.8	67.5	-	-	36.0	55.6	-	-
3.7	DAN (ResNet) ([24])	?	**55.0**	**81.8**	**89.0**	**1**	**39.4**	**69.2**	**79.1**	**2**
3.8	*VSE0*	*1C*	29.8	59.8	71.9	3.0	23.0	48.8	61.0	6.0
3.9	*VSE0*	*RC*	31.6	59.3	71.7	4.0	21.6	50.7	63.8	5.0
3.10	*VSE++*	*1C*	31.9	58.4	68.0	4.0	23.1	49.2	60.7	6.0
3.11	*VSE++*	*RC*	38.6	64.6	74.6	2.0	26.8	54.9	66.8	4.0
3.12	*VSE0 (FT)*	*RC*	37.4	65.4	77.2	3.0	26.8	57.6	69.5	4.0
3.13	*VSE++ (FT)*	*RC*	41.3	69.1	77.9	2.0	31.4	60.0	71.2	3.0
3.14	*VSE0 (ResNet)*	*RC*	36.6	67.3	78.4	3.0	23.3	52.6	66.0	5.0
3.15	*VSE++ (ResNet)*	*RC*	43.7	71.9	82.1	2.0	32.3	60.9	72.1	3.0
3.16	*VSE0 (ResNet, FT)*	*RC*	42.1	73.2	84.0	2.0	31.8	62.6	74.1	3.0
3.17	*VSE++ (ResNet, FT)*	*RC*	52.9	80.5	87.2	1.0	**39.6**	**70.1**	**79.5**	**2.0**