1. Ablation Studies On Data Selection

In this section, we analyze the performance of our MSCN under different data selection strategies, i.e., GMM and BMM. The experiments are conducted on the Flickr30K with 20% noisy correspondence. The results in Tab.1 demonstrate that our MSCN is effective to the choice of data selection strategy. And our proposed meta-data guided method brings the best results.

Table 1. Ablation studies on Flickr30K with 20% noise rate.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Image to Text</th>
<th>Text to Image</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R@1</td>
<td>R@5</td>
</tr>
<tr>
<td>GMM</td>
<td>76.3</td>
<td>94.5</td>
</tr>
<tr>
<td>BMM</td>
<td>76.8</td>
<td>94.7</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>77.4</strong></td>
<td><strong>94.9</strong></td>
</tr>
</tbody>
</table>

2. Experiments on MS-COCO 5K

In this section, we evaluate our trained MSCN on the full 5K test set with different noise ratios, i.e., 20%, 50% and 70%. The results are shown in Tab.2

Table 2. Performance of MSCN on MS-COCO full 5K test set with 20%, 50% and 70% noise ratio.

<table>
<thead>
<tr>
<th>Noisy Ratio</th>
<th>Image to Text</th>
<th>Text to Image</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R@1</td>
<td>R@5</td>
</tr>
<tr>
<td>20%</td>
<td>57.1</td>
<td>84.0</td>
</tr>
<tr>
<td>50%</td>
<td>54.3</td>
<td>82.2</td>
</tr>
<tr>
<td>70%</td>
<td>51.4</td>
<td>80.3</td>
</tr>
</tbody>
</table>

3. Generating of Synthetic Noise

To evaluate our method on a range of noise ratios, we generate the synthetic noisy correspondence data from Flickr30K and MS-COCO. Specifically, we randomly generate a mismatched index list to construct noisy pairs. Here, we provide a pseudo-code to describe the generating of synthetic noisy data:

```python
def generate_noisy_correspondence(images, captions, data_length, noise_ratio):
    t2i_index = np.arange(0, data_length)
    # random produce mismatched idxs
    idx = np.arange(data_length)
    np.random.shuffle(idx)
    noise_length = int(noise_ratio * data_length)
    shuffle_index = t2i_index[idx[:noise_length]]
    np.random.shuffle(shuffle_index)
    t2i_index[idx[:noise_length]] = shuffle_index
    # fixing captions, and using the mismatched idxs to get images
    images = images[t2i_index]
    captions = captions
    return images, captions
```

4. Training Algorithm of MSCN

Algorithm 1 summarizes our proposed MSCN.
Algorithm 1: The MSCN Training Algorithm

Input: Training set $D_{\text{train}}$, meta-data set $D_{\text{meta}}$, models $M^{(1)} = \{F^{(1)}_W, V^{(1)}_{\Theta}\}$ and $M^{(2)} = \{F^{(2)}_W, V^{(2)}_{\Theta}\}$, batch size $n$ and $m$, learning rate $\alpha$ and $\beta$.

1. $M^{(1)}, M^{(2)} \leftarrow \text{WarmUp}(D_{\text{train}}, M^{(1)}, M^{(2)})$

2. while $e < \text{MaxEpoch}$ do

3. Construct negative meta-data and extend the meta-data set as $D'_{\text{meta}}$.

4. $\{S^{(1)}_p, S^{(1)}_N\} \leftarrow \text{GetSimilarityScore}(D'_{\text{meta}}, M^{(1)})$.

5. $\{S^{(2)}_p, S^{(2)}_N\} \leftarrow \text{GetSimilarityScore}(D'_{\text{meta}}, M^{(2)})$.

6. Initialize $BMM^{(1)}$ using $\{S^{(1)}_p, S^{(1)}_N\}$.

7. Initialize $BMM^{(2)}$ using $\{S^{(2)}_p, S^{(2)}_N\}$.

8. $P^{(2)} = \{p_i\}_{i=1}^N \leftarrow BMM^{(1)}(D_{\text{train}}, M^{(1)})$.

9. $P^{(1)} = \{p_i\}_{i=1}^N \leftarrow BMM^{(2)}(D_{\text{train}}, M^{(2)})$.

10. for $K = 1, 2$ do

11. $D'_{\text{train}}^{(K)} = \{(I_i, T_i) | p_i > 0.5, \forall (I_i, T_i, p_i) \in (D_{\text{train}}, P^{(K)})\}$

12. while $t < \text{MaxIteration}$ do

13. From $D'_{\text{train}}^{(K)}$ sample a training mini-batch $\{(I_i, T_i)\}_{i=1}^n$.

14. From $D'_{\text{meta}}$ sample a meta mini-batch $\{(I_i, T_i, y_i)\}_{i=1}^m$.

15. Compute the updated parameters for $M^{(K)}$ with training batch:

16. $W^{(t+1)}(\Theta) = W^{(t)} - \alpha \sum_{i=1}^n \nabla_{W_{\text{train}}} (I_i, T_i)\big|_{W^{(t)}}$.

17. Update the meta-net $V^{(K)}_{\Theta}$ with meta batch:

18. $\Theta^{(t+1)} = \Theta^{(t)} - \beta \frac{1}{m} \sum_{i=1}^m \nabla_{\Theta_{\text{meta}}} (I_i, T_i, y_i)\big|_{\Theta^{(t)}}$.

19. Update the main net $F^{(K)}_W$ parameters with training batch:

20. $W^{(t+1)} = W^{(t)} - \alpha \sum_{i=1}^n \nabla_{W_{\text{train}}} (I_i, T_i)\big|_{W^{(t)}}$.

21. end

22. end

end