Supplementary Materials: Learning the Best Pooling Strategy for Visual Semantic Embedding

Jiacheng Chen[1*] Hexiang Hu[2*] Hao Wu[1] Yuning Jiang[3] Changhu Wang[1]
[1]ByteDance AI Lab [2]University of Southern California [3]Alibaba Inc

In this Supplementary Material , we provide implementation details and experiments omitted in the main text. The content is organized as follows:

1. Additional implementation details, including model settings, details of reproducing baseline methods, training setups for different experiments.
2. More experiments and results, including
   - Ablation studies for better understanding GPO, including the effects of GPO on each modality alone, the importance of the Size Augmentation, and the choice of the sequence model for GPO implementation.
   - Extensions of Table 2 of the main text, with full COCO 5K evaluation results, more combinations of feature backbones, and more related baselines.
   - A synthetic experiment for further verifying the design choices of GPO, as well as motivating the Size Augmentation.
   - Experiments for exploring more complex variants of GPO (*i.e.*, data-dependent pooling and per-dimensional pooling), which provide potential explanations for the failure of complex feature aggregators.

1. Additional Implementation Details

1.1. Image-text Matching

Model settings The dimensionality d_3 of the joint embedding space is 1024 for all experiments. Note when $d_1 \neq d_3$ or $d_2 \neq d_3$, MLPs are applied before GPO to transform the dimensionality of features. The CNN backbones used in our experiments are either ResNet [8] or ResNeXt101 [20], thus d_1 is 2048. For text backbones, we set $d_2 = 1024$ for BiGRU, and pre-trained BERT has default d_2 being 768. We convert the officially released BUTD CNN from Caffe to Pytorch for running experiments related to *Grid* feature. When using *Grid* feature, the dimensionality transformation only uses a single linear layer since the image branch already has a massive CNN. When using *Region* feature, the transformation uses a two-layer MLP with residual link.

Training details The VSE models for image-text matching are trained with AdamW optimizer with weight decay factor 10e-4. The batch size is always 128 and the margin α of the triplet ranking loss is 0.2. The initial learning rate is 5e-4 while different model components have different learning rate multiplier: (1) CNNs pre-trained on ImageNet: 0.1; (2) BUTD or WSL CNNs: 0.01; (3) BERT: 0.1. The models are trained for 25 epochs and the learning rate decays by a factor of 10 for the last 10 epochs. When fine-tuning CNN backbones (*i.e.*, for *Grid* feature), we fix the running statistics of the Batch Normalization layers during the training.

Two warm-up strategies are used during training: (1) At the first epoch, the parameter of the ConvNet is not trained (only for *Grid* feature), and the triplet ranking loss uses all negative examples in the batch instead of using only the hardest negative example; (2) Starting from the second epoch, all parameters are trained end-to-end with Eq. 1 of the main text, and linear learning rate warmup is used.

All experiments are implemented with PyTorch v1.2.0 and run on Tesla V100 PCI-E GPU.

Fixes of CVSE [18] for fair comparison We notice that the official implementation of CVSE [18] shows two unfair experimental setups compared to other image-text matching methods in the literature:

1. The "concept labels"(*i.e.*, words with semantic meaning) are provided as extra input for the model, which is one of the core contributions of CVSE. However, for each text input, its "concept labels" actually come from *all five ground-truth captions* (in COCO and Flickr30k, each image is associated with five captions). This is an unfair information leak compared to other methods, as it makes the model leverage information from five captions to match one image. The valid "concept labels" of each caption should only contain labels from itself.
2. Previous image-text matching methods report COCO 1K results by averaging over five 1K data folds of the test set, but CVSE is only evaluated on the first 1K fold.

We use the official released code[1] to re-train and re-evaluate

*Authors contributed equally

[1] https://github.com/BruceW91/CVSE

Table 1. Variants of VSE∞ on BUTD region feature and BERT, with different aggregator combintations.

Data Split		COCO 5-fold 1K Test [4]			
Eval Task		IMG → TEXT		TEXT → IMG	
Visual Aggr.	Text Aggr.	R@1	R@5	R@1	R@5
AvgPool	CLS	67.7	92.6	54.8	85.1
AvgPool	GPO	71.7	93.8	57.5	87.6
GPO	CLS	73.8	94.8	59.2	88.0
GPO	GPO	79.7	96.4	64.8	91.4

the model while fixing the above two setups, and report the results in the main paper.

1.2. Video-text Matching

Model settings & Training details We follow the official implementation of HGR [3][2] in the video-text matching experiments and plug in our GPO as the video and text feature aggregator. The same as the image-text matching experiments, the dimension of the joint embedding space is 1024 and the margin α of triplet ranking loss is 0.2. All video-text models are trained for 35 epochs with batch size 128. The initial learning rate is 1e-4, and the learning rate decays by a factor of 10 for the last 10 epochs.

We found that re-running the video-text VSE++ baseline in the official HGR code produces obviously higher results compared to those reported in the original paper, thus we used our re-running results to compare VSE++ and VSE∞.

1.3. Setups of Figure 3

To measure the speed of text-based image retrieval with VSE and V+L BERT in Figure 3, we first pre-compute all features/embeddings that are not conditioned on the text query (*i.e.*, holistic embeddings for VSE, and BUTD region features for V+L BERT). Then we compute the similarity scores between the text query and all image candidates. For VSE models, all similarity scores are calculated with a large matrix multiplication, while for V+L BERT, we need to forward the BERT model for n times where n is the number of image candidates. We tune the batch size so that the GPO memory is fully utilized.

2. Additional Experiments and Results

2.1. More Ablation Studies

(1) Do we need GPO for both visual and text features? In Table 1, we use GPO to replace the standard pooling function for BUTD image region feature and BERT text feature (*i.e.*, AvgPool and CLS). The comparisons confirm that GPO is effective for either modality alone, using GPO for both modalities can further improve the results by a large margin.

[2] https://github.com/cshizhe/hgr_v2t

Table 2. The Size Augmentation's effect on different modalities.

Data Split		COCO 5-fold 1K Test [4]			
Eval Task		IMG → TEXT		TEXT → IMG	
Features	Size Aug.	R@1	R@5	R@1	R@5
BUTD (Region) + BiGRU	∅	74.6	95.0	60.6	89.1
	visual	75.4	95.5	61.4	89.5
	visual+text	78.5	96.0	61.7	90.3

Table 3. Different choices of the sequence model used by GPO

Data Split		COCO 5-fold 1K Test [4]			
Eval Task		IMG → TEXT		TEXT → IMG	
Features	Seq. Model	R@1	R@5	R@1	R@5
BUTD (Region) + BiGRU	Transformer	77.4	95.4	61.4	90.1
	BiGRU	78.5	96.0	61.7	90.3
BUTD (Grid) + BiGRU	Transformer	76.6	95.7	62.7	90.7
	BiGRU	78.0	95.8	62.6	90.6

(2) Effectiveness of Size Augmentation Table 2 shows the effectiveness of Size Augmentation. The random dropping of either visual and text inputs boost the multi-modal retrieval performance. A synthetic experiment in Supplementary Material provides more motivations for this augmentation strategy. Surprisingly, we find this strategy also improves the baseline VSE++ [6], potentially as a regularization method. State-of-the-art video-text matching model [3] uses a similar strategy, feature dropout, which adds a dropout layer for input features. The difference is that it randomly set feature values to 0, while Size Augmentation randomly drops entire elements from the feature set.

(3) Different choices of the sequence model in GPO We also try using different sequence model to implement GPO. Table 3 shows that using a Transformer Encoder [17] to replace the simple BiGRU does not yield improvements on two different combinations of features. The sequence model of GPO only takes the positional information as the input without using the exact feature vectors, thus we believe it's not necessary for the sequence model to have large capacity. A simple BiGRU will suffice for both capacity and computational efficiency, and more complex mechanisms like the multi-head attentions of Transformers could even hurt the performance.

2.2. More results and comparisons for image-text matching

In Table 4 and Table 5, we provide results extending Table 2 of the main text. Grid feature with ImageNet-pretrained ResNet-152 was the standard image backbone for image-text matching before BUTD region features became popular. It is worth noting that our re-implementation of VSE++ improves the original VSE++ by a large margin, by increasing the image size from 224×224 to 512×512 as guided by the

Table 4. Extension of Table 2 of the main text, with more image-text matching results on COCO and Flickr30K, using different visual and textual backbones (denoted by **bold section title**). ⋆: Ensemble results of two models; on IN/IN+VG: Models pre-trained on ImageNet [16], ImageNet and VisualGenome [12], respectively. The best and second best results (in RSUM) are marked **bold** in **red** and **black**.

Data Split		COCO 5-fold 1K Test [4]							Flickr30K 1K Test [21]						
Eval Task		IMG → TEXT			TEXT → IMG				IMG → TEXT			TEXT → IMG			
Method	Feature Type	R@1	R@5	R@10	R@1	R@5	R@10	RSUM	R@1	R@5	R@10	R@1	R@5	R@10	RSUM
ResNet-152 on IN [8] + BiGRU															
UVS[11]$_{2014}$	Grid	56.0	85.8	93.5	43.7	79.4	89.7	448.1	42.1	73.2	84.0	31.8	62.6	74.1	367.8
VSE++[6]$_{2017}$	Grid	64.6	90.0	95.7	52.0	84.3	92.0	478.6	52.9	80.5	87.2	39.6	70.1	79.5	409.8
SCO[9]$_{2018}$	Grid	69.9	92.9	97.5	56.7	87.5	94.8	499.3	55.5	82.0	89.3	41.1	70.5	80.1	418.5
GXN⋆[7]$_{2018}$	Grid	68.5	-	97.9	56.6	-	94.5	-	56.8	-	89.6	41.5	-	80.1	-
Our: VSE++	Grid	70.9	93.4	97.5	58.2	87.1	93.5	**500.6**	64.4	87.3	93.1	49.3	77.5	84.7	**456.3**
Our: VSE∞	Grid	76.5	95.3	98.5	62.9	90.6	95.8	**519.6**	77.1	94.5	97.1	58.5	84.1	89.6	**500.9**
ResNet-101 Faster-RCNN on IN+VG (BUTD) [1] + BiGRU															
SCAN⋆[13]$_{2018}$	Region	72.7	94.8	98.4	58.8	88.4	94.8	507.9	67.4	90.3	95.8	48.6	77.7	85.2	465.0
LIWE [19]$_{2019}$	Region	73.2	95.5	98.2	57.9	88.3	94.5	507.6	69.6	90.3	95.6	51.2	80.4	87.2	474.3
VSRN⋆[14]$_{2019}$	Region	76.2	94.8	98.2	62.8	89.7	95.1	516.8	71.3	90.6	96.0	54.7	81.8	88.2	482.6
CVSE [18]$_{2020}$	Region	69.2	93.3	97.5	55.7	86.9	93.8	496.4	70.5	88.0	92.7	54.7	82.2	88.6	476.7
CAAN [22]$_{2020}$	Region	75.5	95.4	98.5	61.3	89.7	95.2	515.6	70.1	91.6	97.2	52.8	79.0	87.9	478.6
IMRAM⋆ [2]$_{2020}$	Region	76.7	95.6	98.5	61.7	89.1	95.0	516.6	74.1	93.0	96.6	53.9	79.4	87.2	484.2
Our: VSE++	Region	68.5	92.6	97.1	54.0	85.6	92.7	490.5	62.2	86.6	92.3	45.7	73.6	81.9	442.3
Our: VSE∞	Region	78.5	96.0	98.7	61.7	90.3	95.6	520.8	76.5	94.2	97.7	56.4	83.4	89.9	498.1
Our: VSE∞	Grid	78.0	95.8	98.5	62.6	90.6	96.0	**521.5**	77.9	93.7	97.4	57.5	83.4	90.2	**500.2**
Our: VSE∞	Region+Grid	80.0	97.0	99.0	64.8	91.6	96.5	**528.8**	80.7	96.4	98.3	60.8	86.3	92.3	**514.8**
ResNet-101 Faster-RCNN on IN+VG (BUTD) [1] + BERT [5]															
Our: VSE++	Region	67.9	91.9	97.0	54.0	85.6	92.5	488.9	63.4	87.2	92.7	45.6	76.4	84.4	449.7
Our: VSE∞	Region	79.7	96.4	98.9	64.8	91.4	96.3	527.5	81.7	95.4	97.6	61.4	85.9	91.5	513.5
Our: VSE∞	Grid	80.4	96.8	99.1	66.4	92.1	96.7	**531.6**	81.5	97.1	98.5	63.7	88.3	93.2	**522.3**
Our: VSE∞	Region+Grid	82.2	97.5	99.5	68.1	92.9	97.2	**537.4**	85.3	97.2	98.9	66.7	89.9	94.0	**532.0**
ResNeXT-101 on IG (WSL) [15] + BERT [5]															
Our: VSE++	Grid	79.6	97.1	99.0	66.4	91.1	95.5	528.7	80.9	96.6	98.9	65.2	89.5	93.7	524.8
Our: VSE∞	Grid	84.5	98.1	99.4	72.0	93.9	97.5	**545.4**	88.4	98.3	99.5	74.2	93.7	96.8	**550.9**
Our: VSE∞ ⋆	Grid	85.6	98.0	99.4	73.1	94.3	97.7	**548.1**	88.7	98.9	99.8	76.1	94.5	97.1	**555.1**

empirical study of [10]. VSE∞ consistently outperforms the improved VSE++ and other baselines on ResNet-152 grid features.

We also include results of three non-VSE methods: SCAN [13], CAAN [22], and IMRAM [2]. These methods rely on fine-grained cross-modality interactions to match image and text, and all of them use BUTD and BiGRU as the feature extractors. Under the same experimental setups, VSE∞ outperforms them without any complex cross-modal modeling.

2.3. Synthetic Experiment for Verifying GPO Design

We further verify the architecture of coefficient generator $g(\cdot,\cdot)$ using synthetic data and pre-determined pooling coefficients (*e.g.*, coefficients for the K-MaxPool with $K = 5$). As a concrete example, we generate a set of random feature vectors as the input to GPO, and then use the pre-determined "ground-truth" pooling strategy to generate the "ground-truth" output. Such synthetic input-output pairs are then used for learning a GPO module. As for evaluation, we took the coefficient generator from GPO and compare the predicted pooling coefficients to the "ground-truth" ones. We report the results in Root Mean Square Error (RMSE).

Evaluation Protocol We designed four types of synthetic pooling patterns: (1). AvgPool (denoted as A), (2). K-MaxPool (denoted as M-K), (3). Top-K% Pooling (denoted as T-K%), and (4). Linearly-decayed pooling weights (denoted as L), in which the pooling weight for the k-th maximum linearly decreases to zero as k goes from 1 to N. To better assess generalization, we set the training feature set sizes to range from 20 to 100, and we evaluate models on test data with the feature set size ranging from 10 to 120. We report results on both SEEN and UNSEEN feature set sizes to investigate GPO's generalization performance.

Table 5. Extension of Table 2 of the main text, with image-text matching results on COCO 5K. ⋆: Ensemble results of two models.

Data Split		COCO 5K Test [4]						
Eval Task		IMG → TEXT			TEXT → IMG			
Method	Feature Type	R@1	R@5	R@10	R@1	R@5	R@10	RSUM
ResNet-152 on IN [8] + BiGRU								
VSE++[6]$_{2017}$	Grid	41.3	71.1	81.2	30.3	59.4	72.4	355.7
SCO[9]$_{2018}$	Grid	42.8	72.3	83.0	33.1	62.9	75.5	369.6
GXN⋆[7]$_{2018}$	Grid	42.0	-	84.7	31.7	-	74.6	-
Our: VSE++	Grid	46.1	76.8	86.6	35.2	65.6	77.3	**387.6**
Our: VSE∞	Grid	55.1	81.9	89.9	40.9	70.6	81.5	**419.9**
ResNet-101 Faster-RCNN on IN+VG (BUTD) [1] + BiGRU								
SCAN⋆[13]$_{2018}$	Region	50.4	82.2	90.0	38.6	69.3	80.4	410.9
VSRN⋆[14]$_{2019}$	Region	53.0	81.1	89.4	40.5	70.6	81.1	415.7
CAAN [22]$_{2020}$	Region	52.5	83.3	90.9	41.2	70.3	82.9	421.1
IMRAM⋆ [2]$_{2020}$	Region	53.7	83.2	91.0	39.7	69.1	79.8	416.5
Our: VSE++	Region	42.9	74.5	85.1	31.7	61.8	74.2	370.2
Our: VSE∞	Region	56.6	83.6	91.4	39.3	69.9	81.1	421.9
Our: VSE∞	Grid	56.2	83.7	90.9	40.8	70.6	81.5	**423.7**
Our: VSE∞	Region+Grid	59.8	86.1	92.8	42.7	72.8	83.3	**437.5**
ResNet-101 Faster-RCNN on IN+VG (BUTD) [1] + BERT [5]								
Our: VSE++	Region	42.1	72.6	83.9	31.0	61.3	73.7	364.7
Our: VSE∞	Region	58.3	85.3	92.3	42.4	72.7	83.2	434.3
Our: VSE∞	Grid	59.1	85.9	92.8	44.1	74.1	84.0	**440.0**
Our: VSE∞	Region+Grid	62.5	87.8	94.0	46.0	75.8	85.7	**451.8**
ResNeXT-101 on IG (WSL) [15] + BERT [5]								
Our: VSE++	Grid	57.9	85.2	92.8	44.9	74.5	84.0	439.2
Our: VSE∞	Grid	66.4	89.3	94.6	51.6	79.3	87.6	**468.9**
Our: VSE∞ ⋆	Grid	68.1	90.2	95.2	52.7	80.2	88.3	**474.8**

Different GPO Designs We compare different design choices of the architecture for $g(\cdot, \cdot)$, including:

- **Cos/Sin+BiGRU** This is the design introduced in the main paper, which uses the positional encoding and learn a BiGRU as the coefficients generator.
- **Interp.** Learn a fixed size vector as the pooling coefficients, and use linear interpolation to get the pooling coefficients for various lengths N. FSPool [23] uses this type of design to handle variable-length inputs.
- **Cos/Sin+MLP** Instead of using a sequence model to handle variable-size features, simply using a MLP to map positional encodings into pooling coefficients. This design assumes the weight generation process for each index k is unaware of the global length.
- **Index+BiGRU** This model transforms the position index into embeddings with a learnable matrix, and learns a BiGRU as the coefficients generator. Without the positional encoding, this design applies no prior knowledge to the ordinal position indices.

Results Table 6 presents the results comparing different design of GPOs. *Cos/Sin+BiGRU* achieves the best overall performances. *Interp.* has clear difficulty in handling K-Max Pooling. *Index+BiGRU* produces slightly worse results, which shows the advantage of using *Cos/Sin* positional encoding. Moreover, generalizing to unseen feature sizes is indeed challenging for GPO, and generalizing to smaller feature sizes is harder than generalizing to larger feature sizes. To make GPO better generalize to inputs with different sizes, we propose the Size Augmentation as discussed in § 3 of the main paper.

2.4. Complex Variants of GPO

We have kept a simple architecture for GPO so that it only adds marginal extra computational cost to VSE models. However, it is worthwhile to verify whether more complex variants of GPO can indeed produce better results. We investigate two modifications:

Are per-dimension pooling coefficients helpful? Instead of generating shared pooling coefficients for all dimensions, we try to generate coefficients for each dimension separately. Per-dimensional pooling has stronger capacity and might improve the performance, like how FSPool [23] is designed. However, results in Table 7 disproves the above statement in the context of VSE∞. The per-dimension variant of GPO

Table 6. Comparisons of different GPO designs on synthetic patterns. Results are reported in RMSE (lower the better).

Input Repr.	Decoder	SEEN SIZES					UNSEEN SIZES (smaller / larger)				
		A	M-1	M-10	T-50%	L	A	M-1	M-10	T-50%	L
Index	Interp.	**0**	.065	.030	.011	.004	**0/0**	.017/.070	.141/.014	**.039**/.005	.016/.002
Cos/Sin	MLP	.012	.003	.047	.032	.022	.054/.006	**.003**/.003	.093/.030	.096/.026	.065/.028
Index	BiGRU	.002	**.002**	.026	.011	.008	.006/.003	.004/**.001**	.052/.018	.049/.007	.015/.005
Cos/Sin	BiGRU	**0**	.005	**.010**	**.006**	**0**	.002/**0**	.010/.004	**.031/.007**	.046/**.004**	**.005/.001**

Table 7. Variants of GPO w/ or w/o per-dimension coefficients.

Data Split		COCO 5-fold 1K Test [4]			
Eval Task		IMG → TEXT		TEXT → IMG	
Features	Per-dim?	R@1	R@5	R@1	R@5
BUTD (Region)	✓	76.5	95.9	61.2	89.6
+ BiGRU	✗	78.5	96.0	61.7	90.3
BUTD (Region)	✓	79.3	96.1	64.7	91.3
+ BERT	✗	79.7	96.4	64.8	91.4

Table 8. Variants of GPO w/ or w/o per-dimension coefficients.

Data Split		COCO 5-fold 1K Test [4]			
Eval Task		IMG → TEXT		TEXT → IMG	
Features	Feature to $g(\cdot,\cdot)$?	R@1	R@5	R@1	R@5
BUTD (Region)	✓	78.0	96.2	61.8	90.2
+ BiGRU	✗	78.5	96.0	61.7	90.3
BUTD (Region)	✓	79.2	96.3	64.7	91.2
+ BERT	✗	79.7	96.4	64.8	91.4

provides no improvements over two combinations of feature extractors, potentially due to over-fitting.

Would GPO be better if $g(\cdot,\cdot)$ also takes feature as input? Another possible modification to GPO is to input both the feature itself and the position index into the coefficient generator $g(\cdot,\cdot)$. With this modification, GPO can be considered as a special form self attention. By intuition, this modified GPO can adaptively change the pooling coefficients according to the exact feature values. However, Table 8 shows that this modification does not bring improvements. Position index along suffices for generating good pooling coefficients.

In § 3.1 of the main paper, we observe that complex aggregators cannot outperform well-selected simple pooling function, and the above two experiments again show that complicated feature aggregation does not necessarily improve VSE models. A possible explanation for these experimental results is that: feature extractors have provided adequate information for multi-modal matching, so the feature aggregators do not have to further contextualize the feature vectors. Too complicated models for feature contextualization might increase the risk of over-fitting and hurt the performance at the end.

References

[1] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 3, 4

[2] H. Chen, G. Ding, Xudong Liu, Zijia Lin, J. Liu, and J. Han. Imram: Iterative matching with recurrent attention memory for cross-modal image-text retrieval. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12652–12660, 2020. 3, 4

[3] Shizhe Chen, Yida Zhao, Qin Jin, and Qi Wu. Fine-grained video-text retrieval with hierarchical graph reasoning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2

[4] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015. 2, 3, 4, 5

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019. 3, 4

[6] Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. Vse++: Improved visual-semantic embeddings. In *BMVC*, 2017. 2, 3, 4

[7] Jiuxiang Gu, Jianfei Cai, Shafiq R Joty, Li Niu, and Gang Wang. Look, imagine and match: Improving textual-visual cross-modal retrieval with generative models. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 3, 4

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015. 1, 3, 4

[9] Yan Huang, Qi Wu, Chunfeng Song, and Liang Wang. Learning semantic concepts and order for image and sentence matching. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 3, 4

[10] Huaizu Jiang, Ishan Misra, Marcus Rohrbach, Erik G. Learned-Miller, and Xinlei Chen. In defense of grid features for visual question answering. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 3

[11] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *NeurIPS Workshop Deep Learning*, 2014. 3

[12] Ranjay Krishna, Yuke Zhu, Oliver Groth, J. M. Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-

Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 123:32–73, 2016. 3

[13] Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. Stacked cross attention for image-text matching. In *Eur. Conf. Comput. Vis.*, 2018. 3, 4

[14] Kunpeng Li, Yulun Zhang, Kai Li, Yuanyuan Li, and Yun Fu. Visual semantic reasoning for image-text matching. In *Int. Conf. Comput. Vis.*, 2019. 3, 4

[15] Dhruv Kumar Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *Eur. Conf. Comput. Vis.*, 2018. 3, 4

[16] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *IJCV*, 115:211–252, 2014. 3

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Adv. Neural Inform. Process. Syst.*, 2017. 2

[18] Haoran Wang, Ying Zhang, Zhong Ji, Yanwei Pang, and Lin Ma. Consensus-aware visual-semantic embedding for image-text matching. *Eur. Conf. Comput. Vis.*, 2020. 1, 3

[19] Jonatas Wehrmann, Mauricio A. Lopes, Douglas M. Souza, and Rodrigo C. Barros. Language-agnostic visual-semantic embeddings. In *Int. Conf. Comput. Vis.*, pages 5803–5812, 2019. 3

[20] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016. 1

[21] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL*, 2:67–78, 2014. 3

[22] Qi Zhang, Zhen Lei, Zhaoxiang Zhang, and S. Li. Context-aware attention network for image-text retrieval. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3533–3542, 2020. 3, 4

[23] Y. Zhang, Jonathon S. Hare, and A. Prügel-Bennett. Fspool: Learning set representations with featurewise sort pooling. *Int. Conf. Learn. Represent.*, 2020. 4