

Text-Based Person Search with Limited Data (Supplementary Material)

Xiao Han^{1,2}

xiao.han@surrey.ac.uk

Sen He^{1,2}

sen.he@surrey.ac.uk

Li Zhang³

lizhangfd@fudan.edu.cn

Tao Xiang^{1,2}

t.xiang@surrey.ac.uk

¹ CVSSP

University of Surrey
Guildford, UK

² iFlyTek-Surrey Joint Research Centre
on Artificial Intelligence

³ School of Data Science
Fudan University
Shanghai, China

A Testing CLIP on person image classification

As illustrated in Figure 1, CLIP cannot distinguish between fine-grained features when we test it on zero-shot fine-grained person image classification. In this toy experiment, each label consists of one color and one garment, *e.g.*, "the color of her bag is orange". However, CLIP tends to predict almost all mentioned garments as orange, demonstrating that it cannot focus on fine-grained information well.

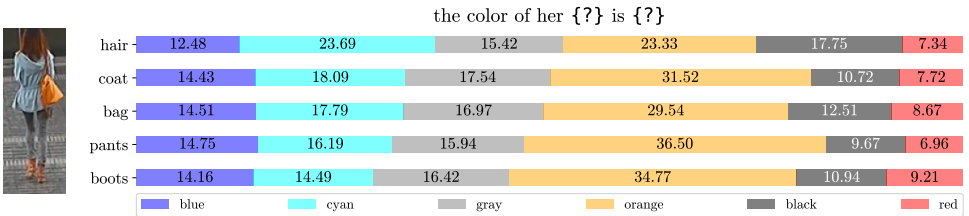


Figure 1: Visualization of the probabilities predicted by CLIP [11] for fine-grained zero-shot person image classification.

An intuitive explanation to this phenomenon is that CLIP is trained to distinguish different visual classes using text, which is limited for intra-class discrimination in TBPS. However, as it can distinguish different visual classes using a single word (representing the class label), it thus learns an informative cross-modal representation for each word. Therefore, we use the text encoder of CLIP to embed words in each sentence, and then append a Bi-GRU to contextualize them.

B Training details

B.1 Modified ResNet101 for CLIP image encoder

According to CLIP [11], this modified ResNet has three improvements over the vanilla version: (1) There are now 3 stem convolutions as opposed to 1 with an average pooling instead of max pooling. (2) It performs anti-aliasing strided convolutions, where an average pooling is prepended to convolutions with stride greater than 1. (3) The final pooling layer is a self-attention pooling instead of a global average pooling.

B.2 Pseudocode of CM-MoCo in Pytorch-style

To better demonstrate our proposed CM-MoCo, we provide a pseudocode in Pytorch-style as following.

```
# f_v_q, f_v_k: encoder networks for visual query and key
# f_t_q, f_t_k: encoder networks for textual query and key
# queue_t, queue_v, queue_id: queues to store K keys
# m: momentum (0.999)
# t: temperature (0.07)
# -----
# bmm: batch matrix multiplication
# mm: matrix multiplication
# cat: concatenation
# complement: get complement set

f_v_k.params, f_t_k.params = f_v_q.params, f_t_q.params # initialize

for v, t, pid in loader: # load a batch data with B samples
    v_q = f_v_q.forward(v) # visual queries: BxD
    t_q = f_t_q.forward(t) # textual queries: BxD
    v_k = f_v_k.forward(v) # visual keys: BxD
    t_k = f_t_k.forward(t) # textual keys: BxD

    # stop gradients for keys
    v_k, t_k = v_k.detach(), t_k.detach()

    # positive logits: Bx1
    v_pos = bmm(v_q.view(B, 1, D), t_k.view(B, D, 1))
    t_pos = bmm(t_q.view(B, 1, D), v_k.view(B, D, 1))

    # get P indexes of the positive instances in the queue,
    # whose identity exist in the current batch
    pos_idx = queue_id.expand(B, K).eq(pid.unsqueeze(-1)).nonzero()[ :, 1]
    neg_idx = arange(K).complement(pos_idx) # negative indexes: K-P

    # negative logits: Bx(K-P)
    v_neg = mm(v_q.view(B, D), queue_t.view(D, K))[ :, neg_idx]
    t_neg = mm(t_q.view(B, D), queue_v.view(D, K))[ :, neg_idx]

    # logits: Bx(1+K-P)
    logits_v = cat([v_pos, v_neg], dim=1)
    logits_t = cat([t_pos, t_neg], dim=1)

    # contrastive loss
    labels = zeros(B) # positives are the 0-th
    loss = CrossEntropyLoss(logits_v / t, labels) \
```

```

+ CrossEntropyLoss(logits_t / t, labels)

# gradient update
loss.backward()

# momentum update
f_v_k.params = m * f_v_k.params + (1 - m) * f_v_k.params
f_t_k.params = m * f_t_k.params + (1 - m) * f_t_k.params

# update queues
enqueue(queue_v, v_k) # enqueue the current batch
enqueue(queue_t, t_k)
enqueue(queue_id, pid)
dequeue(queue_v)      # dequeue the earliest batch
dequeue(queue_t)
dequeue(queue_id)

```

B.3 Alignment loss

We discard the widely used CMPM loss [16] and utilize the logistic-based contrastive loss proposed in ViTAA [15] as our cross-modal alignment loss. Particularly, for the visual side, given an image q-feature \mathbf{V}_i^q and a batch of text q-features \mathbf{T}_q , the cross-modal cosine similarity \mathbf{S}_i is calculated by $\mathbf{S}_i = \mathbf{V}_i^q \otimes \mathbf{T}_q^T$, where $\mathbf{S}_i \in \mathbb{R}^B$ and \otimes denotes matrix multiplication. For the textual side, the calculation is identical and implemented by multiplying the alignment loss by 2. The alignment loss is finally defined as following formula 1, where $\mathbf{S}_i^+/\mathbf{S}_i^-$, τ_p/τ_n and α/β denotes the similarity, temperature and absolute margin for positive/negative pairs, respectively.

$$\mathcal{L}_{align} = \frac{2}{B} \sum_{i=1}^B \left\{ \log \left[1 + e^{-\tau_p(\mathbf{S}_i^+ - \alpha)} \right] + \log \left[1 + e^{\tau_n(\mathbf{S}_i^- - \beta)} \right] \right\}. \quad (1)$$

Our consideration on the alignment loss is two folds: (1) Unlike triplet loss only considers the relative distances or CMPM [16] adopts KL divergence to associate the representations across different modalities in a batch, our alignment loss considers both relative and absolute distances between positive and negative pairs; (2) τ_p and τ_n can adjust the slope of the back propagation gradient according to 2, which will assign higher weights to more informative samples and then lower the risk of slow convergence or even model degeneration.

$$\frac{\partial \mathcal{L}_{align}}{\partial \mathbf{S}_i^+} = \frac{-\tau_p}{1 + e^{\tau_p(\mathbf{S}_i^+ - \alpha)}}, \quad \frac{\partial \mathcal{L}_{align}}{\partial \mathbf{S}_i^-} = \frac{\tau_n}{1 + e^{\tau_n(\beta - \mathbf{S}_i^-)}}. \quad (2)$$

B.4 Identity loss

We also regard identity classification with N labels as an auxiliary task. Cross entropy loss 3 is adopted here to assist the learning of instance discriminative features. $\mathbf{W} \in \mathbb{R}^{D \times N}$ denotes a shared projection matrix following visual and textual streams. Because person identities in the testing set do not appear in the training set, it is of importance to prevent the model from overfitting to the training identities. To this end, we replace the original one-hot label of each identity with a softer version by means of Label Smooth (LS) [9, 13] with the smooth factor $\varepsilon = 0.1$.

$$\mathcal{L}_{id} = \frac{1}{B} \sum_{i=1}^B -\log \left(\frac{e^{\mathbf{W}_{id_i}^\top \mathbf{V}_i^q}}{\sum_j^N e^{\mathbf{W}_j^\top \mathbf{V}_i^q}} \right) + \frac{1}{B} \sum_{i=1}^B -\log \left(\frac{e^{\mathbf{W}_{id_i}^\top \mathbf{T}_i^q}}{\sum_j^N e^{\mathbf{W}_j^\top \mathbf{T}_i^q}} \right). \quad (3)$$

B.5 Rerank post-processing

In the inference stage, only two q-encoders are used. We also incorporate the multimodal k -reciprocal rerank algorithm proposed in NAFS [4] into our post-processing to further improve the performance. For the text-to-image task, the initial ranking list is obtained by sorting the cross-modal cosine similarity calculated by the text query t and each gallery image v . For each image v , the k -nearest neighboring images are obtained with the visual unimodal cosine similarity, denoted as $N_{i2i}(v, k)$. Similarly, the nearest image neighbors for the textual query $N_{i2i}(t, k)$ are obtained based on the cross-modal similarity. Finally, the pair-wise rerank similarity $D_J(v, t)$ is calculated by Jaccard Distance and added to the original cosine similarity with a weight of 0.05. For the image-to-text task, we extend this formula in a symmetrical manner to obtain $D_J(t, v)$.

$$D_J(v, t) = 1 - \frac{N_{i2i}(v, k) \cap N_{i2i}(t, k)}{N_{i2i}(v, k) \cup N_{i2i}(t, k)}, \quad D_J(t, v) = 1 - \frac{N_{i2i}(t, k) \cap N_{i2i}(v, k)}{N_{i2i}(t, k) \cup N_{i2i}(v, k)}. \quad (4)$$

C More evaluation results

C.1 The model pre-trained on MSCOCO

There are many other available models [1, 6, 8, 14] pre-trained on large-scale generic image-text pairs [7, 10]. However, we choose the experiment settings used in VSE++ [3] to prepare our comparative experiments. Our consideration is two-fold: (1) VSE++ is designed in a two-stream manner, which guarantees a high inference speed for TBPS; (2) No detection module, *e.g.*, Faster-RCNN [12], is used in VSE++, leading to a more fair comparison. We change the triplet loss used in VES++ into our alignment loss and no hard example mining is used. The results of our model can be found in Table 1.

Model	Trainset	Image Retrieval			Caption Retrieval		
		R@1	R@5	R@10	R@1	R@5	R@10
VSE++ (VGG19, GRU, FT)	RC+rV	24.1	52.8	66.2	32.9	61.7	74.7
VSE++ (ResNet152, GRU, FT)	RC+rV	30.3	59.4	72.4	41.3	71.1	81.2
Ours (ResNet101, BERT)	RC+rV	33.2	63.5	75.1	46.8	76.3	85.7

Table 1: Comparison between our pre-trained model and VSE++ [3] on MSCOCO [7]. All results are calculated in MSCOCO 5k test split. FT, RC and rV denote fine-tune, random crop and rest validation set, respectively. Please refer to the paper of VSE++ [3] for details.

C.2 Model size and retrieval efficiency

Table 2 shows the comparisons of model size and retrieval efficiency between our method and the previous state of the art. In addition to the higher retrieval performance, our method also has three advantages: (1) Our architecture, no matter is built upon ResNet50 or 101, has much fewer parameters than those of other methods because of the single-scale architecture. A smaller model size leads to less GPU memory usage and faster training speed. (2) Our method has the fastest retrieval time because only global features are used during retrieval. This advantage can guarantee real-time retrieval and thus is friendly to practical deployment. (3) Our method has the least offline feature storage because we do not need to store local

information and our features’ embedding dimension (256) is quite smaller than that of NAFS (768) and TIPCB (2048). Small storage usage is crucial for practical cases with scaled-up data, otherwise it will increase the burden of the whole system and the cost of computing.

Model	Rank-1 \uparrow	Params (M) \downarrow	Retrieval Time (s) \downarrow	Offline Feature Storage \downarrow	
				Visual Side	Textual Side
ViTAA [15]	54.92	176.53	0.02	3MB	6MB
NAFS [4]	59.36	188.75	0.07	9MB	18MB
TIPCB [2]	63.63	184.75	0.20	24MB	48MB
Ours (ResNet50)	61.65	42.33	0.02	3MB	6MB
Ours (ResNet101)	64.08	60.20	0.02	3MB	6MB

Table 2: Comparisons of model size and retrieval efficiency among ViTAA [15], NAFS [4] and our method. Retrieval time is computed by retrieving all text queries (6156) through the whole image gallery (3074) of CUHK-PEDES test set [5].

D More visualization results

D.1 Visualization of self-attention pooling

Figure 2 visualizes the learned attention weight in the self-attention pooling layer of CLIP Image Encoder (ResNet101 version). We can conclude that the visual stream is capable of learning the salient parts related to the garments of a person rather than the background. This visualization further verifies that the model has the ability to learn reasonable features even without the help of multi-scale information.

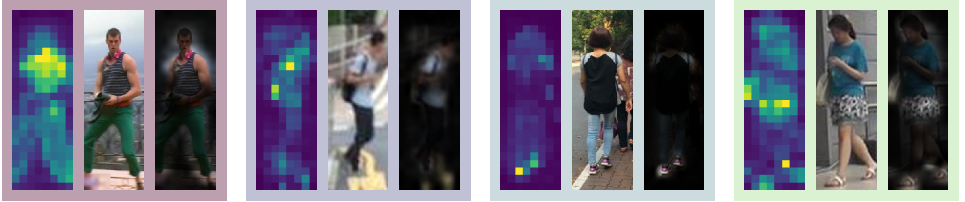


Figure 2: Visualization of the last layer attention map calculated by [CLS] token and other patch tokens in the self-attention pooling layer of CLIP ResNet101. This figure contains attention maps, original images and images multiplied by resized attention map for four different identities randomly sampled from test set.

D.2 More visualized retrieval results

We visualize several typical successful and failure cases of our retrieval results in Figure 3 and 4, respectively. It is apparent that this failure cases are due to the ambiguity in the images or the pragmatic vagueness in the sentences. The predictions of our model are reasonable, and the more specific the search sentence is, the better our search results will be.



Figure 3: Typical successful cases of retrieval results.



Figure 4: Typical failure cases of retrieval results.

References

- [1] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Universal image-text representation learning. In *ECCV*, 2020.
- [2] Yuhao Chen, Guoqing Zhang, Yujiang Lu, Zhenxing Wang, Yuhui Zheng, and Ruili Wang. Tipcb: A simple but effective part-based convolutional baseline for text-based person search. *arXiv preprint*, 2021.
- [3] Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. Vse++: Improving visual-semantic embeddings with hard negatives. In *BMVC*, 2018.
- [4] Chenyang Gao, Guanyu Cai, Xinyang Jiang, Feng Zheng, Jun Zhang, Yifei Gong, Pai Peng, Xiaowei Guo, and Xing Sun. Contextual non-local alignment over full-scale representation for text-based person search. *arXiv preprint*, 2021.
- [5] Shuang Li, Tong Xiao, Hongsheng Li, Bolei Zhou, Dayu Yue, and Xiaogang Wang. Person search with natural language description. In *CVPR*, 2017.
- [6] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, *et al.* Oscar: Object-semantics aligned pre-training for vision-language tasks. In *ECCV*, 2020.
- [7] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.
- [8] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*, 2019.
- [9] Hao Luo, Wei Jiang, Youzhi Gu, Fuxu Liu, Xingyu Liao, Shenqi Lai, and Jianyang Gu. A strong baseline and batch normalization neck for deep person re-identification. *IEEE TMM*, 2019.
- [10] Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *ICCV*, 2015.
- [11] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, *et al.* Learning transferable visual models from natural language supervision. In *ICML*, 2021.
- [12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE TPAMI*, 2016.
- [13] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.
- [14] Hao Tan and Mohit Bansal. LXMERT: Learning cross-modality encoder representations from transformers. In *EMNLP-IJCNLP*, 2019.

-
- [15] Zhe Wang, Zhiyuan Fang, Jun Wang, and Yezhou Yang. Vitaa: Visual-textual attributes alignment in person search by natural language. In *ECCV*, 2020.
 - [16] Ying Zhang and Huchuan Lu. Deep cross-modal projection learning for image-text matching. In *ECCV*, 2018.