QS-Attn: Query-Selected Attention for Contrastive Learning in I2I Translation

Xueqi Hu[1], Xinyue Zhou[1], Qiusheng Huang[1], Zhengyi Shi[1], Li Sun[1,2]*, Qingli Li[1]
[1]Shanghai Key Laboratory of Multidimensional Information Processing,
[2]Key Laboratory of Advanced Theory and Application in Statistics and Data Science,
East China Normal University, Shanghai, China

A. Limitations

We now discuss the limitation mainly existing in the experiments. Our goal is to select the significant features which are domain-specific for contrastive learning, therefore, we choose CUT as a baseline. But we do not implement our QS-Attn module in other I2I models due to limited training resources, such as bi-directional [10] and multi-domain I2I tasks [1, 2], although we think it should also work in them. Moreover, the training speed of our model is slower than CUT due to the complex matrix multiplication in global attention, but our model with local attention mitigates this problem, refer to Appendix C for details.

B. Network Architectures

In this section, we provide the network structure of our method, including the generator and discriminator. The detailed architectures of them are illustrated in Appendix B.1 and Appendix B.2.

B.1. Generator

Our generator G consists of two down-sampling blocks, nine intermediate blocks, two up-sampling blocks and two convolution layers for input and output. We apply Instance Normalization (IN) [8] and ReLU [5] in the generator, except for the output layer, which uses Tanh as the activation function. The network before the sixth residual block is regarded as the encoder E and the rest is the decoder. We adopt the multi-layer feature extraction in CUT, which takes the features from five layers. including the input image, the first and second down-sampling blocks, and the first and fifth residual blocks.

B.2. Discriminator

We apply a PatchGAN [7] discriminator in our model, which contains three down-sampling blocks and two convolution layers. Leaky ReLU [5] is employed as the activation in the discriminator.

C. Training details

Under three settings of QS-Attn, *Global*, *Local*, and *Local+Global*, we train the models for 400 epochs with the batch size of 1, the initial learning rate is set to 2×10^{-4}, and linearly decays to 0 in the last 200 epochs. All networks are optimized by the Adam optimizer [4], in which $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

D. Query-Selection Algorithm

There are three query-selection algorithm we proposed: *Global*, *Local*, and *Local+Global*. The pseudo-code in PyTorch style is provided in Algorithm 1,2 and 3.

E. Additional Ablation Study

In the previous discussion, we placed the emphasis on the query selection and cross domain value routing. Furthermore, in this section, we conduct an additional ablation study of applying our QS-Attn (Global) module in different layers. Quantitative results are illustrated in Appendix E, and visual results are shown in Fig. 1. Model **A** is our initial model, we only employ the QS-Attn on the last two layers, and randomly select points on the first three layers. In model **B**, we intend to utilize features from all lay-

Algorithm 1 QS-Attn (Global)

```
# H: height, W: width, C: dimension, N: number of selected queries
# feat: input tensor (H, W, C)

feat_q = feat.flatten(0, 1)
feat_k = feat_q.permute(1, 0)
dots_global = torch.bmm(feat_q, feat_k)
attn_global = dots_global.softmax(dim=1)
prob = -torch.log(attn_global)
prob = torch.where(torch.isinf(prob), torch.full_like(prob, 0), prob)
entropy = torch.sum(torch.mul(attn_global, prob), dim=1)
_, index = torch.sort(entropy)
patch_id = index[:N]
attn_QS = attn_global[patch_id, :]
feat_out = torch.bmm(attn_QS, feat_q)
```

Algorithm 2 QS-Attn (Local)

```
# H: height, W: width, C: dimension, N: number of selected queries, k: window size
# feat: input tensor (H, W, C)

feat_local = F.unfold(feat, kernel_size=k, stride=1, padding=k//2)
L = feat_local.shape[1] # L = H * W
feat_k = feat_local.permute(1, 0).reshape(L, k * k, C)
feat_q = feat.reshape(L, C, 1)
dots_local = torch.bmm(feat_k, feat_q).squeeze(-1)
attn_local = dots_local.softmax(dim=1)
prob = -torch.log(attn_local)
prob = torch.where(torch.isinf(prob), torch.full_like(prob, 0), prob)
entropy = torch.sum(torch.mul(attn_local, prob), dim=1)
_, index = torch.sort(entropy)
patch_id = index[:N]
attn_QS = attn_local[patch_id, :].unsqueeze(1)
feat_v = feat_k[index, k * k, C]
feat_out = torch.bmm(attn_QS, feat_q).squeeze(1)
```

Algorithm 3 QS-Attn (Local+Global)

```
# H: height, W: width, C: dimension, N: number of selected queries, k: window size
# feat: input tensor (H, W, C)

# Get patch_id from local attention.
feat_local = F.unfold(feat, kernel_size=k, stride=1, padding=k//2)
L = feat_local.shape[1] # L = H * W
feat_k = feat_local.permute(1, 0).reshape(L, k * k, C)
feat_q = feat.reshape(L, C, 1)
dots_local = torch.bmm(feat_k, feat_q).squeeze(-1)
attn_local = dots_local.softmax(dim=1)
prob = -torch.log(attn_local)
prob = torch.where(torch.isinf(prob), torch.full_like(prob, 0), prob)
entropy = torch.sum(torch.mul(attn_local, prob), dim=1)
_, index = torch.sort(entropy)
patch_id = index[:N]

# Select N rows in global attention matrix to route value.
feat_q_global = feat.flatten(0, 1)
feat_k_global = feat_q.permute(1, 0)
dots_global = torch.bmm(feat_q, feat_k)
attn_global = dots_global.softmax(dim=1)
attn_QS = attn_global[patch_id, :]
feat_out = torch.bmm(attn_QS, feat_q_global)
```

ers. However, the spatial dimension of shallow features is large, hence the computation of global attention matrix $A_g \in \mathbb{R}^{HW \times HW}$ is expensive and time-consuming. Take the high demand of computing into consideration, we employ the average pooling to reduce the spatial size of the feature maps in first three layers. Then, the spatial size of features from all layers is set to 64×64, thus $A_g \in \mathbb{R}^{4096 \times 4096}$. Model **B** can compete with model **A** on the two tasks. Although the average pooling decreases the computing effort, the calculation of A_g and entropy still occupies a great quantity of memory and time. As a result, Our model only exerts the QS-Attn on two layers, which is more lightweight and effective.

Method	Num of queries	FID↓	SWD↓
QS-Attn	64	47.9	30.1
CUT		52.6	33.4
QS-Attn	128	**40.9**	**28.5**
CUT		46.5	33.8
QS-Attn	256	41.1	30.3
CUT		45.5	31.5
QS-Attn	512	45.6	31.1
CUT		53.5	32.8

Table 1. Ablation study for number of queries using in QS-Attn and CUT.

Method	Layers	Cat→Dog		Horse→Zebra	
		SWD ↓	FID ↓	SWD ↓	FID ↓
A	last 2	12.8	**72.8**	**30.3**	41.1
B	all	**12.6**	73.1	32.5	**40.4**

Table 2. Ablation study of different layers for QS-Attn. Model A is our initial model, which applies the QS-Attn on the last 2 layers. Model B involves all the five layers for QS-Attn. The best performance is indicated in **bold**.

Moreover, We realize that intentionally selecting query may reduce the required number. So we train QS-Attn (Global) with 64, 128 or 512 queries on *Horse* → *Zebra* dataset, and compare its performance with CUT. Tab. 1 demonstrates that the selection of queries is always valid regardless of the number of queries. Moreover, when selecting 128 queries, the performance of QS-Attn is better than which under our original setting. It proves that entropy selection strategy is effective from another perspective.

F. More Results

Numerous synthesis results of our model on *Cityscapes*, *Cat* → *Dog* and *Horse* → *Zebra* datasets are shown below. Besides, the visual results of baselines are also listed, including FSeSim [9], CUT [6], CycleGAN [10] and MUNIT [3]. Fig. 2, Fig. 3, Fig. 4 illustrate the qualitative results on *Cityscapes*, *Cat* → *Dog* and *Horse* → *Zebra* datasets, respectively.

References

[1] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on*

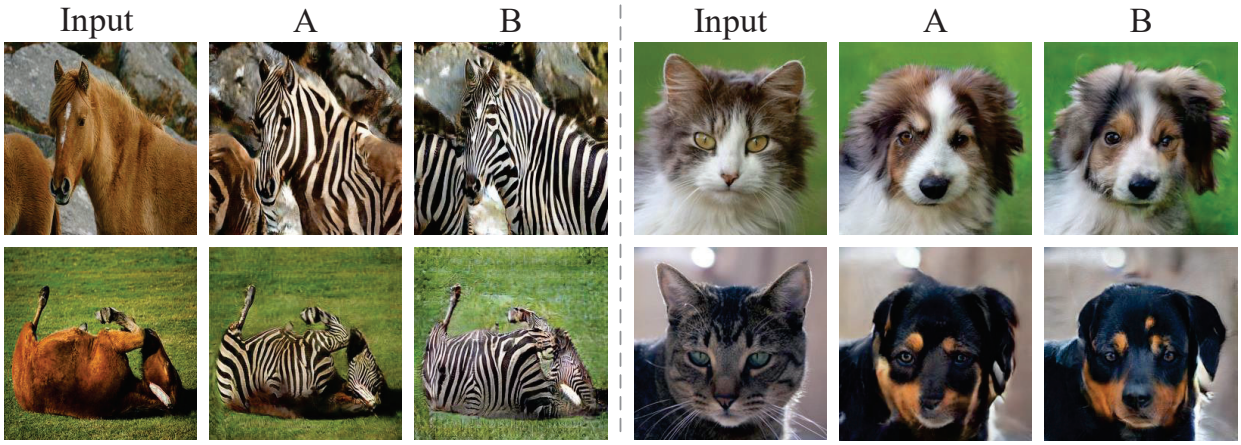

Figure 1. Ablation results of different layers for QS-Attn on two datasets.

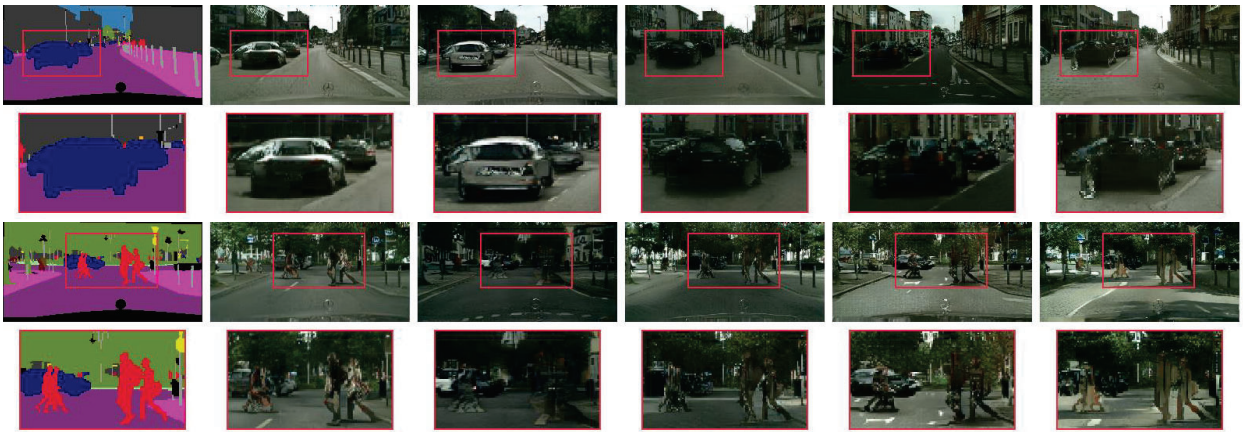

Figure 2. Visual results on *Cityscapes* compared with baselines. The leftmost column are the input source images. In the remaining columns, from left to right, are the translated results of our model, FSeSim, CUT, CycleGAN and MUNIT, respectively.

computer vision and pattern recognition, pages 8789–8797, 2018. 1

[2] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1

[3] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–189, 2018. 2

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[5] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013. 1

[6] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *European Conference on Computer Vision*, pages 319–345. Springer, 2020. 2

[7] Jaemin Son, Sang Jun Park, and Kyu-Hwan Jung. Retinal vessel segmentation in fundoscopic images with generative adversarial networks. *arXiv preprint arXiv:1706.09318*, 2017. 1

[8] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 1

[9] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. The spatially-correlative loss for various image translation tasks. In *CVPR*, 2021. 2

[10] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 1, 2

Figure 3. Visual results on *Cat* $\rightarrow$ *Dog* compared with baselines. The leftmost column are the input source images. In the remaining columns, from left to right, are the translated results of our model, FSeSim, CUT, CycleGAN and MUNIT, respectively.

Figure 4. Visual results on *Horse* $\rightarrow$ *Zebra* compared with baselines. The leftmost column are the input source images. In the remaining columns, from left to right, are the translated results of our model, FSeSim, CUT, CycleGAN and MUNIT, respectively.